UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/019,948	05/06/2025	11695823	ENS2-RE90019948	2943

3624 7590 08/05/2025

VOLPE KOENIG
30 SOUTH 17TH STREET, 18TH FLOOR
PHILADELPHIA, PA 19103

EXAMINER	
PEIKARI, BEHZAD	

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
08/05/2025	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
001

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

DO NOT USE IN PALM PRINTER

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

Amundsen Davis, LLC
111 East Kilbourn Avenue
Suite 1400
Milwaukee, WI 53202

EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. *90/019,948* .

PATENT UNDER REEXAMINATION *11695823* .

ART UNIT *3992* .

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

/PMartin, Paralegal/

PTOL-465 (Rev.07-04)

Ex Parte Reexamination Interview Summary – Pilot Program for Waiver of Patent Owner's Statement	Control No. 90/019,948	Patent Under Reexamination is Requested 11695823	
	Examiner Behzad Peikari	Art Unit 3992	AIA (FITF) Status Yes

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address. --

All participants (USPTO official and patent owner):

(1) Patricia Martin (3) _____

(2) Christopher Horgan, 40394 (4) _____

Date of Telephonic Interview: 23 July 2025.

A. The USPTO official requested waiver of the patent owner's statement pursuant to the pilot program for waiver of patent owner's statement in *ex parte* reexamination proceedings.*

☐ The patent owner **agreed** to waive its right to file a patent owner's statement under 35 U.S.C. 304 in the event reexamination is ordered for the above-identified patent.

☑ The patent owner **did not agree** to waive its right to file a patent owner's statement under 35 U.S.C. 304 at this time.

☐ USPTO personnel were unable to reach the patent owner.**

B. The Patent Owner of record telephoned the Office and indicated they would like to participate in the pilot program for waiver of patent owner's statement in *ex parte* reexamination proceedings.*

☐ The Patent owner of record telephoned the Office and **agreed** to waive its right to file a patent owner's statement under 35 U.S.C. 304 in the event reexamination is ordered for the above-identified patent.

The patent owner is not required to file a written statement of this telephone communication under 37 CFR 1.560(b) or otherwise. However, any disagreement as to this interview summary must be brought to the immediate attention of the USPTO, and no later than one month from the mailing date of this interview summary. Extensions of time are governed by 37 CFR 1.550(c).

*For more information regarding this pilot program, see *Pilot Program for Waiver of Patent Owner's Statement in Ex Parte Reexamination Proceedings, 75 Fed. Reg.* 47269 (August 5, 2010), available on the USPTO Web site at http://www.uspto.gov/patents/law/notices/2010.jsp.

**The patent owner may contact the USPTO personnel at (571) 272-7705 or at the telephone number provided below if the patent owner decides to waive the right to file a patent owner's statement under 35 U.S.C. 304.

/Patricia Martin/
Paralegal Specialist, OCRU (571) 272-5004
Signature and telephone number of the USPTO official, who contacted, was contacted by, or attempted to contact the patent owner.

cc: Requester (if third party requester)

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
003

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/019,948	05/06/2025	11695823	UNP-16002	2943

3624 7590 07/22/2025

VOLPE KOENIG
30 SOUTH 17TH STREET, 18TH FLOOR
PHILADELPHIA, PA 19103

EXAMINER	
PEIKARI, BEHZAD	

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
07/22/2025	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
004

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

DO NOT USE IN PALM PRINTER

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

Amundsen Davis, LLC
111 East Kilbourn Avenue
Suite 1400
Milwaukee, WISCONSIN 53202

EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM

REEXAMINATION CONTROL NO. *90/019,948* .

PATENT UNDER REEXAMINATION *11695823* .

ART UNIT *3992* .

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

Order Granting Request For Ex Parte Reexamination	Control No.	Patent Under Reexamination	
	90/019,948	11695823	
	Examiner	Art Unit	AIA (FITF) Status
	BEHZAD PEIKARI	3992	Yes

--The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

The request for *ex parte* reexamination filed 05/06/2025 has been considered and a determination has been made. An identification of the claims, the references relied upon, and the rationale supporting the determination are attached.

Attachments: a)☐ PTO-892, b)☑ PTO/SB/08, c)☐ Other: ____

1. ☑ The request for *ex parte* reexamination is GRANTED.

RESPONSE TIMES ARE SET AS FOLLOWS:

For Patent Owner's Statement (Optional): TWO MONTHS from the mailing date of this communication (37 CFR 1.530 (b)). **EXTENSIONS OF TIME ARE GOVERNED BY 37 CFR 1.550(c).**

For Requester's Reply (optional): TWO MONTHS from the **date of service** of any timely filed Patent Owner's Statement (37 CFR 1.535). **NO EXTENSION OF THIS TIME PERIOD IS PERMITTED.** If Patent Owner does not file a timely statement under 37 CFR 1.530(b), then no reply by requester is permitted.

/B. JAMES PEIKARI/ Primary Examiner, Art Unit 3992		

cc:Requester (if third party requester)

ORDER GRANTING REQUEST FOR *EX PARTE* REEXAMINATION

The request filed by a third party requester on May 6, 2025 alleges that there is at least one substantial new question of patentability (SNQ) affecting claims 1-4, 8, 15 and 19 of U.S. Patent Number 11,695,823 (the '823 patent).

For the reasons explained below, the request for *ex parte* reexamination is **GRANTED**.

Patent Owner's Statement

Patent owner has not requested and has not agreed to waive the right to file a patent owner's statement under 35 U.S.C. 304 at this time.

Information Disclosure Statement

With respect to the Information Disclosure Statement(s) (PTO/SB/08A and 08B or its equivalent) attached hereto, the information cited thereon has been considered to the extent set forth in MPEP §§ 2256 and 2656.

MPEP §§ 2256 and 2656 indicate that degree of consideration to be given to such information will be normally limited by the degree to which the party filing the information citation has actually explained the (1) content and (2) relevance of the information.

References

The alleged SNQs are based on the following references:

(1) U.S. Patent Application Publication No. 2012/0278454 to Stewart et al. ("Stewart")

(2) U.S. Patent Application Publication No. 2013/0054682 to Malik et al. ("Malik'').

Extrinsic Evidence:

"The Relational and Network Approaches: Comparison of the Application Programming Interfaces," C. J. Date and E. F. Dodd. SIGFIDET. 1974. pages 83-113. (cited to show an early use of the term "API", i.e., application programming interfaces).

Note: None of the above references were made of record during the prosecution of the '823 patent.

Note: The Declaration of James T. Geier has been considered in this Determination.

Brief Overview of the Patent

The '823 patent is directed to a system with a programmable network device hosting network device applications and a programmable cloud device hosting cloud applications, such that the network device applications and the cloud applications are in secure communication with each other to form distributed applications.

The network device applications and the cloud applications device work together to enable a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware.

Note Figure 16 below:

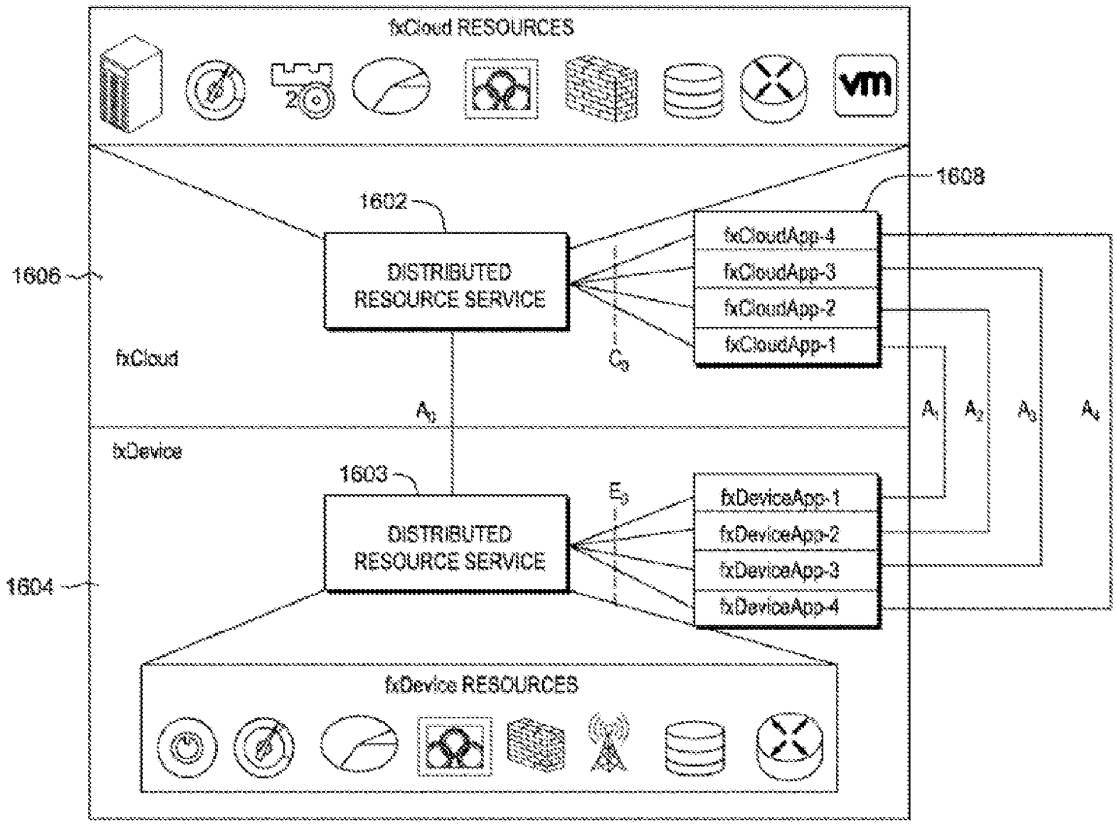


FIG. 16

Prosecution History

The '823 patent matured from U.S. Patent Application No. 17/142,983 (the '983 application), filed on January 6, 2021.

The '823 patent claims priority through a string of continuations to a provisional application filed June 13, 2013.

The relevant prosecution history of the '983 application may be summarized as follows:

- The '983 application was filed on January 6, 2021, with claims 1-18.

- On April 19, 2021, corrected application papers were filed, including a new set of claims, 1-20. However, these may have been included in error, as they were not mentioned again by the examiner or by patent owner during the remainder of the prosecution.
- On January 5, 2022, a non-final Office action was mailed wherein claims 1-18 were rejected under statutory and non-statutory double patenting.
- On February 13, 2022, a response was filed wherein terminal disclaimers were filed.
- On April 8, 2022, a final Office action was mailed wherein claims 1-18 were rejected under statutory double patenting.
- On July 25, 2022, a request for continued examination (RCE) was filed wherein claims 1-18 were cancelled and new claims 19-38 were added.
- On September 13, 2022, a non-final Office action was mailed wherein claims 19-38 were rejected under various statutes.
- On September 19, 2022, an amendment was filed wherein claims 19-36 were amended, claims 37 and 38 were cancelled, and new claim 39 was added.
- On December 16, 2022, a final Office action was mailed wherein claims 19-36 and 39 were rejected under 35 U.S.C. 112.
- On April 17, 2022, a request for continued examination (RCE) was filed wherein claims 19, 22- 24, 30 and 39 were amended.
- On May 9, 2023, a Notice of Allowance was mailed, indicating the allowability of claims 19-36 and 39.

- No reasons for allowance were included in the Notice. However, in the non-final rejection of September 13, 2022, the examiner provided reasons for the allowability of claims 19-36 as follows:

  "The prior art does not teach or render obvious to one of ordinary skill in the art, before the earliest effective filing date of the claimed invention, in the specific combinations and manner recited within the claims, the features of:

  ...wherein the programmable network device and programmable cloud device form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) that allow for simultaneous programming of the programmable network device and programmable cloud device independent of network device hardware and cloud device hardware."

- On July 4, 2023, the '983 application issued as U.S. Patent Number 11,695,823.

SNQs Raised in the Request

The request proposed the following SNQs:

- **Issue 1:** A SNQ as to claims 1-4, 8, 15 and 19 as anticipated by Stewart.
- **Issue 2:** A SNQ as to claims 1-4, 8, 15 and 19 for obviousness over Stewart.
- **Issue 3:** A SNQ as to claims 1-4, 15 and 19 as anticipated by Malik.
- **Issue 4:** A SNQ as to claims 1-4, 15 and 19 for obviousness over Malik.

Discussion of SNQs

A prior art patent or printed publication raises a substantial new question of patentability where there is:

(A) a substantial likelihood that an Examiner would reasonably consider the prior art patent or printed publication important in deciding whether or not the claim is patentable; see MPEP §2242 (I), and

(B) the same question of patentability as to the claim has not been decided in a previous or pending proceeding or in a final holding of invalidity by a federal court; see MPEP §2242 (III).

For any reexamination ordered on or after November 2, 2002, reliance on previously cited/considered art, i.e., "old art," does not necessarily preclude the existence of a substantial new question of patentability that is based exclusively on that old art. Rather, determinations on whether a substantial new question of patentability exists in such an instance shall be based upon a fact-specific inquiry done on a case-by-case basis. See MPEP 2242.

Issues 1 & 2

A SNQ as to claims 1-4, 8, 15 and 19 view of Stewart.

Stewart is directed to a gateway device application development system wherein a gateway device may provide reported information to and receive commands from a cloud server, which may be in communication with other devices which may be used to view reported information of or control the managed devices.

In the Stewart system, the cloud server may also be configured to install applications, e.g., for execution by the cloud server, the gateway device, and/or other devices. The Stewart system determines a plurality of applications to install, provides those applications to the gateway device over a wide area network, then the gateway device executes those applications.

Note Stewart Figure 1B:

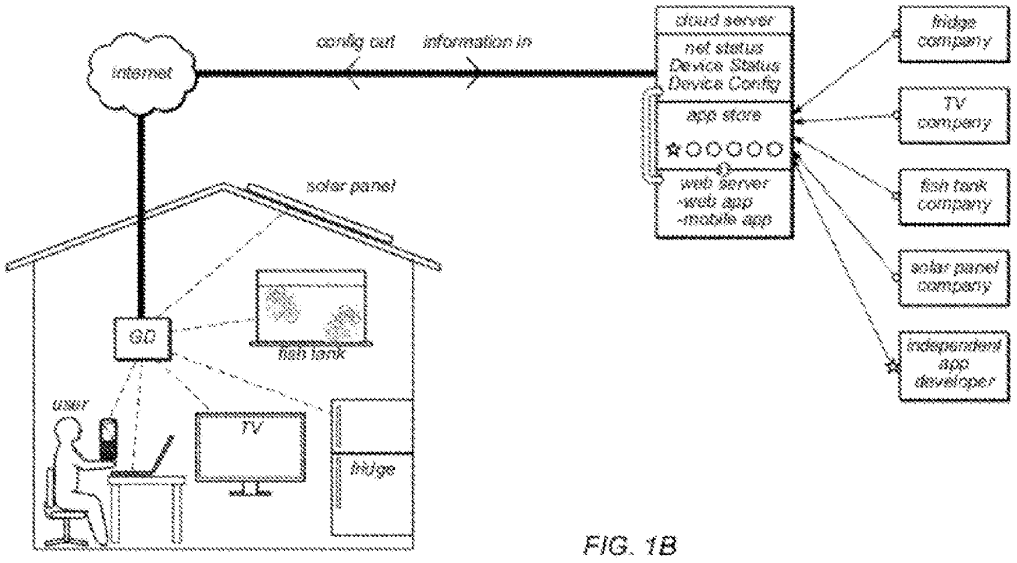


FIG. 1B

Thus, Stewart discloses features of claims 1-4, 8, 15 and 19 which the examiner considered and addressed during the original prosecution of the '823 patent, such as capabilities enabling a plurality of upper layer application programming interfaces (APIs) that allow for programming of the programmable network device and programmable cloud device.

There is a substantial likelihood that an examiner would consider the teachings of Stewart important in deciding the patentability of these claims.

Stewart was not made of record in the file of the patent and is not cumulative to the art of record in the original file. The teachings of Stewart were not subject to a final holding of invalidity by a federal court.

Accordingly, it is **agreed** that Stewart raises a SNQ as to claims 1-4, 8, 15 and 19.

Issues 3 & 4

A SNQ as to claims 1-4, 15 and 19 view of Malik.

Malik discloses deploying software modules using a device management platform and providing, at one or more server computers, a file server configured to host a plurality of feature modules, a module manager configured to maintain the plurality of feature modules, and a manifest server configured to maintain a plurality of "manifests". Each manifest conveys the feature module components that at least one endpoint should have installed.

The Malik system provides one or more agents to a plurality of endpoints, where the agents are configured to communicate with the one or more server computers to receive at least one manifest and subsequently receive a subset of the plurality of feature modules from the file server. An execution engine is configured to execute a plurality of scripts contained in the feature modules and the agents report data from the feature modules to the one or more servers.

Thus, Malik discloses features of claims 1-4, 15 and 19 which the examiner considered and addressed during the original prosecution of the '823 patent, such as capabilities enabling a plurality of upper layer application programming interfaces (APIs) that allow for programming of the programmable network device and programmable cloud device.

There is a substantial likelihood that an examiner would consider the teachings of Malik important in deciding the patentability of these claims.

Malik was not made of record in the file of the patent and is not cumulative to the art of record in the original file. The teachings of Malik were not subject to a final holding of invalidity by a federal court.

Accordingly, it is **agreed** that Malik raises a SNQ as to claims 1-4, 15 and 19.

35 USC 325(d)

35 USC 325(d) states in part that "[I]n determining whether to institute or order a proceeding under this chapter, chapter 30, or chapter 31, the Director may take into account whether, and reject the petition or request because, the same or substantially the same prior art or arguments previously were presented to the Office." Thus, pursuant to the statute, in order for the Director to exercise discretion as to whether to Order a reexamination proceeding under chapter 30, the request must first be determined to be based on the same or substantially the same prior art or arguments previously were presented to the Office.

A review of the post grant history of the '823 patent indicates that the patent was the subject of a single prior Office post grant challenge.

On February 17, 2025 petitioner, Microft Corporation filed a petition seeking *inter partes review* of claims 1-5, 7, 8, 12-15, 18 and 19 of the '823 patent (IPR2025-00618) which asserted the following grounds as raising an RLP:

IPR2025-00618

RLP #	Claims	Basis	Prior Art
1	1, 2, 12-15 and 19	35 USC § 103	Vasell, Alves, Rellermeyer
2	3-5, 7, 8 and 18	35 USC § 103	Vasell, Alves, Rellermeyer, Hall

As of the mailing date of this Order granting reexamination, the Office has yet to issue a decision on institution of the IPR2025-00618 petition.

A comparison between the prior art and arguments presented in the instant Request *for ex parte* reexamination of the '823 patent (90/019,948) indicates that the neither the prior art nor arguments are the same or substantially the same as the art or arguments presented in the prior pending *inter partes* review petition (IPR2025-00618).

As stated above, the instant request (90/019,948) asserts the following grounds as raising an SNQ to claims 1-4, 8, 15 and 19 of the '823 patent.

RLP #	Claims	Basis	Prior Art
1	1-4, 8, 15 and 19	35 USC § 102	Stewart
2	1-4, 15 and 19	35 USC § 103	Stewart
3	1-4, 15 and 19	35 USC § 102	Malik
4	1-4, 15 and 19	35 USC § 103	Malik

As an initial matter, none of the prior art references submitted in the current request (Stewart and Malik) were presented in the IPR2025-00618 petition. Further, the Stewart and Malik reference have different disclosures and are not cumulative to the prior art references presented in the pending IPR petition (Vasell, Alves, Rellermeyer, Hall). Therefore, the prior art and arguments presented in the Request are not the same or substantially the same as the prior art or arguments previously presented to the Office.

Accordingly, the statutory threshold for the Director to exercise discretion pursuant to 35 USC 325(d) is not met and the Request for reexamination is Ordered in view of the determination above that the Request raises a substantial new question of patentability as to claims 1-4, 8, 15 and 19 of the '823 patent.

Conclusion

Claims 1-4, 8, 15 and 19 are subject to reexamination.

Since Requester did not request reexamination of claims 5-7, 9-14 and 16-18 and did not assert the existence of a substantial new question of patentability (SNQ) for such claims (see 35 U.S.C. § 311 (b)(2); see also 37 CFR 1.915b and 1.923), such claims will not be reexamined.

This matter was directly addressed in *Sony Computer Entertainment America Inc., et al. v. Jon W. Dudas*, Civil Action No. 1:05CV1447 (E.D.Va. May 22, 2006), Slip Copy, 2006 WL 1472462. The District Court upheld the Office's discretion to not reexamine claims in an *inter partes* reexamination proceeding other than those claims for which reexamination had specifically been requested. The Court stated:

> "To be sure, a party may seek, and the PTO may grant, ... review of each and every claim of a patent. Moreover, while the PTO in its discretion may review claims for which ..... review was not requested, nothing in the statute compels it to do so. To ensure that the PTO considers a claim for ..... review, § 311 (b)(2) requires that the party seeking reexamination demonstrate why the PTO should reexamine each and every claim for which it seeks review. Here, it is undisputed that Sony did not seek review of every claim under the '213 and '333 patents. Accordingly, Sony cannot now claim that the PTO wrongly failed to reexamine claims for which Sony never requested review, and its argument that AIPA compels a contrary result is unpersuasive."

Waiver of Right to File Patent Owner Statement

In a reexamination proceeding, Patent Owner may waive the right under 37 C.F.R. 1.530 to file a Patent Owner Statement. The document needs to contain a

statement that Patent Owner waives the right under 37 C.F.R. 1.530 to file a Patent Owner Statement and proof of service in the manner provided by 37 C.F.R. 1.248, if the request for reexamination was made by a third party requester, see 37 C.F.R 1.550(f).

Service of Papers

After filing of a request for *ex parte* reexamination by a third party requester, any document filed by either the patent owner or the third party requester must be served on the other party (or parties where two or more third party requester proceedings are merged) in the reexamination proceeding in the manner provided in 37 CFR 1.248. The document must reflect service or the document may be refused consideration by the Office. See 37 CFR 1.550(f).

Extensions of Time

Extensions of time under 37 CFR 1.136(a) will not be permitted in these proceedings because the provisions of 37 CFR 1.136 apply only to "an applicant" and not to parties in a reexamination proceeding. Additionally, 35 U.S.C. 305 requires that *ex parte* reexamination proceedings "will be conducted with special dispatch" (37 CFR 1.550(a)). Extensions of time in *ex parte* reexamination proceedings are provided for in 37 CFR 1.550(c).

Litigation Reminder

The patent owner is reminded of the continuing responsibility under 37 CFR 1.565(a) to apprise the Office of any litigation activity, or other prior or concurrent

proceeding, involving the patent throughout the course of this reexamination proceeding. The third party requester is also reminded of the ability to similarly apprise the Office of any such activity or proceeding throughout the course of this reexamination proceeding. See MPEP §§ 2207, 2282 and 2286.

All correspondence relating to this *Ex Parte* reexamination proceeding should be directed:

By Mail to: Mail Stop *Ex Parte* Reexam
Attn: Central Reexamination Unit
Commissioner of Patents
United States Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

By FAX to: (571) 273-9900
Central Reexamination Unit

By hand to: Customer Service Window
Randolph Building
501 Dulany Street
Alexandria, VA 22314

To file and manage patent submissions in Patent Center, please visit https://patentcenter.uspto.gov. Visit https://www.uspto.gov/patents/apply/patent-center for more information about Patent Center and https://www.uspto.gov/patents/docx for information about filing in DOCX format.

For Patent Center transmissions, 37 CFR 1.8(a)(1)(i)(C) and (ii) states that correspondence (except for a request for reexamination and a corrected or replacement request for reexamination) will be considered timely filed if (a) it is transmitted via the Office's electronic filing system in accordance with 37 CFR 1.6(a)(4) , and (b) includes a certificate of transmission for each piece of correspondence stating the date of transmission, which is prior to the expiration of the set period of time in the Office action.

Any inquiry concerning this communication or earlier communications from the examiner, or as to the status of this proceeding, should be directed to B. James Peikari at telephone number 571-272-4185. The reexamination specialist can normally be reached on Mon-Fri from 8:30am to 5:30pm. If attempts to reach the specialist by telephone are unsuccessful, the examiner's supervisor, Alexander Kosowski, SPE Art Unit 3992, can be reached at 571-272-3744. General inquiries may be directed to the Central Reexamination Unit at telephone number 571-272-7705.

/B. James Peikari/
Reexamination Specialist, Art Unit 3992

Conferees:

/ADAM L BASEHOAR/
Primary Examiner, Art Unit 3992

/ALEXANDER J KOSOWSKI/
Supervisory Patent Examiner, Art Unit 3992

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,948	11695823
	Examiner	Art Unit
	BEHZAD PEIKARI	3992

CPC - Searched*		
Symbol	Date	Examiner

CPC Combination Sets - Searched*		
Symbol	Date	Examiner

US Classification - Searched*			
Class	Subclass	Date	Examiner

* See search history printout included with this form or the SEARCH NOTES box below to determine the scope of the search.

Search Notes		
Search Notes	Date	Examiner
Litigation Search Review	06/04/2025	BJP
PTACTS for AIA proceedings	06/04/2025	BJP
PTAB-WEB for PTAB decisions	06/04/2025	BJP
PDP (Patent Data Portal) for continuity data	06/04/2025	BJP
Reviewed the prosecution history of U.S. Patent No. 11,695,823; 10,893,095; 10,686,871; and 9,843,624	06/04/2025	BJP
Reviewed IPR2025-00618	06/04/2025	BJP

Interference Search			
US Class/CPC Symbol	US Subclass/CPC Group	Date	Examiner

/B. JAMES PEIKARI/ Primary Examiner, Art Unit 3992	

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
021

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,948	11695823
	Examiner	Art Unit
	BEHZAD PEIKARI	3992

/B. JAMES PEIKARI/ Primary Examiner, Art Unit 3992	

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
022

Reexamination	Application/Control No.	Applicant(s)/Patent Under Reexamination
	90/019,948	11695823
	Certificate Date	Certificate Number

Requester Correspondence Address: ☐ Patent Owner ☑ Third Party
Amundsen Davis, LLC 111 East Kilbourn Avenue Suite 1400 Milwaukee, WISCONSIN 53202

LITIGATION REVIEW ☑	BJP (examiner initials)	03 June 2025 (date)
Case Name		Director Initials
Edge Networking Systems, LLC v. Microsoft Corporation (W.D. Tex., Case No. 1:24cv215, Feb. 29, 2024) (open)		
Edge Networking Systems LLC v. Amazon.com, Inc., et al. (E.D. Tex., Case No. 2:24cv887, Nov. 1, 2024) (open)		

COPENDING OFFICE PROCEEDINGS	
TYPE OF PROCEEDING	**NUMBER**
Inter Partes Review	IPR2025-00618

/B. JAMES PEIKARI/ Primary Examiner, Art Unit 3992	

Doc code: IDS
Doc description: Information Disclosure Statement (IDS) Filed

PTO/SB/08 (01-25)
Approved for use through 11/30/2027. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

Application Number	17142983
Filing Date	2021-01-06
First Named Inventor	Pouya Taaghol
Art Unit	~~2441~~
Examiner Name	~~Unknown~~ Peikari
Attorney Docket Number	Unknown

U.S. PATENTS

Examiner Initial*	Cite No	Patent Number	Kind Code[1]	Issue Date	Name of Inventor of cited Document	Pages, Columns, Lines where Relevant Passages or Relevant Figures Appear
	1	5691897		1997-11-25	Roy-G-Biv Corporation	

If you wish to add additional U.S. Patent citation information, please click the Add button.

U.S. PATENT APPLICATION PUBLICATIONS

Examiner Initial*	Cite No	Publication Number	Kind Code[1]	Publication Date	Name of Inventor or Applicant of cited Document	Pages, Columns, Lines where Relevant Passages or Relevant Figures Appear
	1	20120278454		2012-11-01	Stewart et al.	
	2	20130054682		2013-02-28	Malik et al.	

If you wish to add additional U.S. Published Application citation information, please click the Add button.

FOREIGN PATENT DOCUMENTS

Examiner Initial*	Cite No	Foreign Document Nu	Country Code[2] i	Kind Code[4]	Publication Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines where Relevant Passages or Relevant Figures Appear	T[5]
	1							☐

If you wish to add additional Foreign Patent Document citation information, please click the Add button

NON-PATENT LITERATURE DOCUMENTS

PC.1.19.25

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /BP/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT		
	Application Number	17142983
	Filing Date	2021-01-06
	First Named Inventor	Pouya Taaghol
	Art Unit	~~2441~~ Peikari
	Examiner Name	~~Unknown~~
	Attorney Docket Number	Unknown

Examiner Initials*	Cite No	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc), date, pages(s), volume-issue number(s), publisher, city and/or country where published.	T[5]
	1	Declaration of James Geier, signed May 5, 2025, filed with Request for Ex Parte Reexamination, pp. 1-45.	☐
	2	"The Relational and Network Approaches: Comparison of the Application Programming Interfaces," C. J. Date and E. F. Dodd, SIGFIDET, 1974, pages 83-113.	☐

If you wish to add additional non-patent literature document citation information, please click the Add button

EXAMINER SIGNATURE

Examiner Signature	/B. JAMES PEIKARI/	Date Considered	06/04/2025

*EXAMINER: Initial if reference considered and whether or not citation is in conformance with MPEP 609. Draw line through a citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1] See Kind Codes of USPTO Patent Documents at www.USPTO.GOV or MPEP 901.04. [2] Enter office that issued the document, by the two-letter code (WIPO Standard ST.3). [3] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [4] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. [5] Applicant is to place a check mark here if English language translation is attached.

PC.1.19.25

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /BP/

INFORMATION DISCLOSURE STATEMENT BY APPLICANT		
	Application Number	17142983
	Filing Date	2021-01-06
	First Named Inventor	Pouya Taaghol
	Art Unit	~~2441~~
	Examiner Name	~~Unknown~~ Peikari
	Attorney Docket Number	Unknown

TIMING STATEMENT

Please see 37 CFR 1.97(e) to make the appropriate selection(s):

☐ That each item of information contained in the information disclosure statement was first cited in any communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of the information disclosure statement. See 37 CFR 1.97(e)(1).

OR

☐ That no item of information contained in the information disclosure statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to the knowledge of the person signing the certification after making reasonable inquiry, no item of information contained in the information disclosure statement was known to any individual designated in 37 CFR 1.56(c) more than three months prior to the filing of the information disclosure statement. See 37 CFR 1.97(e)(2).

☐ See attached timing statement under 37 CFR 1.97(e).

☒ A timing statement under 37 CFR 1.97(e) is not submitted herewith.

TIMING FEE

☐ The fee set forth in 37 CFR 1.17(p) has been submitted herewith.

COPIES

☐ An identification of an earlier application pursuant to 37 CFR 1.98(d)(1) is attached.

SIZE FEE ASSERTION

Please see 37 CFR 1.17(v) and the IDS Auto-Load Instructions for completing this form to make the appropriate selection of an assertion under 37 CFR 1.98. For the information disclosure statement (IDS) submitted herewith, the applicant or patent owner certifies the following with respect to the cumulative number of applicant-provided or patent-owner provided items of information submitted to date including those in the accompanying IDS (select only one):

☒ No IDS size fee is required under 37 CFR 1.17(v) at this time.

☐ The IDS is accompanied by the IDS size fee under 37 CFR 1.17(v)(1).

☐ The IDS is accompanied by the IDS size fee under 37 CFR 1.17(v)(2).

☐ The IDS is accompanied by the IDS size fee under 37 CFR 1.17(v)(3).

SIGNATURE

A signature of the applicant or representative is required in accordance with 37 CFR 1.33 and 11.18. Please see 37 CFR 1.4(d) for the form of the signature.

Signature	/John T. Pienkos/	Date (YYYY-MM-DD)	2025-05-06
Name/Print	John T. Pienkos	Registration Number	42,997

PC.1.19.25

INFORMATION DISCLOSURE STATEMENT BY APPLICANT		
	Application Number	17142983
	Filing Date	2021-01-06
	First Named Inventor	Pouya Taaghol
	Art Unit	~~2441~~
	Examiner Name	~~Unknown~~ Peikari
	Attorney Docket Number	Unknown

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 1 hour to complete, including gathering, preparing and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these record s.

2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.

3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.

4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).

5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.

6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).

7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.

8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.

9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
90/019,948	05/06/2025	11695823	UNP-16002

CONFIRMATION NO. 2943

112449
Roark IP
1438 Dahlia Loop
San Jose, CA 95126

POWER OF ATTORNEY NOTICE

OC000000090589162

Date Mailed: 06/25/2025

NOTICE REGARDING CHANGE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 06/23/2025.

- The Power of Attorney to you in this application has been revoked by the applicant. Future correspondence will be mailed to the new address of record(37 CFR 1.33).

Questions about the contents of this notice and the requirements it sets forth should be directed to the Office of Data Management, Application Assistance Unit, at **(571) 272-4000** or **(571) 272-4200** or **1-888-786-0101**.

/rbell/

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
90/019,948	05/06/2025	11695823	UNP-16002

CONFIRMATION NO. 2943

3624
VOLPE KOENIG
30 SOUTH 17TH STREET, 18TH FLOOR
PHILADELPHIA, PA 19103

POA ACCEPTANCE LETTER

OC000000090589207

Date Mailed: 06/25/2025

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 06/23/2025.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

Questions about the contents of this notice and the requirements it sets forth should be directed to the Office of Data Management, Application Assistance Unit, at **(571) 272-4000** or **(571) 272-4200** or **1-888-786-0101**.

/rbell/

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 2943

SERIAL NUMBER	FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
90/019,948	05/06/2025 RULE	709	3992	ENS2-RE90019948

APPLICANTS

Edge Networking Systems, LLC (PO), Residence Not Provided;
Unified Patents LLC (3rd Pty Req.), Chevy Chase, MD;

INVENTORS

11695823, Austin, TX;
Amundsen Davis, LLC, Residence Not Provided;
Pouya Taaghol, San Jose, CA;
Vivek Ramanna, San Jose, CA;
Edge Networking Systems LLC (Assignee), Austin, TX;

**** CONTINUING DATA ***************************

This application is a REX of 17/142,983 01/06/2021 PAT 11695823
which is a CON of 16/900,963 06/14/2020 PAT 10893095
which is a CON of 15/836,824 12/09/2017 PAT 10686871
which is a CON of 14/295,331 06/04/2014 PAT 9843624
which claims benefit of 61/834,807 06/13/2013

**** FOREIGN APPLICATIONS ***************************

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****

Foreign Priority claimed ❑ Yes ❑ No 35 USC 119(a-d) conditions met ❑ Yes ❑ No Verified and Acknowledged _____ Examiner's Signature	❑ Met after Allowance _____ Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
		TX		19	2

ADDRESS

VOLPE KOENIG
30 SOUTH 17TH STREET, 18TH FLOOR
PHILADELPHIA, PA 19103
UNITED STATES

TITLE

Distributed Software Defined Networking

FILING FEE RECEIVED		
0.00	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	❑ All Fees ❑ 1.16 Fees (Filing) ❑ 1.17 Fees (Processing Ext. of time) ❑ 1.18 Fees (Issue) ❑ Other _____ ❑ Credit

BIB (Rev. 05/07).

PTO/AIA/96 (08-12)
Approved for use through 01/31/2013. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(c)

Applicant/Patent Owner: EDGE NETWORKING SYSTEMS, LLC

Application No./Patent No.: 90/019,948 Filed/Issue Date: May 6, 2025

Titled: DISTRIBUTED SOFTWARE DEFINED NETWORKING

EDGE NETWORKING SYSTEMS, LLC, a CORPORATION

(Name of Assignee) (Type of Assignee, e.g., corporation, partnership, university, government agency, etc.)

states that, for the patent application/patent identified above, it is (choose **one** of options 1, 2, 3 or 4 below):

1. ☑ The assignee of the entire right, title, and interest.

2. ☐ An assignee of less than the entire right, title, and interest (check applicable box):

   ☐ The extent (by percentage) of its ownership interest is _____%. Additional Statement(s) by the owners holding the balance of the interest must be submitted to account for 100% of the ownership interest.

   ☐ There are unspecified percentages of ownership. The other parties, including inventors, who together own the entire right, title and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

3. ☐ The assignee of an undivided interest in the entirety (a complete assignment from one of the joint inventors was made). The other parties, including inventors, who together own the entire right, title, and interest are:

   Additional Statement(s) by the owner(s) holding the balance of the interest must be submitted to account for the entire right, title, and interest.

4. ☐ The recipient, via a court proceeding or the like (*e.g.*, bankruptcy, probate), of an undivided interest in the entirety (a complete transfer of ownership interest was made). The certified document(s) showing the transfer is attached.

The interest identified in option 1, 2 or 3 above (not option 4) is evidenced by either (choose **one** of options A or B below):

A. ☐ An assignment from the inventor(s) of the patent application/patent identified above. The assignment was recorded in the United States Patent and Trademark Office at Reel _____, Frame _____, or for which a copy thereof is attached.

B. ☑ A chain of title from the inventor(s), of the patent application/patent identified above, to the current assignee as follows:

1. From: VIVEK RAMANNA To: POUYA TAAGHOL

   The document was recorded in the United States Patent and Trademark Office at
   Reel 035569, Frame 0334, or for which a copy thereof is attached.

2. From: POUYA TAAGHOL To: BIG DATA FEDERATION, INC.

   The document was recorded in the United States Patent and Trademark Office at
   Reel 050963, Frame 0054, or for which a copy thereof is attached.

This collection of information is required by 37 CFR 3.73(b). The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450**.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/AIA/96 (08-12)
Approved for use through 01/31/2013. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

STATEMENT UNDER 37 CFR 3.73(c)

3. From: BIG DATA FEDERATION, INC. To: POUYA TAAGHOL

The document was recorded in the United States Patent and Trademark Office at
Reel 063290, Frame 0800, or for which a copy thereof is attached.

4. From: POUYA TAAGHOL To: EDGE NETWORKING SYSTEMS, LLC

The document was recorded in the United States Patent and Trademark Office at
Reel 063454, Frame 0352, or for which a copy thereof is attached.

5. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at
Reel _____, Frame _____, or for which a copy thereof is attached.

6. From: _____ To: _____

The document was recorded in the United States Patent and Trademark Office at
Reel _____, Frame _____, or for which a copy thereof is attached.

☐ Additional documents in the chain of title are listed on a supplemental sheet(s).

☐ As required by 37 CFR 3.73(c)(1)(i), the documentary evidence of the chain of title from the original owner to the assignee was, or concurrently is being, submitted for recordation pursuant to 37 CFR 3.11.

[NOTE: A separate copy (i.e., a true copy of the original assignment document(s)) must be submitted to Assignment Division in accordance with 37 CFR Part 3, to record the assignment in the records of the USPTO. See MPEP 302.08]

The undersigned (whose title is supplied below) is authorized to act on behalf of the assignee.

/Brandon R. Theiss/	June 23, 2025
Signature	Date
Brandon R. Theiss	70,507
Printed or Typed Name	Title or Registration Number

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requiresthat you begiven certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirementsof the Act, please be advised that: (1)the general authority for the collectionof this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary;and (3) the principalpurpose for whichthe informationis used by the U.S. Patent and Trademark Office is to process and/or examine your submission related toa patent application or patent. If you do not furnishthe requested information, the U.S. Patent and Trademark Officemay not be able to process and/or examine yoursubmission, which may result in termination of proceedings orabandonment of the application or expiration of the patent.

Theinformation provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need forthe information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. Arecord related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (*i.e.*, GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further,a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. Arecord from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
90/019,948	**06/23/2025 02:40:46 PM Z ET**	**UNP-16002**

Title of Invention

Distributed Software Defined Networking

Application Information

APPLICATION TYPE	Utility / ex parte reexam	PATENT #	11695823
CONFIRMATION #	2943	FILED BY	Heather King
PATENT CENTER #	71019259	FILING DATE	05/06/2025
CUSTOMER #	112449	FIRST NAMED INVENTOR	- 11695823
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	Brandon Theiss

Documents

TOTAL DOCUMENTS: 2

DOCUMENT	PAGES	DESCRIPTION	SIZE (KB)
ENS2RE90019948_373_PC.pdf	3	Assignee showing of ownership per 37 CFR 3.73	246 KB
ENS2RE90019948_PowerofAttorney_PC.pdf	3	Power of Attorney	158 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
ENS2RE90019948_373_PC.pdf	91AFA5A4E2053A60E50D19628AEBE57E9AAB24208E53830A1E39029C316D0DD5667A77AD905B495EF256A5276699B434E24A520F21E9B4827A92D20BE147E77A
ENS2RE90019948_PowerofAttorney_PC.pdf	6A60D6D3DA8E1F4CBCE56951515DD5A0F0EB4606EADCD7E912F3C23A8E345828C061B6FED66280C7C3F97DC9CE6BCAC

42F7CEF7C7B3866DDB62F4988FE2F70F8

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Doc Code: PA..
Document Description: Power of Attorney

PTO/AIA/82A (07-13)
Approved for use through 03/31/2021. OMB 0651-0035
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FOR POWER OF ATTORNEY TO ONE OR MORE REGISTERED PRACTITIONERS

NOTE: This form is to be submitted with the Power of Attorney by Applicant form (PTO/AIA/82B) to identify the application to which the Power of Attorney is directed, in accordance with 37 CFR 1.5, unless the application number and filing date are identified in the Power of Attorney by Applicant form. If neither form PTO/AIA/82A nor form PTO/AIA82B identifies the application to which the Power of Attorney is directed, the Power of Attorney will not be recognized in the application.

Application Number	90019948
Filing Date	May 6, 2025
First Named Inventor	Pouya Taaghol
Title	DISTRIBUTED SOFTWARE DEFINED NETWORKING
Art Unit	3992
Examiner Name	BEHZAD PEIKARI
Attorney Docket Number	ENS2-RE90019948

SIGNATURE of Applicant or Patent Practitioner

Signature	/Brandon R. Theiss/	Date (Optional)	
Name	Brandon R. Theiss	Registration Number	70,507
Title (if Applicant is a juristic entity)			
Applicant Name (if Applicant is a juristic entity)			

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4(d) for signature requirements and certifications. If more than one applicant, use multiple forms.

☑ *Total of 1 forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Docusign Envelope ID: 2B069254-9CDC-4AEF-98D6-99B49A7BC8F5

Document Description: Power of Attorney

PTO/AIA/82B (07-13)
Approved for use through 09/30/2025. OMB 0651-0035
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

POWER OF ATTORNEY BY APPLICANT

I hereby revoke all previous powers of attorney given in the application identified in either the attached transmittal letter or the boxes below.

Application Number	Filing Date
90/019,948	5/6/2025

(Note: The boxes above may be left blank if information is provided on form PTO/AIA/82A.)

☑ I hereby appoint the Patent Practitioner(s) associated with the following Customer Number as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above: 3624

OR

☐ I hereby appoint Practitioner(s) named in the attached list (form PTO/AIA/82C) as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the patent application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above. (Note: Complete form PTO/AIA/82C.)

Please recognize or change the correspondence address for the application identified in the attached transmittal letter or the boxes above to:

☑ The address associated with the above-mentioned Customer Number

OR

☐ The address associated with Customer Number:

OR

☐ Firm or Individual Name					
Address					
City		State		Zip	
Country					
Telephone		Email			

I am the Applicant (if the Applicant is a juristic entity, list the Applicant name in the box):

Edge Networking Systems, LLC

☐ Inventor or Joint Inventor (title not required below)

☐ Legal Representative of a Deceased or Legally Incapacitated Inventor (title not required below)

☑ Assignee or Person to Whom the Inventor is Under an Obligation to Assign (provide signer's title if applicant is a juristic entity)

☐ Person Who Otherwise Shows Sufficient Proprietary Interest (e.g., a petition under 37 CFR 1.46(b)(2) was granted in the application or is concurrently being filed with this document) (provide signer's title if applicant is a juristic entity)

SIGNATURE of Applicant for Patent

The undersigned (whose title is supplied below) is authorized to act on behalf of the applicant (e.g., where the applicant is a juristic entity).

Signature	Hyundong Kim	Date (Optional)	6/4/2025 | 6:35 오전 EDT
Name	Grant Hyundong Kim		
Title	Chief Executive Officer		

NOTE: Signature - This form must be signed by the applicant in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications. If more than one applicant, use multiple forms.

☑ Total of 1 forms are submitted.

A Federal agency may not conduct or sponsor, and a person is not required to respond to, nor shall a person be subject to a penalty for failure to comply with an information collection subject to the requirements of the Paperwork Reduction Act of 1995, unless the information collection has a currently valid OMB Control Number. The OMB Control Number for this information collection is 0651-0035. Public burden for this form is estimated to average 30 minutes per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the information collection. Send comments regarding this burden estimate or any other aspect of this information collection, including suggestions for reducing this burden to the Chief Administrative Officer, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450 or email InformationCollection@uspto.gov. **DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS.** If filing this completed form by mail, send to: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The **Privacy Act of 1974 (P.L. 93-579)** requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Litigation Search Report CRU 3999
Reexam No. 90/019,948

To: Behzad Peikari	**From: Patricia Martin, Paralegal**
Location: CRU	**Location: OCRU**
Art Unit: 3992	**Phone: (571) 272-5004**
Date: May 13, 2025	

Search Notes

U.S. Patent No. 11,695,823

Litigation: Found

1. Performed a KeyCite Search in Westlaw, which retrieves all history on the patent including any litigation.
2. Performed a search on the patent in Lexis CourtLink for any open dockets or closed cases.

.

Paper No. Date: 5/13/2025

Citing References (11)

Treatment	Title	Date	Type	Depth	Headnote(s)
Examined by	**1.** Petition for Inter Partes Review of U.S. Patent No. 11,695,823 Under 35 U.S.C. ss 311-319 and 37 C.F.R. ss 42.1-.80, 42.100-.107 MICROSOFT CORPORATION, Petitioner, v. EDGE NETWORKING SYSTEMS, LLC, Patent Owner. 2025 WL 523797, *1+ , Patent Tr. & App. Bd. (Administrative Filing)	Feb. 17, 2025	Administrative Filing		—
Discussed by	**2.** Microsoft Corporation's Motion to Dismiss Plaintiff's Corrected Complaint (Dkt. No. 4) Pursuant to Rule 12(b)(6) EDGE NETWORKING SYSTEMS, LLC, Plaintiff, v. MICROSOFT CORPORATION, Defendant. 2024 WL 3903928, *1+ , W.D.Tex. (Trial Motion, Memorandum and Affidavit)	May 09, 2024	Motion		—
Cited by	**3.** Microsoft Corporation's Reply in Support of Motion to Dismiss Plaintiff's Corrected Complaint Pursuant to Rule 12(b)(6) EDGE NETWORKING SYSTEMS, LLC, Plaintiff, v. MICROSOFT CORPORATION, Defendant. 2024 WL 3903931, *1+ , W.D.Tex. (Trial Motion, Memorandum and Affidavit)	May 30, 2024	Motion		—
Cited by	**4.** Plaintiff Edge Networking Systems, LLC's Opposition to Defendants' Rule 12(b)(6) Motion to Dismiss EDGE NETWORKING SYSTEMS, LLC, Plaintiff, v. MICROSOFT CORPORATION., Defendant. 2024 WL 3903929, *1 , W.D.Tex. (Trial Motion, Memorandum and Affidavit)	May 23, 2024	Motion		—
Mentioned by	**5.** Petition for Inter Partes Review of U.S. Patent No. 10,686,871 Under 35 U.S.C. ss 311-319 and 37 C.F.R. ss 42.1-.80, 42.100-.107 MICROSOFT CORPORATION, Petitioner, v. EDGE NETWORKING SYSTEMS, LLC, Patent Owner. 2025 WL 523798, *1 , Patent Tr. & App. Bd. (Administrative Filing)	Feb. 17, 2025	Administrative Filing		—
—	**6.** Title Available on Document Page	—	Docket Summaries	—	—
—	**7.** Edge Networking Systems, LLC v. Microsoft Corporation	Feb. 29, 2024	Docket Summaries	—	—
—	**8.** DISTRIBUTED SOFTWARE DEFINED NETWORKING Out Of Print US PAT 12126673 , U.S. PTO Utility The Distributed Software Defined Network (dSDN) disclosed herein is an end to-end architecture that enables secure and flexible programmability across a network with full lifecycle...	Oct. 22, 2024	Patents	—	—

WESTLAW © 2025 Thomson Reuters. No claim to original U.S. Government Works.

Treatment	Title	Date	Type	Depth	Headnote(s)
—	**9.** METHOD FOR MANAGING UPDATES TO A DISTRIBUTED NETWORK THROUGH AN APPLICATION MANAGEMENT PORTAL Out Of Plan US PAT 12126674 , U.S. PTO Utility The Distributed Software Defined Network (dSDN) disclosed herein is an end-to-end architecture that enables secure and flexible programmability across a network with full lifecycle...	Oct. 22, 2024	Patents	—	—
—	**10.** METHOD FOR MANAGING UPDATES TO A DISTRIBUTED NETWORK WITH NO INTERRUPTION TO OPERATIONS Out Of Plan US PAT 12113850 , U.S. PTO Utility The Distributed Software Defined Network (dSDN) disclosed herein is an end-to-end architecture that enables secure and flexible programmability across a network with full lifecycle...	Oct. 08, 2024	Patents	—	—
—	**11.** METHOD FOR MANAGING UPDATES TO A DISTRIBUTED NETWORK INDEPENDENT OF HARDWARE Out Of Plan US PAT 12113851 , U.S. PTO Utility The Distributed Software Defined Network (dSDN) disclosed herein is an end-to-end architecture that enables secure and flexible programmability across a network with full lifecycle...	Oct. 08, 2024	Patents	—	—

WESTLAW © 2025 Thomson Reuters. No claim to original U.S. Government Works.

User Name: Patricia Martin
Date and Time: Tuesday, May 13, 2025 7:13 AM EDT
Job Number: 252698259

Results List (includes up to 250)

1. Results list for:11695823 or 11,695,823
   Client/Matter: -None-
   Terms: 11695823 or 11,695,823
   Search Type: Terms and Connectors
   Narrowed by:

Content Type	Narrowed by
US Dockets	Case Status: Open,Unknown,Closed; Case Type: Civil

LexisNexis | About LexisNexis | Privacy Policy | Terms & Conditions | Copyright © 2025 LexisNexis

LexisNexis

Results for: **11695823 or 11,695,823**

Dockets

1. Microsoft Corporation Vs. Edge Networking Systems, LLC

... Yes 1002 Prosecution history of the ???823 **Patent** (??????823 File History???) 68 2025-02-17 1001 U.S. **Patent** No. **11,695,823** (the ??????823 **Patent** ???) 66877803 Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1001 ...
... 1005 File history of the parent **patent** to the ???823 **Patent** '624 65 2025-02-17 1004 U.S. **Patent** No. 6,496,575 to Vasell (???Vasell **Patent** ???) 66877805 Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1004 ...
... 1007 File history of the other **patent** from which the ???823 **Patent** '871 63 2025-02-17 1006 File history of the **patent** from which the ???823 **Patent** '905 66877807 Filed By: PETITIONER Document Type: ...
... 1006 File history of the **patent** from which the ???823 **Patent** '905 64 2025-02-17 1005 File history of the parent **patent** to the ???823 **Patent** '624 66877806 Filed By: PETITIONER Document Type: ...
... Depth???, Manning Publication 62 2025-02-17 1007 File history of the other **patent** from which the ???823 **Patent** '871 66877808 Filed By: PETITIONER Document Type: ...

Court: United States Patent Trial and Appeal Board | **Date Filed**: Feb 17, 2025 | **Docket Number**: IPR2025-00618 | **Nature of Suit**: Unknown | **Status**: Open

2. Edge Networking Systems Llc V. Amazon.Com, Inc., Et Al

... Sheet)(Rubino, Vincent) (Entered: 11/01/2024) 8034735360 This is a complaint for **patent** infringement filed by Edge Networking Systems LLC against Amazon.com, Inc., Amazon.com ...
... Kubernetes Service (EKS) Web Services infringe three of Edge Networking Systems' **patents** related to a distributed software defined networking architecture.The complaint asserts that Amazon directly and indirectly infringes the **patents** by making, using, selling, offering to sell, and/or importing the accused ...
... royalty, as well as attorneys' fees and costs. 2025-05-11 COMPLAINT FOR **PATENT** INFRINGEMENT against All Defendants (Filing fee $ 405 receipt number ...
... Edge Networking Systems LLC. (Attachments: # 1 Exhibit A - U.S. **Patent** No. **11,695,823**, # 2 Exhibit B - U.S. **Patent** No. 10,893,095, # 3 Exhibit C - U.S. **Patent** No. 10,686,871, # 4 Civil Cover Sheet)(Rubino, Vincent) (Entered: 11/01/2024)...
... Exhibit A - U.S. **Patent** No. **11,695,823** Exhibit A - U.S. **Patent** No. **11,695,823** , # 2 Exhibit B - U.S. **Patent** No. 10,893,095 Exhibit B - U.S. **Patent** No. 10,893,095...
... AMENDED COMPLAINT FOR **PATENT** INFRINGEMENT against Amazon.com Services LLC, filed by Edge Networking Systems LLC. (Attachments: # 1 Exhibit A - U.S. **Patent** No. **11,695,823** , # 2 Exhibit B - U.S. **Patent** No. 10,893,095 , # 3 ...

Court: United States District Court, Texas Eastern | **Date Filed**: Nov 01, 2024 | **Docket Number**: 2:24cv887 | **Nature of Suit**: Patent | **Cause**: Patent Infringement | **Status**: Open

3. Edge Networking Systems, Llc V. Microsoft Corporation

... Exhibit 2 - U.S. **Patent** No. 10,893,095 Exhibit 2 - U.S. **Patent** No. 10,893,095 , # 3 Exhibit 3 - U.S. **Patent** No. **11,695,823** Exhibit 3 - U.S. **Patent** No. **11,695,823** ...
... COMPLAINT for **Patent** Infringement (Filing fee $ 405 receipt number ATXWDC-18479169), filed by ...
... Edge Networking Systems, LLC. (Attachments: # 1 Exhibit 1 - U.S. **Patent** No. 10,686,871, # 2 Exhibit 2 - U.S. **Patent** No. 10,893,095, # 3 Exhibit 3 - U.S. **Patent** No. **11,695,823**, # 4 Civil Cover Sheet)(Hecht,

David) (Entered: 02/29/2024) 2 2024-02-29 Notice of Filing of **Patent**/Trademark Form (AO 120). AO 120 forwarded to the Director of the U.S. **Patent** and Trademark Office. (Hecht, David) (Entered: 02/29/2024) 8022310117 2025-04-30 Notice of Filing of **Patent**/Trademark Form (AO 120). AO 120 forwarded to the Director of the U.S. **Patent** and Trademark Office. (Hecht, David) (Entered: 02/29/2024)...
Patent Infringement 35:100 0 2025-03-21T00:00:00 0 None 1 2024-02-29 COMPLAINT for **Patent** Infringement (Filing fee $ 405 receipt number ATXWDC-18479169), filed by ...

Court: United States District Court, Texas Western | **Date Filed**: Feb 29, 2024 | **Docket Number**: 1:24cv215 | **Nature of Suit**: Patent | **Cause**: Patent Infringement | **Status**: Open

4. Microsoft Corporation Vs. Edge Networking Systems, LLC

... Depth???, Manning Publication 62 2025-02-17 1007 File history of related U.S. **Patent** No. **11,695,823** (the ??????823 **Patent**???) 66877882 Filed By: PETITIONER Document Type: ...
... Yes 1002 Prosecution history of the ???871 **Patent** (??????871 File History???) 68 2025-02-17 1001 U.S. **Patent** No. 10,686,871 (the ??????871 **Patent** ???) 66877877 Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1001 ...
... 1006 File history of related U.S. **Patent** No. 10,893,095 (the ??????095 **Patent**???) 64 2025-02-17 1005 File history of related U.S. **Patent** No. 9,483,624 (the ??????624 **Patent**???) 66877880 Filed By: PETITIONER Document Type: ...
... 1007 File history of related U.S. **Patent** No. **11,695,823** (the ??????823 **Patent**???) 63 2025-02-17 1006 File history of related U.S. **Patent** No. 10,893,095 (the ??????095 **Patent**???) 66877881 Filed By: PETITIONER Document Type: ...
... accorded Case Type IPR: Inter partes review Technical Center Number 2400 **Patent** Application Number 15836824 **Patent** Number 10686871 Microsoft Corporation P Petitioner MICROSOFT CORPORATION 2025-04-28T01:04:11.197 Edge Networking ...
... Erez Zadok, (Zadok) 67 2025-02-17 1002 Prosecution history of the ???871 **Patent** (??????871 File History???) 66877878 Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit ...

Court: United States Patent Trial and Appeal Board | **Date Filed**: Feb 17, 2025 | **Docket Number**: IPR2025-00619 | **Nature of Suit**: Unknown | **Status**: Open

User Name: Patricia Martin
Date and Time: Tuesday, May 13, 2025 7:18 AM EDT
Job Number: 252698451

Document (1)

1. IPR2025-00618, Microsoft Corporation Vs. Edge Networking Systems, LLC
   Client/Matter: -None-
   Search Terms: 11695823 or 11,695,823
   Search Type: Terms and Connectors
   Narrowed by:

Content Type	Narrowed by
Dockets	Case Status: Open,Unknown,Closed; Case Type: Civil

LexisNexis | About LexisNexis | Privacy Policy | Terms & Conditions | Copyright © 2025 LexisNexis

IPR2025-00618, Microsoft Corporation Vs. Edge Networking Systems, LLC

US Patent Trial and Appeals Board - Alexandria

Alexandria

This case was retrieved on 05/13/2025

Header

Case Number: IPR2025-00618
Date Filed: 02/17/2025
Date Full Case Retrieved: 05/13/2025
Status: Open
Misc: (0) Unknown; Civil

Summary

Court Case Status: Notice of filing date accorded
Case Type: IPR: Inter partes review
Technical Center Number: 2400
Patent Application Number: 17142983
Patent Number: 11695823

Participants

Litigants	Attorneys
Microsoft Corporation **Petitioner**	
Edge Networking Systems, LLC **PatentOwner**	

Proceedings

Date	#	Proceeding Text	Details
02/17/2025	1	1072 Scheduling Order, filed as Document No. 27, on September 4, 2024	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1072 Public? Yes
02/17/2025	2	1071 Defendant Microsoft Corporation???s Responsive Claim Construction Brief	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1071 Public? Yes
02/17/2025	3	1070 Plaintiff Edge Networking Systems, LLC???s Opening Claim Construction	Filed By: PETITIONER Document Type: EXHIBIT

Patricia Martin

Date	#	Proceeding Text	Details
			Paper/Exhibit No: 1070 Public? Yes
02/17/2025	4	1069 ???Avalanche Effect???, Wikipedia, archived on April 30, 2010 at Internet	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1069 Public? Yes
02/17/2025	5	1068 J. Preshing, ???Hash Collision Probabilities???, Preshing on Programming	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1068 Public? Yes
02/17/2025	6	1067 U.S. Department of Commerce, FIPS PUB 186-2, Digital Signature	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1067 Public? Yes
02/17/2025	7	1066 ???Apache Archive Distribution Directory???, Apache Software Foundation	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1066 Public? Yes
02/17/2025	8	1065 ???apache_1.3.19.tar.gz.asc??? Signature File, Apache Software Foundation	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1065 Public? Yes
02/17/2025	9	1064 ???apache_1.3.19.tar.gz.md5??? Hash File, Apache Software Foundation	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1064 Public? Yes
02/17/2025	10	1063 Excerpts from Stallings, ???Operating Systems Internals and Design	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1063 Public? Yes

Date	#	Proceeding Text	Details
02/17/2025	11	1062 Excerpts from Benevenuti, ???Understanding Linux Network Internals???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1062 Public? Yes
02/17/2025	12	1061 Excerpts from McKusick and Neville-Neil, ???The Design and ...	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1061 Public? Yes
02/17/2025	13	1060 Excerpts from Silberschatz et al., Operating System Concepts	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1060 Public? Yes
02/17/2025	14	1059 Excerpts from Silberschatz et al., Operating System Concepts	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1059 Public? Yes
02/17/2025	15	1058 Usenix Jails in FreeBSD for Fun and Profit, Paco Hope, USENIX ;login	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1058 Public? Yes
02/17/2025	16	1057 Microsoft Computer Dictionary, 3rd ed., 1997, excerpts	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1057 Public? Yes
02/17/2025	17	1056 ???Using Java??? RMI with SSL???, Oracle Java SE Documentation	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1056 Public? Yes
02/17/2025	18	1055 ???Using Custom Socket Factories with Java RMI???, Oracle Java SE Document	Filed By: PETITIONER Document Type: EXHIBIT

Date	#	Proceeding Text	Details
			Paper/Exhibit No: 1055 Public? Yes
02/17/2025	19	1054 ???Java??? Remote Method Invocation API (Java RMI)???, Oracle Java SE	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1054 Public? Yes
02/17/2025	20	1053 ???Java Secure Socket Extension (JSSE) Reference Guide for Java Platform	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1053 Public? Yes
02/17/2025	21	1052 Excerpts from B. Sosinsky, ???Cloud Computing Bible???, Wiley Publishing	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1052 Public? Yes
02/17/2025	22	1051 R. Hall, ???Oscar Bundle Repository???, archived on June 30, 2004	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1051 Public? Yes
02/17/2025	23	1050 The Open Services Gateway Initiative, ???Specification Overview???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1050 Public? Yes
02/17/2025	24	1048 ???REST Security Cheat Sheet???, The Open Web Application Security Project	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1048 Public? Yes
02/17/2025	25	1047 ???Web Service Security Cheat Sheet???, The Open Web Application Security	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1047 Public? Yes

Date	#	Proceeding Text	Details
02/17/2025	26	1046 Oasis Standard Specification, ???Web Services Security: SOAP Message	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1046 Public? Yes
02/17/2025	27	1045 Excerpts from S. Garfinkel, et al., ???Web Security & Commerce???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1045 Public? Yes
02/17/2025	28	1044 Excerpts from A.S. Tanenbaum, ???Modern Operating Systems???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1044 Public? Yes
02/17/2025	29	1043 R. Kawashima, ???vNFC: A Virtual Networking Function Container for SDN-e	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1043 Public? Yes
02/17/2025	30	1042 ???Knoplerfish Pro Premium??? and ???Knoplerfish Pro Enterprise??? Data Sheets	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1042 Public? Yes
02/17/2025	31	1041 P. Kriens, ???The Bundle Repository???, OSGi Alliance Blog, April 7, 2006	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1041 Public? Yes
02/17/2025	32	1039 Li, et al., ???ExpoNet: A Flexible Platform for Concurrent Experiments	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1039 Public? Yes
02/17/2025	33	1036 S. Soltesz, et al., ???Container-based Operating System Virtualization	Filed By: PETITIONER Document Type: EXHIBIT

Date	#	Proceeding Text	Details
			Paper/Exhibit No: 1036 Public? Yes
02/17/2025	34	1035 Curriculum Vitae (CV) of Petitioner???s Expert Dr. Erez Zadok.	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1035 Public? Yes
02/17/2025	35	1034 Excerpts from E. Cole, et al., ???Network Security Bible???, Wiley Publish	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1034 Public? Yes
02/17/2025	36	1033 Excerpts from D. Downing, et al., ???Dictionary of Computer and Internet	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1033 Public? Yes
02/17/2025	37	1032 Excerpts from A.S. Tanenbaum, ???Computer Networks???, Prentice Hall	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1032 Public? Yes
02/17/2025	38	1031 Excerpts from A.S. Tanenbaum, ???Modern Operating Systems???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1031 Public? Yes
02/17/2025	39	1030 U.S. Patent Publication No. 2013/0086147 to Kashyap	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1030 Public? Yes
02/17/2025	40	1029 T. Forst, ???Cisco Application eXtension Platform ??? OSGi Add-on	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1029 Public? Yes

Date	#	Proceeding Text	Details
02/17/2025	41	1028 The OSGi Alliance, ???OSGi Service Platform Enterprise Specification???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1028 Public? Yes
02/17/2025	42	1027 ???About Makewave: History in Making???, Makewave AB website	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1027 Public? Yes
02/17/2025	43	1026 ???About Us: Background???, Gatespace AB website	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1026 Public? Yes
02/17/2025	44	1025 ???Management Team???, Gatespace AB website, archived on February 8, 2002	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1025 Public? Yes
02/17/2025	45	1024 ???Open Services Gateway Initiative Elects New Leadership???, Home Toys In	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1024 Public? Yes
02/17/2025	46	1023 ???Open Services Gateway Initiative: Officers???, OSGi website	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1023 Public? Yes
02/17/2025	47	1022 ???Open Services Gateway Initiative: Charter???, OSGi website	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1022 Public? Yes
02/17/2025	48	1021 ???People???, Carlstedt Research & Technology AB website	Filed By: PETITIONER Document Type: EXHIBIT

Date	#	Proceeding Text	Details
			Paper/Exhibit No: 1021 Public? Yes
02/17/2025	49	1020 ???Fifteen industry leaders to create standard for bringing Internet	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1020 Public? Yes
02/17/2025	50	1019 ???Ericsson e-services is a part of the Open Services Gateway Initiative	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1019 Public? Yes
02/17/2025	51	1018 ???Carlstedt Research & Technology Forms New Company for Developing	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1018 Public? Yes
02/17/2025	52	1017 ???About Gatespace???, Gatespace AB website, archived on January 6, 2000	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1017 Public? Yes
02/17/2025	53	1016 U.S. Provisional Patent Application No. 60/123,971	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1016 Public? Yes
02/17/2025	54	1015 U.S. Provisional Patent Application No. 60/088,437	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1015 Public? Yes
02/17/2025	55	1014 Declaration of June A. Munford	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1014 Public? Yes

Date	#	Proceeding Text	Details
02/17/2025	56	1013 OSGi Alliance, ???About the OSGi Service Platform???, Technical Whitepaper	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1013 Public? Yes
02/17/2025	57	1012 S. K??chele, et al., ???Component-based scalability for cloud application	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1012 Public? Yes
02/17/2025	58	1011 J. S. Rellermeyer and S. Bagchi, ???Dependability as a cloud service	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1011 Public? Yes
02/17/2025	59	1010 J. E. Kim, et al., ???Seamless Integration of Heterogeneous Devices	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1010 Public? Yes
02/17/2025	60	1009 Excerpts from R. Hall, et al., ???OSGi in Action - Creating Modular	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1009 Public? Yes
02/17/2025	61	1008 Excerpts from A. de Castro Alves, ???OSGi in Depth???, Manning Publication	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1008 Public? Yes
02/17/2025	62	1007 File history of the other patent from which the ???823 Patent '871	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1007 Public? Yes
02/17/2025	63	1006 File history of the patent from which the ???823 Patent '905	Filed By: PETITIONER Document Type: EXHIBIT

Date	#	Proceeding Text	Details
			Paper/Exhibit No: 1006 Public? Yes
02/17/2025	64	1005 File history of the parent patent to the ???823 Patent '624	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1005 Public? Yes
02/17/2025	65	1004 U.S. Patent No. 6,496,575 to Vasell (???Vasell Patent???)	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1004 Public? Yes
02/17/2025	66	1003 Declaration of Petitioner???s Expert, Dr. Erez Zadok, (Zadok)	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1003 Public? Yes
02/17/2025	67	1002 Prosecution history of the ???823 Patent (??????823 File History???)	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1002 Public? Yes
02/17/2025	68	1001 U.S. Patent No. 11,695,823 (the ??????823 Patent???)	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1001 Public? Yes
04/15/2025	69	Notice: NOTICE OF FILING DATE ACCORDED TO PETITION AND TIME FOR FILING PATENT OWNER PRELIMINARY RESPONSE	Filed By: BOARD Document Type: PAPER Paper/Exhibit No: 6 Public? Yes
04/15/2025	70	EXPUNGED	Filed By: BOARD Document Type: PAPER Paper/Exhibit No: 5 Public? No

Date	#	Proceeding Text	Details
03/31/2025	71	Notice: Power of Attorney	Filed By: PATENT OWNER Document Type: PAPER Paper/Exhibit No: 4 Public? Yes
03/31/2025	72	Notice: Mandatory Notice	Filed By: PATENT OWNER Document Type: PAPER Paper/Exhibit No: 3 Public? Yes
02/17/2025	73	Petition: as filed	Filed By: PETITIONER Document Type: PAPER Paper/Exhibit No: 2 Public? Yes
02/17/2025	74	Notice: Power of Attorney	Filed By: PETITIONER Document Type: PAPER Paper/Exhibit No: 1 Public? Yes

Copyright © LexisNexis CourtLink, Inc. All Rights Reserved.
*** THIS DATA IS FOR INFORMATIONAL PURPOSES ONLY ***

End of Document

User Name: Patricia Martin
Date and Time: Tuesday, May 13, 2025 7:19 AM EDT
Job Number: 252698478

Document (1)

1. 2:24cv887, Edge Networking Systems Llc V. Amazon Com, Inc., Et Al
   Client/Matter: -None-
   Search Terms: 11695823 or 11,695,823
   Search Type: Terms and Connectors
   Narrowed by:

Content Type	Narrowed by
Dockets	Case Status: Open,Unknown,Closed; Case Type: Civil

2:24cv887, Edge Networking Systems Llc V. Amazon.Com, Inc., Et Al

US District Court Docket

United States District Court, Texas Eastern

(Marshall)

This case was retrieved on 05/12/2025

Header

Case Number: 2:24cv887
Date Filed: 11/01/2024
Assigned To: District Judge Rodney Gilstrap
Referred To: Magistrate Judge Roy S. Payne
Nature of Suit: ***Patent*** (830)
Cause: ***Patent*** Infringement
Lead Docket: None
Other Docket: None
Jurisdiction: Federal Question

Class Code: Open
Statute: 28:1338
Jury Demand: Both
Demand Amount: $0
NOS Description: ***Patent***

Participants

Litigants

Edge Networking Systems LLC
Plaintiff

Attorneys

Alfred Ross Fabricant
ATTORNEY TO BE NOTICED
Fabricant LLP - NY
230 Park Avenue 3rd Floor W.
New York, NY 10169
USA
212-257-5797 Fax: 212-257-5796
Email:Ffabricant@fabricantllp.Com

Jacob Daniel Ostling
ATTORNEY TO BE NOTICED
Fabricant LLP - Rye
411 Theodore Fremd Road Suite 206 South
Rye, NY 10580
USA
212-257-5797 Fax: 212-257-5796
Email:Jostling@fabricantllp.Com

Joseph Michael Mercadante
ATTORNEY TO BE NOTICED
Fabricant LLP - NY
230 Park Avenue 3rd Floor W.
New York, NY 10169
USA
212-257-5797 Fax: 212-257-5796
Email:Jmercadante@fabricantllp.Com

Julian Glenn Pymento
ATTORNEY TO BE NOTICED

Patricia Martin

Litigants	Attorneys
	Fabricant LLP 51 Jfk Parkway Short Hills, NJ 07078 USA 646-797-4341 Email:Jpymento@fabricantllp.Com Justin Kurt Truelove ATTORNEY TO BE NOTICED Truelove Law Firm 207 N Wellington P O Box 1409 Marshall, TX 75671 USA 903-938-8321 Fax: 903-215-8510 Email:Kurt@truelovelawfirm.Com Peter Lambrianakos ATTORNEY TO BE NOTICED Fabricant LLP 411 Theodore Fremd Avenue Suite 206 South Rye, NY 10580 USA 212-257-5797 Fax: 212-257-5796 Email:Plambrianakos@fabricantllp.Com Vincent J Rubino , Iii ATTORNEY TO BE NOTICED Fabricant LLP - NY 230 Park Avenue 3rd Floor W. New York, NY 10169 USA 212-257-5797 Fax: 212.257.5796 Email:Vrubino@fabricantllp.Com
Amazon.Com, Inc., **Defendant**	Srinivas Giriraj Pathmanaban LEAD ATTORNEY;ATTORNEY TO BE NOTICED Cleary Gottlieb Steen & Hamilton LLP 1841 Page Mill Road Suite 250 Palo Alto, CA 94304 USA 650-815-4100 Email:Gpathmanaban@cgsh.Com Clement Naples ATTORNEY TO BE NOTICED Cleary Gottlieb Steen & Hamilton LLP One Liberty Plaza New York, NY 10006 USA 212-225-2516 Email:Cnaples@cgsh.Com Klaus H Hamm ATTORNEY TO BE NOTICED Klarquist Sparkman LLP One World Trade Center 121 Sw Salmon Street, Suite 1600 Portland, OR 97204 USA 503/595-5300 Fax: 503/595-5301 Email:Klaus.Hamm@klarquist.Com Michael E Bern PRO HAC VICE

Litigants	Attorneys
	[Terminated: 04/21/2025] Latham & Watkins LLP - D.C. 555 Eleventh Street, Nw Suite 1000 Washington, DC 20004 USA 202-637-2200 Fax: 202-637-2201 Email:Michael.Bern@lw.Com Michael E Jones ATTORNEY TO BE NOTICED Potter Minton 102 North College Suite 900 Tyler, TX 75702 USA 903-525-2239 Email:Mikejones@potterminton.Com Robert Cruzen ATTORNEY TO BE NOTICED Klarquist Sparkman LLP One World Trade Center 121 Sw Salmon Street, Suite 1600 Portland, OR 97204 USA 503/595-5300 Fax: 505/595-5301 Email:Rob.Cruzen@klarquist.Com Shaun William Hassett ATTORNEY TO BE NOTICED Potter Minton 102 North College Suite 900 Tyler, TX 75702 USA 903-525-2272 Email:Shaunhassett@potterminton.Com
AMAZON.COM SERVICES LLC **Defendant**	Srinivas Giriraj Pathmanaban LEAD ATTORNEY;ATTORNEY TO BE NOTICED Cleary Gottlieb Steen & Hamilton LLP 1841 Page Mill Road Suite 250 Palo Alto, CA 94304 USA 650-815-4100 Email:Gpathmanaban@cgsh.Com Clement Naples ATTORNEY TO BE NOTICED Cleary Gottlieb Steen & Hamilton LLP One Liberty Plaza New York, NY 10006 USA 212-225-2516 Email:Cnaples@cgsh.Com Klaus H Hamm ATTORNEY TO BE NOTICED Klarquist Sparkman LLP One World Trade Center 121 Sw Salmon Street, Suite 1600 Portland, OR 97204 USA 503/595-5300 Fax: 503/595-5301 Email:Klaus.Hamm@klarquist.Com Michael E Bern PRO HAC VICE

Litigants	Attorneys
	[Terminated: 04/16/2025] Latham & Watkins LLP - D.C. 555 Eleventh Street, Nw Suite 1000 Washington, DC 20004 USA 202-637-2200 Fax: 202-637-2201 Email:Michael.Bern@lw.Com Michael E Jones ATTORNEY TO BE NOTICED Potter Minton 102 North College Suite 900 Tyler, TX 75702 USA 903-525-2239 Email:Mikejones@potterminton.Com Robert Cruzen ATTORNEY TO BE NOTICED Klarquist Sparkman LLP One World Trade Center 121 Sw Salmon Street, Suite 1600 Portland, OR 97204 USA 503/595-5300 Fax: 505/595-5301 Email:Rob.Cruzen@klarquist.Com Shaun William Hassett ATTORNEY TO BE NOTICED Potter Minton 102 North College Suite 900 Tyler, TX 75702 USA 903-525-2272 Email:Shaunhassett@potterminton.Com
AMAZON WEB SERVICES, INC. **Defendant**	Srinivas Giriraj Pathmanaban LEAD ATTORNEY;ATTORNEY TO BE NOTICED Cleary Gottlieb Steen & Hamilton LLP 1841 Page Mill Road Suite 250 Palo Alto, CA 94304 USA 650-815-4100 Email:Gpathmanaban@cgsh.Com Clement Naples ATTORNEY TO BE NOTICED Cleary Gottlieb Steen & Hamilton LLP One Liberty Plaza New York, NY 10006 USA 212-225-2516 Email:Cnaples@cgsh.Com Klaus H Hamm ATTORNEY TO BE NOTICED Klarquist Sparkman LLP One World Trade Center 121 Sw Salmon Street, Suite 1600 Portland, OR 97204 USA 503/595-5300 Fax: 503/595-5301 Email:Klaus.Hamm@klarquist.Com Michael E Bern PRO HAC VICE;ATTORNEY TO BE NOTICED

Litigants	Attorneys
	Latham & Watkins LLP - D.C. 555 Eleventh Street, Nw Suite 1000 Washington, DC 20004 USA 202-637-2200 Fax: 202-637-2201 Email:Michael.Bern@lw.Com Michael E Jones ATTORNEY TO BE NOTICED Potter Minton 102 North College Suite 900 Tyler, TX 75702 USA 903-525-2239 Email:Mikejones@potterminton.Com Robert Cruzen ATTORNEY TO BE NOTICED Klarquist Sparkman LLP One World Trade Center 121 Sw Salmon Street, Suite 1600 Portland, OR 97204 USA 503/595-5300 Fax: 505/595-5301 Email:Rob.Cruzen@klarquist.Com Shaun William Hassett ATTORNEY TO BE NOTICED Potter Minton 102 North College Suite 900 Tyler, TX 75702 USA 903-525-2272 Email:Shaunhassett@potterminton.Com
AMAZON.COM SERVICES LLC **Counter Claimant**	Srinivas Giriraj Pathmanaban LEAD ATTORNEY;ATTORNEY TO BE NOTICED Cleary Gottlieb Steen & Hamilton LLP 1841 Page Mill Road Suite 250 Palo Alto, CA 94304 USA 650-815-4100 Email:Gpathmanaban@cgsh.Com Clement Naples ATTORNEY TO BE NOTICED Cleary Gottlieb Steen & Hamilton LLP One Liberty Plaza New York, NY 10006 USA 212-225-2516 Email:Cnaples@cgsh.Com Klaus H Hamm ATTORNEY TO BE NOTICED Klarquist Sparkman LLP One World Trade Center 121 Sw Salmon Street, Suite 1600 Portland, OR 97204 USA 503/595-5300 Fax: 503/595-5301 Email:Klaus.Hamm@klarquist.Com Michael E Bern

Litigants	Attorneys
	[Terminated: 04/16/2025] Latham & Watkins LLP - D.C. 555 Eleventh Street, Nw Suite 1000 Washington, DC 20004 USA 202-637-2200 Fax: 202-637-2201 Email:Michael.Bern@lw.Com Michael E Jones ATTORNEY TO BE NOTICED Potter Minton 102 North College Suite 900 Tyler, TX 75702 USA 903-525-2239 Email:Mikejones@potterminton.Com Robert Cruzen ATTORNEY TO BE NOTICED Klarquist Sparkman LLP One World Trade Center 121 Sw Salmon Street, Suite 1600 Portland, OR 97204 USA 503/595-5300 Fax: 505/595-5301 Email:Rob.Cruzen@klarquist.Com Shaun William Hassett ATTORNEY TO BE NOTICED Potter Minton 102 North College Suite 900 Tyler, TX 75702 USA 903-525-2272 Email:Shaunhassett@potterminton.Com
Edge Networking Systems LLC **Counter Defendant**	Alfred Ross Fabricant ATTORNEY TO BE NOTICED Fabricant LLP - NY 230 Park Avenue 3rd Floor W. New York, NY 10169 USA 212-257-5797 Fax: 212-257-5796 Email:Ffabricant@fabricantllp.Com Jacob Daniel Ostling ATTORNEY TO BE NOTICED Fabricant LLP - Rye 411 Theodore Fremd Road Suite 206 South Rye, NY 10580 USA 212-257-5797 Fax: 212-257-5796 Email:Jostling@fabricantllp.Com Joseph Michael Mercadante ATTORNEY TO BE NOTICED Fabricant LLP - NY 230 Park Avenue 3rd Floor W. New York, NY 10169 USA 212-257-5797 Fax: 212-257-5796 Email:Jmercadante@fabricantllp.Com

Litigants	Attorneys
	Julian Glenn Pymento ATTORNEY TO BE NOTICED Fabricant LLP 51 Jfk Parkway Short Hills, NJ 07078 USA 646-797-4341 Email:Jpymento@fabricantllp.Com Justin Kurt Truelove ATTORNEY TO BE NOTICED Truelove Law Firm 207 N Wellington P O Box 1409 Marshall, TX 75671 USA 903-938-8321 Fax: 903-215-8510 Email:Kurt@truelovelawfirm.Com Peter Lambrianakos ATTORNEY TO BE NOTICED Fabricant LLP 411 Theodore Fremd Avenue Suite 206 South Rye, NY 10580 USA 212-257-5797 Fax: 212-257-5796 Email:Plambrianakos@fabricantllp.Com Vincent J Rubino , III ATTORNEY TO BE NOTICED Fabricant LLP - NY 230 Park Avenue 3rd Floor W. New York, NY 10169 USA 212-257-5797 Fax: 212.257.5796 Email:Vrubino@fabricantllp.Com
AMAZON.COM SERVICES LLC **Counter Claimant**	Srinivas Giriraj Pathmanaban LEAD ATTORNEY;ATTORNEY TO BE NOTICED Cleary Gottlieb Steen & Hamilton LLP 1841 Page Mill Road Suite 250 Palo Alto, CA 94304 USA 650-815-4100 Email:Gpathmanaban@cgsh.Com Clement Naples ATTORNEY TO BE NOTICED Cleary Gottlieb Steen & Hamilton LLP One Liberty Plaza New York, NY 10006 USA 212-225-2516 Email:Cnaples@cgsh.Com Klaus H Hamm ATTORNEY TO BE NOTICED Klarquist Sparkman LLP One World Trade Center 121 Sw Salmon Street, Suite 1600 Portland, OR 97204 USA 503/595-5300 Fax: 503/595-5301 Email:Klaus.Hamm@klarquist.Com

Litigants	Attorneys
	Michael E Bern [Terminated: 04/16/2025] Latham & Watkins LLP - D.C. 555 Eleventh Street, Nw Suite 1000 Washington, DC 20004 USA 202-637-2200 Fax: 202-637-2201 Email:Michael.Bern@lw.Com Michael E Jones ATTORNEY TO BE NOTICED Potter Minton 102 North College Suite 900 Tyler, TX 75702 USA 903-525-2239 Email:Mikejones@potterminton.Com Robert Cruzen ATTORNEY TO BE NOTICED Klarquist Sparkman LLP One World Trade Center 121 Sw Salmon Street, Suite 1600 Portland, OR 97204 USA 503/595-5300 Fax: 505/595-5301 Email:Rob.Cruzen@klarquist.Com Shaun William Hassett ATTORNEY TO BE NOTICED Potter Minton 102 North College Suite 900 Tyler, TX 75702 USA 903-525-2272 Email:Shaunhassett@potterminton.Com
Edge Networking Systems LLC **Counter Defendant**	Alfred Ross Fabricant ATTORNEY TO BE NOTICED Fabricant LLP - NY 230 Park Avenue 3rd Floor W. New York, NY 10169 USA 212-257-5797 Fax: 212-257-5796 Email:Ffabricant@fabricantllp.Com Jacob Daniel Ostling ATTORNEY TO BE NOTICED Fabricant LLP - Rye 411 Theodore Fremd Road Suite 206 South Rye, NY 10580 USA 212-257-5797 Fax: 212-257-5796 Email:Jostling@fabricantllp.Com Joseph Michael Mercadante ATTORNEY TO BE NOTICED Fabricant LLP - NY 230 Park Avenue 3rd Floor W. New York, NY 10169 USA 212-257-5797 Fax: 212-257-5796

Litigants

Attorneys

Email:Jmercadante@fabricantllp.Com

Julian Glenn Pymento
ATTORNEY TO BE NOTICED
Fabricant LLP
51 Jfk Parkway
Short Hills, NJ 07078
USA
646-797-4341 Email:Jpymento@fabricantllp.Com

Justin Kurt Truelove
ATTORNEY TO BE NOTICED
Truelove Law Firm
207 N Wellington P O Box 1409
Marshall, TX 75671
USA
903-938-8321 Fax: 903-215-8510
Email:Kurt@truelovelawfirm.Com

Peter Lambrianakos
ATTORNEY TO BE NOTICED
Fabricant LLP
411 Theodore Fremd Avenue Suite 206 South
Rye, NY 10580
USA
212-257-5797 Fax: 212-257-5796
Email:Plambrianakos@fabricantllp.Com

Vincent J Rubino , III
ATTORNEY TO BE NOTICED
Fabricant LLP - NY
230 Park Avenue 3rd Floor W.
New York, NY 10169
USA
212-257-5797 Fax: 212.257.5796
Email:Vrubino@fabricantllp.Com

Proceedings

#	Date	Proceeding Text	Source
1	11/01/2024	COMPLAINT FOR ***PATENT*** INFRINGEMENT against All Defendants (Filing fee $ 405 receipt number ATXEDC-10458847.), filed by Edge Networking Systems LLC. (Attachments: # 1 Exhibit A - U.S. ***Patent*** No. ***11,695,823***, # 2 Exhibit B - U.S. ***Patent*** No. 10,893,095, # 3 Exhibit C - U.S. ***Patent*** No. 10,686,871, # 4 Civil Cover Sheet)(Rubino, Vincent) (Entered: 11/01/2024)	
2	11/01/2024	Notice of Filing of ***Patent***/Trademark Form (AO 120). AO 120 emailed to the Director of the U.S. ***Patent*** and Trademark Office. (Rubino, Vincent) (Entered: 11/01/2024)	
3	11/01/2024	Fed. R. Civ. P. 7.1(a)(1) Disclosure Statement filed by Edge Networking Systems LLC (Rubino, Vincent) (Entered: 11/01/2024)	
4	11/01/2024	NOTICE of Attorney Appearance by Vincent J Rubino, III on behalf of Edge Networking Systems LLC (Rubino, Vincent) (Entered: 11/01/2024)	
5	11/01/2024	NOTICE of Attorney Appearance by Alfred Ross Fabricant on behalf of Edge Networking Systems LLC (Fabricant, Alfred)	

#	Date	Proceeding Text	Source
		(Entered: 11/01/2024)	
6	11/01/2024	NOTICE of Attorney Appearance by Peter Lambrianakos on behalf of Edge Networking Systems LLC (Lambrianakos, Peter) (Entered: 11/01/2024)	
	11/04/2024	District Judge Rodney Gilstrap and Magistrate Judge Roy S. Payne added. (KLC) (Entered: 11/04/2024)	
	11/04/2024	In accordance with the provisions of 28 USC Section 636(c), you are hereby notified that a U.S. Magistrate Judge of this district court is available to conduct any or all proceedings in this case including a jury or non-jury trial and to order the entry of a final judgment. The form Consent to Proceed Before Magistrate Judge is available on our website. All signed consent forms, excluding pro se parties, should be filed electronically using the event Notice Regarding Consent to Proceed Before Magistrate Judge. (KLC) (Entered: 11/04/2024)	
7	11/04/2024	SUMMONS Issued as to Amazon Web Services, Inc., Amazon.Com, Inc.,, Amazon.com Services LLC. (Attachments: # 1 Additional Attachment(s), # 2 Additional Attachment(s))(NKL) (Entered: 11/04/2024)	
8	11/13/2024	SUMMONS Returned Executed by Edge Networking Systems LLC. Amazon.Com, Inc., served on 11/5/2024. (Rubino, Vincent) (Entered: 11/13/2024)	
9	11/13/2024	SUMMONS Returned Executed by Edge Networking Systems LLC. Amazon.com Services LLC served on 11/12/2024. (Rubino, Vincent) (Entered: 11/13/2024)	
10	11/13/2024	SUMMONS Returned Executed by Edge Networking Systems LLC. Amazon Web Services, Inc. served on 11/5/2024. (Rubino, Vincent) (Entered: 11/13/2024)	
11	11/19/2024	Defendant's Unopposed First Application for Extension of Time to Answer Complaint re Amazon.Com, Inc.(Jones, Michael) (Entered: 11/19/2024)	
12	11/19/2024	Defendant's Unopposed First Application for Extension of Time to Answer Complaint re Amazon.com Services LLC.(Jones, Michael) (Entered: 11/19/2024)	
13	11/19/2024	Defendant's Unopposed First Application for Extension of Time to Answer Complaint re Amazon Web Services, Inc.(Jones, Michael) (Entered: 11/19/2024)	
	11/20/2024	Defendant's Unopposed First Application for Extension of Time to Answer Complaint is granted pursuant to Local Rule CV-12 for Amazon.Com, Inc., to 1/10/2025. 45 Days Granted for Deadline Extension.(NKL) (Entered: 11/20/2024)	
	11/20/2024	Defendant's Unopposed First Application for Extension of Time to Answer Complaint is granted pursuant to Local Rule CV-12 for Amazon.com Services LLC to 1/17/2025. 45 Days Granted for Deadline Extension.(NKL) (Entered: 11/20/2024)	
	11/20/2024	Defendant's Unopposed First Application for Extension of Time to Answer Complaint is granted pursuant to Local Rule CV-12 for Amazon Web Services, Inc. to 1/10/2025. 45 Days Granted for Deadline Extension.(NKL) (Entered: 11/20/2024)	
14	11/20/2024	NOTICE of Attorney Appearance by Shaun William Hassett on behalf of Amazon.Com, Inc.,, Amazon.com Services LLC, Amazon Web Services, Inc. (Hassett, Shaun) (Entered: 11/20/2024)	
15	11/20/2024	NOTICE of Attorney Appearance by Srinivas Giriraj Pathmanaban on behalf of Amazon.Com, Inc.,, Amazon.com Services LLC, Amazon Web Services, Inc. (Pathmanaban, Srinivas) (Entered: 11/20/2024)	
16	11/20/2024	NOTICE of Attorney Appearance by Clement Naples on behalf of	

#	Date	Proceeding Text	Source
		Amazon.Com, Inc.., Amazon.com Services LLC, Amazon Web Services, Inc. (Naples, Clement) (Entered: 11/20/2024)	
17	11/22/2024	NOTICE of Attorney Appearance by Joseph Michael Mercadante on behalf of Edge Networking Systems LLC (Mercadante, Joseph) (Entered: 11/22/2024)	
18	11/22/2024	NOTICE of Attorney Appearance by Julian Glenn Pymento on behalf of Edge Networking Systems LLC (Pymento, Julian) (Entered: 11/22/2024)	
19	11/22/2024	NOTICE of Attorney Appearance by Jacob Daniel Ostling on behalf of Edge Networking Systems LLC (Ostling, Jacob) (Entered: 11/22/2024)	
20	12/05/2024	NOTICE of Attorney Appearance - Pro Hac Vice by Michael E Bern on behalf of Amazon.Com, Inc.., Amazon.com Services LLC, Amazon Web Services, Inc.. Filing fee $ 100, receipt number ATXEDC-10534891. (Bern, Michael) (Entered: 12/05/2024)	
21	12/09/2024	UNOPPOSED MOTION for Extension of Time to File Answer re 1 Complaint, by Amazon.Com, Inc.., Amazon.com Services LLC, Amazon Web Services, Inc.. (Attachments: # 1 Proposed Order)(Jones, Michael) (Entered: 12/09/2024)	
22	12/10/2024	ORDER setting Scheduling Conference for 1/21/2025 at 01:30 PM before District Judge Rodney Gilstrap.. Signed by Magistrate Judge Roy S. Payne on 12/9/2024. (KLC) (Entered: 12/10/2024)	
23	12/10/2024	ORDER granting 21 UNOPPOSED MOTION for Extension of Time to File Answer re 1 Complaint, Amazon Web Services, Inc. answer due 2/9/2025; Amazon.Com, Inc., answer due 2/9/2025; Amazon.com Services LLC answer due 2/16/2025. Signed by Magistrate Judge Roy S. Payne on 12/10/2024. (NKL) Modified on 12/10/2024 (NKL). (Entered: 12/10/2024)	
24	12/12/2024	NOTICE of Attorney Appearance by Justin Kurt Truelove on behalf of Edge Networking Systems LLC (Truelove, Justin) (Entered: 12/12/2024)	
25	12/12/2024	JOINT MOTION for Extension of Time for Plaintiff to Comply with 3-1 and 3-2(Infringement Contentions) and Defendants to Comply with 3-3 and 3-4 (Invalidity Contentions) and Comply with Standing Order Regarding Subject-Matter Eligibility Contentions by Edge Networking Systems LLC. (Attachments: # 1 Proposed Order)(Rubino, Vincent) (Entered: 12/12/2024)	
26	12/16/2024	ORDER granting 25 Motion for Extension of Time to Comply with Infringement, Invalidity and Subject Matter Eligibility Contentions. Signed by Magistrate Judge Roy S. Payne on 12/13/2024. (NKL) (Entered: 12/16/2024)	
27	01/09/2025	JOINT MOTION for Second Extension of Time for Plaintiff to Comply with P.R. 3-1 and 3-2 (Infringement Contentions) and Defendants to Comply with P.R. 3-3 and 3-4 (Invalidity Contentions) and to Comply with Standing Order Regarding Subject-Matter Eligibility Contentions by Edge Networking Systems LLC. (Attachments: # 1 Proposed Order)(Rubino, Vincent) (Entered: 01/09/2025)	
28	01/10/2025	ORDER granting 27 Motion for Second Extension of Time for Plaintiff to Comply with P.R. 3-1 and 3-2 (Infringement Contentions) and Defendants to Comply with P.R. 3-3 and 3-4 (Invalidity Contentions) and to Comply with Standing Order Regarding Subject-Matter Eligibility Contentions. Signed by Magistrate Judge Roy S. Payne on 1/10/2025. (NKL) (Entered: 01/10/2025)	
	01/21/2025	Minute Entry for proceedings held before District Judge Rodney Gilstrap: Scheduling Conference/Case Management Conference held on 1/21/2025 from 01:32 PM - 02:35 PM. Counsel for the parties appeared. Court asked whether the parties consented to	

#	Date	Proceeding Text	Source
		trial before the U.S. Magistrate Judge. Court proceeded to provide counsel with Claim Construction and Jury Selection/Trial dates. (Court Reporter - Shawn McRoberts) (aeb) (Entered: 01/28/2025)	
29	02/04/2025	JOINT MOTION for Entry of Agreed Docket Control Order by Edge Networking Systems LLC. (Attachments: # 1 Exhibit A - Proposed Docket Control Order)(Rubino, Vincent) (Entered: 02/04/2025)	
30	02/04/2025	JOINT MOTION for Entry of Agreed Discovery Order by Edge Networking Systems LLC. (Attachments: # 1 Exhibit A - Proposed Discovery Order)(Rubino, Vincent) (Entered: 02/04/2025)	
31	02/05/2025	NOTICE OF COMPLIANCE by Edge Networking Systems LLC Regarding P.R. 3.1 & 3.2 Infringement Contentions (Rubino, Vincent) (Entered: 02/05/2025)	
32	02/06/2025	DOCKET CONTROL ORDER: Pretrial Conference set for 6/29/2026 at 09:00 AM before Magistrate Judge Roy S. Payne. Jury Selection set for 8/7/2026 at 09:00AM before District Judge Rodney Gilstrap. Markman Hearing set for 2/6/2026 at 09:00 AM before Magistrate Judge Roy S. Payne.. Signed by Magistrate Judge Roy S. Payne on 02/05/2025. (KLC) (Entered: 02/06/2025)	
33	02/06/2025	DISCOVERY ORDER granting 30 JOINT MOTION for Entry of Agreed Discovery Order. Signed by Magistrate Judge Roy S. Payne on 2/5/2025. (slo) (Entered: 02/06/2025)	
34	02/07/2025	UNOPPOSED MOTION for Extension of Time to File Answer re 1 Complaint, One-Day Extension by Amazon.Com, Inc.,, Amazon Web Services, Inc.. (Attachments: # 1 Proposed Order)(Jones, Michael) (Entered: 02/07/2025)	
35	02/09/2025	MOTION to Dismiss for Improper Venue by Amazon.Com, Inc.,, Amazon.com Services LLC, Amazon Web Services, Inc.. (Attachments: # 1 Hunt Declaration, # 2 Klein Declaration, # 3 Proposed Order)(Hassett, Shaun) (Entered: 02/09/2025)	
36	02/11/2025	NOTICE OF COMPLIANCE by Amazon.Com, Inc.,, Amazon.com Services LLC, Amazon Web Services, Inc. regarding Initial and Additional Disclosures (Hassett, Shaun) (Entered: 02/11/2025)	
37	02/11/2025	JOINT MOTION for Entry of Agreed Protective Order by Edge Networking Systems LLC. (Attachments: # 1 Exhibit A - Proposed Protective Order)(Rubino, Vincent) (Entered: 02/11/2025)	
38	02/12/2025	ORDER granting 34 UNOPPOSED MOTION for Extension of Time to File Answer re 1 Complaint, One-Day Extension Amazon Web Services, Inc. answer due 2/10/2025; Amazon.Com, Inc., answer due 2/10/2025. Signed by Magistrate Judge Roy S. Payne on 2/12/2025. (NKL) (Entered: 02/12/2025)	
39	02/12/2025	PROTECTIVE ORDER re 37 Motion for Entry of Agreed Protective Order. Signed by Magistrate Judge Roy S. Payne on 2/12/2025. (NKL) (Entered: 02/12/2025)	
40	02/14/2025	NOTICE OF COMPLIANCE by Edge Networking Systems LLC with Service of its Initial Disclosures (Rubino, Vincent) (Entered: 02/14/2025)	
41	02/16/2025	ANSWER to 1 Complaint, , COUNTERCLAIM against Edge Networking Systems LLC by Amazon.com Services LLC.(Pathmanaban, Srinivas) (Entered: 02/16/2025)	
42	02/19/2025	UNOPPOSED MOTION for Extension of Time to Respond to Defendants' Motion to Dismiss Complaint for Improper Venue by Edge Networking Systems LLC. (Attachments: # 1 Proposed Order)(Rubino, Vincent) (Entered: 02/19/2025)	
43	02/24/2025	ORDER granting 42 Motion for Extension of Time to Respond to Defendants' Motion to Dismiss Complaint for Improper Venue. Signed by Magistrate Judge Roy S. Payne on 2/21/2025. (NKL) (Entered: 02/24/2025)	

#	Date	Proceeding Text	Source
44	03/07/2025	UNOPPOSED MOTION to Extend Plaintiff's Deadline to Respond to Motion to Dismiss Complaint for Improper Venue (Dkt. 35) and to Answer Amazon.com Services LLC's Counterclaims (Dkt. 41) by Edge Networking Systems LLC. (Attachments: # 1 Proposed Order)(Rubino, Vincent) (Entered: 03/07/2025)	
45	03/10/2025	ORDER granting 44 Motion to Extend Deadline to File its Response in Opposition to Defendants Motion to Dismiss for Improper Venue and to Answer Amazon.com Services LLCs Counterclaims. Signed by Magistrate Judge Roy S. Payne on 3/8/2025. (NKL) (Entered: 03/10/2025)	
46	03/12/2025	OPPOSED MOTION for Leave to File Amended Complaint Against Amazon.com Services LLC by Edge Networking Systems LLC. (Attachments: # 1 Declaration of Vincent J. Rubino, III, # 2 Exhibit 1, # 3 Exhibit 2, # 4 Exhibit 3, # 5 Exhibit 4, # 6 Exhibit 5, # 7 Exhibit 6, # 8 Exhibit 7, # 9 Proposed Order)(Rubino, Vincent) (Entered: 03/12/2025)	
47	03/14/2025	UNOPPOSED MOTION to Further Extend Plaintiff's Deadline to File its Response in Opposition to Motion to Dismiss Complaint for Improper Venue (Dkt. 35) and to Answer Amazon.com Services LLC's Counterclaims (Dkt. 41) by Edge Networking Systems LLC. (Attachments: # 1 Proposed Order)(Rubino, Vincent) (Entered: 03/14/2025)	
48	03/18/2025	ORDER granting 47 Motion to Further Extend Deadline to File its Response in Opposition to Defendants Motion to Dismiss for Improper Venue and to Answer Amazon.com Services LLCs Counterclaims. Signed by Magistrate Judge Roy S. Payne on 3/17/2025. (NKL) (Entered: 03/18/2025)	
49	03/19/2025	NOTICE of Non-Opposition by Amazon.Com, Inc.,, Amazon.com Services LLC, Amazon Web Services, Inc. re 46 OPPOSED MOTION for Leave to File Amended Complaint Against Amazon.com Services LLC (Jones, Michael) (Entered: 03/19/2025)	
50	03/19/2025	JOINT MOTION to Amend/Correct 32 Scheduling Order/Docket Control Order, by All Parties. (Attachments: # 1 Proposed Order)(Jones, Michael) (Entered: 03/19/2025)	
51	03/21/2025	ORDER granting 46 Motion for Leave to File. Signed by Magistrate Judge Roy S. Payne on 03/20/2025. (KLC) (Entered: 03/21/2025)	
52	03/21/2025	FIRST AMENDED DOCKET CONTROL ORDER. Signed by Magistrate Judge Roy S. Payne on 03/20/2025. (KLC) (Entered: 03/21/2025)	
53	03/24/2025	AMENDED COMPLAINT FOR ***PATENT*** INFRINGEMENT against Amazon.com Services LLC, filed by Edge Networking Systems LLC. (Attachments: # 1 Exhibit A - U.S. ***Patent*** No. ***11,695,823***, # 2 Exhibit B - U.S. ***Patent*** No. 10,893,095, # 3 Exhibit C - U.S. ***Patent*** No. 10,686,871, # 4 Exhibit D - U.S. ***Patent*** No. 12,113,850, # 5 Exhibit E - U.S. ***Patent*** No. 12,113,851, # 6 Exhibit F - U.S. ***Patent*** No. 12,126,673, # 7 Exhibit G - U.S. ***Patent*** No. 12,126,674)(Rubino, Vincent) (Entered: 03/24/2025)	
54	03/24/2025	AMENDED Notice of Filing of ***Patent***/Trademark Form (AO 120). AO 120 emailed to the Director of the U.S. ***Patent*** and Trademark Office. (Rubino, Vincent) (Entered: 03/24/2025)	
55	03/24/2025	NOTICE of Attorney Appearance by Klaus H Hamm on behalf of Amazon.com Services LLC, Amazon.Com, Inc.,, Amazon Web Services, Inc. (Hamm, Klaus) (Entered: 03/24/2025)	
56	03/24/2025	NOTICE of Attorney Appearance by Robert Cruzen on behalf of Amazon.com Services LLC, Amazon.Com, Inc.,, Amazon Web Services, Inc. (Cruzen, Robert) (Entered: 03/24/2025)	

#	Date	Proceeding Text	Source
57	04/07/2025	UNOPPOSED MOTION for Extension of Time to File Answer re 53 Amended Complaint, by Amazon.com Services LLC. (Attachments: # 1 Proposed Order)(Hassett, Shaun) (Entered: 04/07/2025)	
58	04/07/2025	NOTICE of Mootness by Amazon.com Services LLC re 35 MOTION to Dismiss for Improper Venue (Hassett, Shaun) (Entered: 04/07/2025)	
59	04/09/2025	ANSWER to 53 Amended Complaint, , COUNTERCLAIM against Edge Networking Systems LLC by Amazon.com Services LLC.(Jones, Michael) (Entered: 04/09/2025)	
60	04/09/2025	ORDER granting 57 Motion for Extension of Time to Answer Amazon.com Services LLC answer due 4/10/2025. Signed by Magistrate Judge Roy S. Payne on 4/9/25. (bga) (Entered: 04/09/2025)	
61	04/09/2025	ORDER denying as moot 35 Motion to Dismiss. Signed by Magistrate Judge Roy S. Payne on 4/9/25. (bga) (Entered: 04/09/2025)	
62	04/14/2025	UNOPPOSED MOTION to Withdraw as Attorney Michael E. Bern by Amazon.com Services LLC. (Attachments: # 1 Proposed Order)(Hassett, Shaun) (Entered: 04/14/2025)	
63	04/16/2025	ORDER granting 62 Motion to Withdraw as Attorney. Attorney Michael E Bern terminated as counsel for Amazon.com Services LLC. Signed by Magistrate Judge Roy S. Payne on 04/15/2025. (KLC) (Entered: 04/16/2025)	
64	04/18/2025	MOTION to Dismiss Defendant Amazon.com Services LLC's Counterclaims and to Strike Defendant's Affirmative Defenses (Dkt. 59) by Edge Networking Systems LLC. (Attachments: # 1 Proposed Order)(Rubino, Vincent) (Entered: 04/18/2025)	
65	05/02/2025	AMENDED ANSWER to 53 Amended Complaint, by Amazon.com Services LLC. (Jones, Michael) (Entered: 05/02/2025)	

Copyright © LexisNexis CourtLink, Inc. All Rights Reserved.
*** THIS DATA IS FOR INFORMATIONAL PURPOSES ONLY ***

End of Document

User Name: Patricia Martin
Date and Time: Tuesday, May 13, 2025 7:23 AM EDT
Job Number: 252698603

Document (1)

1. 1:24cv215, Edge Networking Systems, Llc V. Microsoft Corporation
   Client/Matter: -None-
   Search Terms: 11695823 or 11,695,823
   Search Type: Terms and Connectors
   Narrowed by:

Content Type	Narrowed by
Dockets	Case Status: Open,Unknown,Closed; Case Type: Civil

LexisNexis | About LexisNexis | Privacy Policy | Terms & Conditions | Copyright © 2025 LexisNexis

1:24cv215, Edge Networking Systems, Llc V. Microsoft Corporation

US District Court Docket

United States District Court, Texas Western

(Austin)

This case was retrieved on 05/13/2025

Header

Case Number: 1:24cv215
Date Filed: 02/29/2024
Assigned To: Judge David A. Ezra
Nature of Suit: Patent (830)
Cause: Patent Infringement
Lead Docket: None
Other Docket: None
Jurisdiction: Federal Question

Class Code: Open
Statute: 35:100
Jury Demand: Both
Demand Amount: $0
NOS Description: Patent

Participants

Litigants

Edge Networking Systems, LLC
Plaintiff

Microsoft Corporation
Defendant

Attorneys

David L. Hecht
ATTORNEY TO BE NOTICED
Hecht Partners LLP
125 Park Avenue, 25th Floor
New York, NY 10017
USA
(212) 851-6821 Fax: (646) 492-5111
Email:Dhecht@hechtpartners.Com

Irene Yang
LEAD ATTORNEY;ATTORNEY TO BE NOTICED
Sidley Austin LLP
555 California Street, Suite 2000
San Francisco, CA 94104
USA
415-772-1200 Fax: 415-772-7400
Email:Irene.Yang@sidley.Com

Richard M. Chen
LEAD ATTORNEY;ATTORNEY TO BE NOTICED
Sidley Austin LLP
One South Dearborn Street
Chicago, IL 60603
USA
(312) 853-7000 Fax: (312) 853-7036
Email:Rchen@sidley.Com

Andrew W. Stinson
ATTORNEY TO BE NOTICED
Ramey & Flock, PC
100 E Ferguson Street Suite 500

Patricia Martin

Litigants

Attorneys

Tyler, TX 75702
USA
903-597-3301 Email:Astinson@rameyflock.Com

Brooke Shanelle Boll
ATTORNEY TO BE NOTICED
Sidley Austin LLP
. 350 South Grand Avenue
Los Angeles, CA 90071
USA
213-896-6000 Fax: 213-896-6600
Email:Brooke.Boll@sidley.Com

James Travis Underwood
ATTORNEY TO BE NOTICED
Gillam & Smith
102 N. College, Suite 800
Tyler, TX 75702
USA
(903) 934-8450 Fax: (903) 934-9257
Email:Travis@gillamsmithlaw.Com

Melissa Richards Smith
ATTORNEY TO BE NOTICED
Gillam and Smith, LLP
303 South Washington Avenue
Marshall, TX 75670
USA
(903)934-8450 Fax: (903)934-9257
Email:Melissa@gillamsmithlaw.Com

Morgan R. Mendicino
ATTORNEY TO BE NOTICED
Sidley Austin LLP
2021 Mckinney Avenue, Suite 2000
Dallas, TX 75201
USA
(214) 981-3300 Fax: (214) 981-3400
Email:Mmendicino@sidley.Com

Proceedings

#	Date	Proceeding Text	Source
1	02/29/2024	COMPLAINT for Patent Infringement (Filing fee $ 405 receipt number ATXWDC-18479169), filed by Edge Networking Systems, LLC. (Attachments: # 1 Exhibit 1 - U.S. Patent No. 10,686,871, # 2 Exhibit 2 - U.S. Patent No. 10,893,095, # 3 Exhibit 3 - U.S. Patent No. 11,695,823, # 4 Civil Cover Sheet)(Hecht, David) (Entered: 02/29/2024)	
2	02/29/2024	Notice of Filing of Patent/Trademark Form (AO 120). AO 120 forwarded to the Director of the U.S. Patent and Trademark Office. (Hecht, David) (Entered: 02/29/2024)	
3	02/29/2024	REQUEST FOR ISSUANCE OF SUMMONS by Edge Networking Systems, LLC. (Hecht, David) (Entered: 02/29/2024)	
	02/29/2024	Case assigned to Judge Docket II - Austin. CM WILL NOW REFLECT THE JUDGE INITIALS AS PART OF THE CASE NUMBER. PLEASE APPEND THESE JUDGE INITIALS TO THE	

#	Date	Proceeding Text	Source
		CASE NUMBER ON EACH DOCUMENT THAT YOU FILE IN THIS CASE. (kp) (Entered: 03/01/2024)	
	02/29/2024	If ordered by the court, all referrals and consents in this case will be assigned to Magistrate Judge Susan Hightower. (kp) (Entered: 03/01/2024)	
	02/29/2024	DEMAND for Trial by Jury by Edge Networking Systems, LLC. (kp) (Entered: 03/01/2024)	
4	03/01/2024	CORRECTED COMPLAINT filed per Clerk's Instructions, corrected to correct case heading against Microsoft Corporation amending 1 Complaint,., filed by Edge Networking Systems, LLC.(Hecht, David) Modified on 3/4/2024 (dm). (Entered: 03/01/2024)	
5	03/01/2024	Summons Issued as to Microsoft Corporation. (kp) (Entered: 03/01/2024)	
6	03/26/2024	SUMMONS Returned Executed by Edge Networking Systems, LLC. Microsoft Corporation served on 3/5/2024, answer due 3/26/2024. (Hecht, David) (Entered: 03/26/2024)	
7	03/26/2024	NOTICE/MOTION for Extension of Time to Respond to Complaint by Edge Networking Systems, LLC re 4 Amended Complaint (Hecht, David) Modified on 3/27/2024 to create motion and correct event type (dm). (Entered: 03/26/2024)	
	03/27/2024	Text Order GRANTING 7 Motion for Extension of Time to Answer entered by Judge Robert Pitman. IT IS ORDERED that Defendant shall answer or otherwise respond to Plaintiff's complaint on or before May 9, 2024. (This is a text-only entry generated by the court. There is no document associated with this entry.) (tclc) (Entered: 03/27/2024)	
	03/27/2024	Reset Deadlines: Microsoft Corporation answer due 5/9/2024. (dm) (Entered: 03/28/2024)	
8	05/09/2024	NOTICE of Attorney Appearance by Irene Yang on behalf of Microsoft Corporation. Attorney Irene Yang added to party Microsoft Corporation(pty:dft) (Yang, Irene) (Entered: 05/09/2024)	
9	05/09/2024	NOTICE of Attorney Appearance by Brooke S. Boll on behalf of Microsoft Corporation. Attorney Brooke S. Boll added to party Microsoft Corporation(pty:dft) (Boll, Brooke) (Entered: 05/09/2024)	
10	05/09/2024	NOTICE of Attorney Appearance by Morgan R. Mendicino on behalf of Edge Networking Systems, LLC. Attorney Morgan R. Mendicino added to party Microsoft Corporation (pty:pla) (Mendicino, Morgan) Modified docket text to update filer on 5/9/2024 (pg). (Entered: 05/09/2024)	
11	05/09/2024	NOTICE of Attorney Appearance by Melissa Richards Smith on behalf of Microsoft Corporation. Attorney Melissa Richards Smith added to party Microsoft Corporation(pty:dft) (Smith, Melissa) (Entered: 05/09/2024)	
12	05/09/2024	RULE 7 DISCLOSURE STATEMENT filed by Microsoft Corporation. (Smith, Melissa) (Entered: 05/09/2024)	
13	05/09/2024	MOTION to Dismiss Plaintiff's Corrected Complaint (Dkt. No. 4) Pursuant to Rule 12(b)(6) re 4 Amended Complaint by Microsoft Corporation. (Attachments: # 1 Proposed Order)(Smith, Melissa) (Entered: 05/09/2024)	
14	05/23/2024	Memorandum in Opposition to Motion, filed by Edge Networking Systems, LLC, re 13 MOTION to Dismiss Plaintiff's Corrected Complaint (Dkt. No. 4) Pursuant to Rule 12(b)(6) re 4 Amended Complaint filed by Defendant Microsoft Corporation (Hecht, David) (Entered: 05/23/2024)	
15	05/30/2024	MOTION to Appear Pro Hac Vice by Irene Yang for Richard M. Chen (Filing fee $ 100 receipt number ATXWDC-18824469) by on behalf of Microsoft Corporation. (Yang, Irene) (Entered:	

#	Date	Proceeding Text	Source
		05/30/2024)	
16	05/30/2024	REPLY to Response to Motion, filed by Microsoft Corporation, re 13 MOTION to Dismiss Plaintiff's Corrected Complaint (Dkt. No. 4) Pursuant to Rule 12(b)(6) re 4 Amended Complaint filed by Defendant Microsoft Corporation (Smith, Melissa) (Entered: 05/30/2024)	
17	06/03/2024	ORDER GRANTING 15 Motion to Appear Pro Hac Vice as to Richard Chen. Pursuant to our Administrative Policies and Procedures for Electronic Filing, the attorney hereby granted to practice pro hac vice in this case must register for electronic filing with our court within 10 days of this order. Registration is managed by the PACER Service Center. Signed by Judge Robert Pitman. (dm) (Entered: 06/03/2024)	
18	07/17/2024	Order for Proposed Scheduling Order. Proposed scheduling order due to the Court within sixty (60) days after the appearance of any defendant. Proposed Scheduling Order due by 7/29/2024. Signed by Judge Susan Hightower. (dm) (Entered: 07/17/2024)	
19	07/29/2024	Proposed Order -Proposed Scheduling Order to 18 Order for Scheduling Recommendations/Proposed Scheduling Order by Microsoft Corporation. (Yang, Irene) (Entered: 07/29/2024)	
	07/31/2024	DEFICIENCY NOTICE: re 19 Proposed Scheduling Order. Your pleading is deficient because the proposed order does not contain a signature line for the judges signature. Please correct your proposed order to include a signed on date line and a signature line for the presiding judge then resubmit. (cnr) (Entered: 07/31/2024)	
20	08/01/2024	Proposed Order /(Corrected) Proposed Scheduling Order to 19 Proposed Order by Microsoft Corporation. (Yang, Irene) (Entered: 08/01/2024)	
21	08/02/2024	NOTICE CONCERNING REFERENCE TO UNITED STATES MAGISTRATE JUDGE by Microsoft Corporation (Smith, Melissa) (Entered: 08/02/2024)	
22	08/06/2024	ORDER Setting Hearing on Initial Pretrial Telephonic Conference set for 08/21/2024 at 10:30 A.M. Signed by Magistrate Judge Susan Hightower. (cnr) (Entered: 08/06/2024)	
23	08/09/2024	TRANSFER ORDER. Case reassigned to Judge David A. Ezra for all proceedings. Judge Docket II - Austin no longer assigned to case. Signed by Judge Robert Pitman. (cnr) (Entered: 08/09/2024)	
24	08/21/2024	NOTICE of Attorney Appearance by James Travis Underwood on behalf of Microsoft Corporation. Attorney James Travis Underwood added to party Microsoft Corporation(pty:dft) (Underwood, James) (Entered: 08/21/2024)	
25	08/21/2024	Minute Entry for proceedings held before Judge Susan Hightower: Initial Pretrial Conference held on 8/21/2024. Written order forthcoming. (Minute entry documents are not available electronically.). (Court Reporter AT&T Conference Line.)(dm) (Entered: 08/22/2024)	
26	08/28/2024	Proposed Scheduling Order by Microsoft Corporation. (Smith, Melissa) (Entered: 08/28/2024)	
27	09/04/2024	SCHEDULING ORDER: ADR Report Deadline due by 3/21/2025, Amended Pleadings due by 3/21/2025, Fact Discovery due by 6/27/2025, Joinder of Parties due by 3/21/2025, Dispositive Motions due by 11/17/2025. Signed by Judge David A. Ezra. (dm) (Entered: 09/04/2024)	
28	10/16/2024	Proposed Protective Order by Edge Networking Systems, LLC. (Hecht, David) (Entered: 10/16/2024)	
29	10/17/2024	PROTECTIVE ORDER. Signed by Judge David A. Ezra. (dm) (Entered: 10/17/2024)	

#	Date	Proceeding Text	Source
30	10/25/2024	NOTICE of Attorney Appearance by Andrew W. Stinson on behalf of Microsoft Corporation. Attorney Andrew W. Stinson added to party Microsoft Corporation(pty:dft) (Stinson, Andrew) (Entered: 10/25/2024)	
31	12/17/2024	Opening Claim Construction Brief by Edge Networking Systems, LLC. (Attachments: # 1 Declaration of David L. Hecht, # 2 Exhibit A - EDGE_MSFT- 00000885, # 3 Exhibit B - EDGE_MSFT- 00000887, # 4 Exhibit C - Preliminary Invalidity Contentions Cover Pleading)(Hecht, David) (Entered: 12/17/2024)	
32	01/17/2025	BRIEF -Responsive Claim Construction Brief regarding 31 Claim Construction Brief, by Microsoft Corporation. (Attachments: # 1 Declaration of Brooke Boll, # 2 Exhibit 1 - Declaration of Dr. Jon Weissman, # 3 Appendix A to Weissman Dec., # 4 Appendix B to Weissman Dec., # 5 Exhibit 2, # 6 Exhibit 3, # 7 Exhibit 4, # 8 Exhibit 5, # 9 Exhibit 6, # 10 Exhibit 7, # 11 Exhibit 8, # 12 Exhibit 9, # 13 Exhibit 10, # 14 Exhibit 11, # 15 Exhibit 12, # 16 Exhibit 13, # 17 Exhibit 14)(Yang, Irene) (Entered: 01/17/2025)	
33	01/21/2025	Joint MOTION Requesting Markman Hearing Date by Microsoft Corporation. (Attachments: # 1 Proposed Order)(Smith, Melissa) (Entered: 01/21/2025)	
34	01/22/2025	ORDER DENYING 13 Motion to Dismiss. Signed by Judge David A. Ezra. (dm) (Entered: 01/22/2025)	
35	02/05/2025	ANSWER to 4 Amended Complaint with Jury Demand AND AFFIRMATIVE DEFENSES by Microsoft Corporation.(Smith, Melissa) (Entered: 02/05/2025)	
36	03/05/2025	STIPULATION to Extend Time by Edge Networking Systems, LLC. (Hecht, David) (Entered: 03/05/2025)	
37	03/06/2025	ORDER Granting Extension re 36 Stipulation to Extend Time filed by Edge Networking Systems, LLC. Signed by Judge David A. Ezra. (dm) (Entered: 03/06/2025)	
38	03/21/2025	ADR Report Filed - CONSENT to Alternate Dispute Resolution (ADR) by Edge Networking Systems, LLC(Hecht, David) (Entered: 03/21/2025)	

Copyright © LexisNexis CourtLink, Inc. All Rights Reserved.
*** THIS DATA IS FOR INFORMATIONAL PURPOSES ONLY ***

End of Document

User Name: Patricia Martin
Date and Time: Tuesday, May 13, 2025 7:25 AM EDT
Job Number: 252698665

Document (1)

1. IPR2025-00619, Microsoft Corporation Vs. Edge Networking Systems, LLC
   Client/Matter: -None-
   Search Terms: 11695823 or 11,695,823
   Search Type: Terms and Connectors
   Narrowed by:

Content Type	Narrowed by
Dockets	Case Status: Open,Unknown,Closed; Case Type: Civil

LexisNexis | About LexisNexis | Privacy Policy | Terms & Conditions | Copyright © 2025 LexisNexis

IPR2025-00619, Microsoft Corporation Vs. Edge Networking Systems, LLC

US Patent Trial and Appeals Board - Alexandria

Alexandria

This case was retrieved on 05/13/2025

Header

Case Number: IPR2025-00619
Date Filed: 02/17/2025
Date Full Case Retrieved: 05/13/2025
Status: Open
Misc: (0) Unknown; Civil

Summary

Court Case Status: Notice of filing date accorded
Case Type: IPR: Inter partes review
Technical Center Number: 2400
Patent Application Number: 15836824
Patent Number: 10686871

Participants

Litigants	Attorneys
Microsoft Corporation **Petitioner** Edge Networking Systems, LLC **PatentOwner**	

Proceedings

Date	#	Proceeding Text	Details
02/17/2025	1	1072 Scheduling Order, filed as Document No. 27, on September 4, 2024	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1072 Public? Yes
02/17/2025	2	1071 Defendant Microsoft Corporation???s Responsive Claim Construction Brief	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1071 Public? Yes
02/17/2025	3	1070 Plaintiff Edge Networking Systems, LLC???s Opening Claim Construction	Filed By: PETITIONER Document Type: EXHIBIT

Patricia Martin

Date	#	Proceeding Text	Details
			Paper/Exhibit No: 1070 Public? Yes
02/17/2025	4	1069 ???Avalanche Effect???, Wikipedia, archived on April 30, 2010 at Internet	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1069 Public? Yes
02/17/2025	5	1068 J. Preshing, ???Hash Collision Probabilities???, Preshing on Programming	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1068 Public? Yes
02/17/2025	6	1067 U.S. Department of Commerce, FIPS PUB 186-2, Digital Signature	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1067 Public? Yes
02/17/2025	7	1066 ???Apache Archive Distribution Directory???, Apache Software Foundation	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1066 Public? Yes
02/17/2025	8	1065 ???apache_1.3.19.tar.gz.asc??? Signature File, Apache Software Foundation	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1065 Public? Yes
02/17/2025	9	1064 ???apache_1.3.19.tar.gz.md5??? Hash File, Apache Software Foundation	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1064 Public? Yes
02/17/2025	10	1063 Excerpts from Stallings, Operating Systems Internals and Design	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1063 Public? Yes

Date	#	Proceeding Text	Details
02/17/2025	11	1062 Excerpts from Benevenuti, Understanding Linux Network Internals (2006)	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1062 Public? Yes
02/17/2025	12	1061 Excerpts from McKusick and Neville-Neil, The Design and Implementation	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1061 Public? Yes
02/17/2025	13	1060 Excerpts from Silberschatz et al., Operating System Concepts	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1060 Public? Yes
02/17/2025	14	1059 Excerpts from Silberschatz et al., Operating System Concepts	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1059 Public? Yes
02/17/2025	15	1058 Usenix Jails in FreeBSD for Fun and Profit, Paco Hope, USENIX ;login	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1058 Public? Yes
02/17/2025	16	1057 Microsoft Computer Dictionary, 3rd ed., 1997, excerpts	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1057 Public? Yes
02/17/2025	17	1056 ???Using Java??? RMI with SSL???, Oracle Java SE Documentation	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1056 Public? Yes
02/17/2025	18	1055 ???Using Custom Socket Factories with Java RMI???, Oracle Java SE Document	Filed By: PETITIONER Document Type: EXHIBIT

Date	#	Proceeding Text	Details
			Paper/Exhibit No: 1055 Public? Yes
02/17/2025	19	1054 ???Java??? Remote Method Invocation API (Java RMI)???, Oracle Java SE	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1054 Public? Yes
02/17/2025	20	1053 ???Java Secure Socket Extension (JSSE) Reference Guide for Java Platform	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1053 Public? Yes
02/17/2025	21	1052 Excerpts from B. Sosinsky, ???Cloud Computing Bible???, Wiley Publishing	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1052 Public? Yes
02/17/2025	22	1051 R. Hall, ???Oscar Bundle Repository???, archived on June 30, 2004	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1051 Public? Yes
02/17/2025	23	1050 The Open Services Gateway Initiative, ???Specification Overview???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1050 Public? Yes
02/17/2025	24	1048 ???REST Security Cheat Sheet???, The Open Web Application Security Project	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1048 Public? Yes
02/17/2025	25	1047 ???Web Service Security Cheat Sheet???, The Open Web Application Security	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1047 Public? Yes

Date	#	Proceeding Text	Details
02/17/2025	26	1046 Oasis Standard Specification, ???Web Services Security: SOAP Message	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1046 Public? Yes
02/17/2025	27	1045 Excerpts from S. Garfinkel, et al., ???Web Security & Commerce???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1045 Public? Yes
02/17/2025	28	1044 Excerpts from A.S. Tanenbaum, ???Modern Operating Systems???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1044 Public? Yes
02/17/2025	29	1043 R. Kawashima, ???vNFC: A Virtual Networking Function Container for SDN-e	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1043 Public? Yes
02/17/2025	30	1042 ???Knoplerfish Pro Premium??? and ???Knoplerfish Pro Enterprise??? Data Sheets	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1042 Public? Yes
02/17/2025	31	1041 P. Kriens, ???The Bundle Repository???, OSGi Alliance Blog, April 7, 2006	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1041 Public? Yes
02/17/2025	32	1039 Li, et al., ???ExpoNet: A Flexible Platform for Concurrent Experiments	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1039 Public? Yes
02/17/2025	33	1036 S. Soltesz, et al., ???Container-based Operating System Virtualization:	Filed By: PETITIONER Document Type: EXHIBIT

Date	#	Proceeding Text	Details
			Paper/Exhibit No: 1036 Public? Yes
02/17/2025	34	1035 Curriculum Vitae (CV) of Petitioner???s Expert Dr. Erez Zadok.	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1035 Public? Yes
02/17/2025	35	1034 Excerpts from E. Cole, et al., ???Network Security Bible???, Wiley Publish	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1034 Public? Yes
02/17/2025	36	1033 Excerpts from D. Downing, et al., ???Dictionary of Computer and Internet	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1033 Public? Yes
02/17/2025	37	1032 Excerpts from A.S. Tannenbaum, ???Computer Networks???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1032 Public? Yes
02/17/2025	38	1031 Excerpts from A.S. Tanenbaum, ???Modern Operating Systems???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1031 Public? Yes
02/17/2025	39	1030 U.S. Patent Publication No. 2013/0086147 to Kashyap	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1030 Public? Yes
02/17/2025	40	1029 T. Forst, ???Cisco Application eXtension Platform ??? OSGi Add-on	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1029 Public? Yes

Date	#	Proceeding Text	Details
02/17/2025	41	1028 The OSGi Alliance, ???OSGi Service Platform Enterprise Specification???	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1028 Public? Yes
02/17/2025	42	1027 ???About Makewave: History in Making???, Makewave AB website	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1027 Public? Yes
02/17/2025	43	1026 ???About Us: Background???, Gatespace AB website	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1026 Public? Yes
02/17/2025	44	1025 ???Management Team???, Gatespace AB website, archived on February 8, 2002	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1025 Public? Yes
02/17/2025	45	1024 ???Open Services Gateway Initiative Elects New Leadership???, Home Toys In	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1024 Public? Yes
02/17/2025	46	1023 ???Open Services Gateway Initiative: Officers???, OSGi website	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1023 Public? Yes
02/17/2025	47	1022 ???Open Services Gateway Initiative: Charter???, OSGi website	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1022 Public? Yes
02/17/2025	48	1021 ???People???, Carlstedt Research & Technology AB website	Filed By: PETITIONER Document Type: EXHIBIT

Date	#	Proceeding Text	Details
			Paper/Exhibit No: 1021 Public? Yes
02/17/2025	49	1020 ???Fifteen industry leaders to create standard for bringing Internet	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1020 Public? Yes
02/17/2025	50	1019 ???Ericsson e-services is a part of the Open Services Gateway Initiative	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1019 Public? Yes
02/17/2025	51	1018 ???Carlstedt Research & Technology Forms New Company for Developing Soft	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1018 Public? Yes
02/17/2025	52	1017 ???About Gatespace???, Gatespace AB website, archived on January 6, 2000	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1017 Public? Yes
02/17/2025	53	1016 U.S. Provisional Patent Application No. 60/123,971	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1016 Public? Yes
02/17/2025	54	1015 U.S. Provisional Patent Application No. 60/088,437	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1015 Public? Yes
02/17/2025	55	1014 Declaration of June A. Munford	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1014 Public? Yes

Date	#	Proceeding Text	Details
02/17/2025	56	1013 OSGi Alliance, ???About the OSGi Service Platform???, Technical Whitepaper	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1013 Public? Yes
02/17/2025	57	1012 S. K??chele, et al., ???Component-based scalability for cloud application	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1012 Public? Yes
02/17/2025	58	1011 J. S. Rellermeyer and S. Bagchi, ???Dependability as a cloud service	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1011 Public? Yes
02/17/2025	59	1010 J. E. Kim, et al., ???Seamless Integration of Heterogeneous Devices	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1010 Public? Yes
02/17/2025	60	1009 Excerpts from R. Hall, et al., ???OSGi in Action - Creating Modular	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1009 Public? Yes
02/17/2025	61	1008 Excerpts from A. de Castro Alves, ???OSGi in Depth???, Manning Publication	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1008 Public? Yes
02/17/2025	62	1007 File history of related U.S. Patent No. 11,695,823 (the ??????823 Patent???)	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1007 Public? Yes
02/17/2025	63	1006 File history of related U.S. Patent No. 10,893,095 (the ??????095 Patent???)	Filed By: PETITIONER Document Type: EXHIBIT

Date	#	Proceeding Text	Details
			Paper/Exhibit No: 1006 Public? Yes
02/17/2025	64	1005 File history of related U.S. Patent No. 9,483,624 (the ??????624 Patent???)	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1005 Public? Yes
02/17/2025	65	1004 U.S. Patent No. 6,496,575 to Vasell, et al.	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1004 Public? Yes
02/17/2025	66	1003 Declaration of Petitioner???s Expert, Dr. Erez Zadok, (Zadok)	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1003 Public? Yes
02/17/2025	67	1002 Prosecution history of the ???871 Patent (??????871 File History???)	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1002 Public? Yes
02/17/2025	68	1001 U.S. Patent No. 10,686,871 (the ??????871 Patent???)	Filed By: PETITIONER Document Type: EXHIBIT Paper/Exhibit No: 1001 Public? Yes
04/15/2025	69	Notice: NOTICE OF FILING DATE ACCORDED TO PETITION AND TIME FOR FILING PATENT OWNER PRELIMINARY RESPONSE	Filed By: BOARD Document Type: PAPER Paper/Exhibit No: 5 Public? Yes
03/31/2025	70	Notice: Power of Attorney	Filed By: PATENT OWNER Document Type: PAPER Paper/Exhibit No: 4 Public? Yes

Date	#	Proceeding Text	Details
03/31/2025	71	Notice: Mandatory Notice	Filed By: PATENT OWNER Document Type: PAPER Paper/Exhibit No: 3 Public? Yes
02/17/2025	72	Petition: as filed	Filed By: PETITIONER Document Type: PAPER Paper/Exhibit No: 2 Public? Yes
02/17/2025	73	Notice: Power of Attorney	Filed By: PETITIONER Document Type: PAPER Paper/Exhibit No: 1 Public? Yes

Copyright © LexisNexis CourtLink, Inc. All Rights Reserved.
*** THIS DATA IS FOR INFORMATIONAL PURPOSES ONLY ***

End of Document

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

REEXAM CONTROL NUMBER	FILING OR 371 (c) DATE	PATENT NUMBER
90/019,948	05/06/2025	11695823

CONFIRMATION NO. 2943

REEXAMINATION REQUEST NOTICE

Amundsen Davis, LLC
111 East Kilbourn Avenue
Suite 1400
Milwaukee, WISCONSIN 53202

OC000000088400553

Date Mailed: 05/09/2025

NOTICE OF REEXAMINATION REQUEST FILING DATE

(Third Party Requester)

Requester is hereby notified that the filing date of the request for reexamination is 05/06/2025, the date that the filing requirements of 37 CFR § 1.510 were received.

A decision on the request for reexamination will be mailed within three months from the filing date of the request for reexamination. (See 37 CFR 1.515(a)).

A copy of the Notice is being sent to the person identified by the requester as the patent owner. Further patent owner correspondence will be the latest attorney or agent of record in the patent file. (See 37 CFR 1.33). Any paper filed should include a reference to the present request for reexamination (by Reexamination Control Number).

cc: Patent Owner
112449
Roark IP
1438 Dahlia Loop
San Jose, CA 95126

/rbell/

Legal Instruments Examiner
Central Reexamination Unit 571-272-7705; FAX No. 571-273-9900

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

REEXAM CONTROL NUMBER	FILING OR 371 (c) DATE	PATENT NUMBER
90/019,948	05/06/2025	11695823

CONFIRMATION NO. 2943

REEXAM ASSIGNMENT NOTICE

OC000000088400555

112449
Roark IP
1438 Dahlia Loop
San Jose, CA 95126

Date Mailed: 05/09/2025

NOTICE OF ASSIGNMENT OF REEXAMINATION REQUEST

The above-identified request for reexamination has been assigned to Art Unit 3992. All future correspondence to the proceeding should be identified by the control number listed above and directed to the assigned Art Unit.

A copy of this Notice is being sent to the latest attorney or agent of record in the patent file or to all owners of record. (See 37 CFR 1.33(c)). If the addressee is not, or does not represent, the current owner, he or she is required to forward all communications regarding this proceeding to the current owner(s). An attorney or agent receiving this communication who does not represent the current owner(s) may wish to seek to withdraw pursuant to 37 CFR 1.36 in order to avoid receiving future communications. If the address of the current owner(s) is unknown, this communication should be returned within the request to withdraw pursuant to Section 1.36.

NOTICE OF USPTO EX PARTE REEXAMINATION PATENT OWNER STATEMENT WAIVER PROGRAM

The USPTO has implemented a pilot program where, after a reexamination proceeding has been granted a filing date and before the examiner begins his or her review, the patent owner may orally waive the right to file a patent owner's statement. See *"Pilot Program for Waiver of Patent Owner's Statement in Ex Parte Reexamination Proceedings,"* 75 FR 47269 (August 5, 2010). One goal of the pilot program is to reduce the pendency of reexamination proceedings and improve the efficiency of the reexamination process.

Ordinarily when ex parte reexamination is ordered, the USPTO must wait until after the receipt of the patent owner's statement and the third party requester's reply, or after the expiration of the time period for filing the statement and reply (a period that can be as long as 5 to 6 months), before mailing a first determination of patentability. The USPTO's first determination of patentability is usually a first Office action on the merits or a Notice of Intent to Issue Reexamination Certificate (NIRC).

Under the pilot program, the patent owner's oral waiver allows the USPTO to act on the first determination of patentability immediately after determining that reexamination will be ordered, and in a suitable case issue the reexamination order and the first determination of patentability (which could be a NIRC if the claims under reexamination are confirmed) at the same time.

Benefits to the Patent Owner for participating in this pilot program include reduction in pendency.

To participate in this pilot program, Patent Owners may contact the USPTO's Central Reexamination Unit (CRU) at 571-272-7705. The USPTO will make the oral waiver of record in the reexamination file in an interview summary and a copy will be mailed to the patent owner and any third party requester.

cc: Third Party Requester(if any)
Amundsen Davis, LLC
111 East Kilbourn Avenue
Suite 1400
Milwaukee, WISCONSIN 53202

/rbell/

Legal Instruments Examiner
Central Reexamination Unit 571-272-7705; FAX No. 571-273-9900

Patent Assignment Abstract of Title

Total Assignments: 2

Application #: 17142983 **Filing Dt:** 01/06/2021 **Patent #:** 11695823 **Issue Dt:** 07/04/2023
PCT #: NONE **Intl Reg #:** **Publication #:** NONE **Pub Dt:**
Inventors: Pouya Taaghol, Vivek Ramanna
Title: Distributed Software Defined Networking

Assignment: 1

Reel/Frame: 063290 / 0800 **Received:** 04/11/2023 **Recorded:** 04/11/2023 **Mailed:** 04/12/2023 **Pages:** 4
Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).
Assignor: BIG DATA FEDERATION, INC. **Exec Dt:** 03/16/2023
Assignee: TAAGHOL, POUYA
2143 MENALTO AVENUE
MENLO PARK, CALIFORNIA 94025
Correspondent: ROARK IP
1438 DAHLIA LOOP
SAN JOSE, CA 95126

Assignment: 2

Reel/Frame: 063454 / 0352 **Received:** 04/26/2023 **Recorded:** 04/26/2023 **Mailed:** 04/28/2023 **Pages:** 4
Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).
Assignor: TAAGHOL, POUYA **Exec Dt:** 04/20/2023
Assignee: EDGE NETWORKING SYSTEMS, LLC
5900 BALCONES DRIVE
STE 100
AUSTIN, TEXAS 78731
Correspondent: ROARK IP
1438 DAHLIA LOOP
SAN JOSE, CA 95126

Search Results as of: 05/08/2025 06:37 AM

If you have any comments or questions concerning the data displayed, contact PRD / Assignments at 571-272-3350. v.2.6
Web interface last modified: Jun 26, 2017 v.2.6

Patent Assignment Abstract of Title

Total Assignments: 2

Application #: 17142983 **Filing Dt:** 01/06/2021 **Patent #:** 11695823 **Issue Dt:** 07/04/2023
PCT #: NONE **Intl Reg #:** **Publication #:** NONE **Pub Dt:**
Inventors: Pouya Taaghol, Vivek Ramanna
Title: Distributed Software Defined Networking

Assignment: 1

Reel/Frame: 063290 / 0800 **Received:** 04/11/2023 **Recorded:** 04/11/2023 **Mailed:** 04/12/2023 **Pages:** 4
Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).
Assignor: BIG DATA FEDERATION, INC. **Exec Dt:** 03/16/2023
Assignee: TAAGHOL, POUYA
2143 MENALTO AVENUE
MENLO PARK, CALIFORNIA 94025
Correspondent: ROARK IP
1438 DAHLIA LOOP
SAN JOSE, CA 95126

Assignment: 2

Reel/Frame: 063454 / 0352 **Received:** 04/26/2023 **Recorded:** 04/26/2023 **Mailed:** 04/28/2023 **Pages:** 4
Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).
Assignor: TAAGHOL, POUYA **Exec Dt:** 04/20/2023
Assignee: EDGE NETWORKING SYSTEMS, LLC
5900 BALCONES DRIVE
STE 100
AUSTIN, TEXAS 78731
Correspondent: ROARK IP
1438 DAHLIA LOOP
SAN JOSE, CA 95126

Search Results as of: 05/08/2025 06:37 AM

If you have any comments or questions concerning the data displayed, contact PRD / Assignments at 571-272-3350. v.2.6
Web interface last modified: Jun 26, 2017 v.2.6

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 2943

SERIAL NUMBER	FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
90/019,948	05/06/2025 RULE	709	3992	UNP-16002

APPLICANTS

Edge Networking Systems, LLC, Residence Not Provided;
Unified Patents LLC, Chevy Chase, MD;

INVENTORS

11695823, Residence Not Provided;
Amundsen Davis, LLC, Residence Not Provided;
Pouya Taaghol, San Jose, CA;
Vivek Ramanna, San Jose, CA;

**** CONTINUING DATA ***************************

This application is a REX of 17/142,983 01/06/2021 PAT 11695823
which is a CON of 16/900,963 06/14/2020 PAT 10893095
which is a CON of 15/836,824 12/09/2017 PAT 10686871
which is a CON of 14/295,331 06/04/2014 PAT 9843624
which claims benefit of 61/834,807 06/13/2013

**** FOREIGN APPLICATIONS ***************************

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****

Foreign Priority claimed ❑ Yes ❑ No 35 USC 119(a-d) conditions met ❑ Yes ❑ No Verified and Acknowledged _____ Examiner's Signature	❑ Met after Allowance _____ Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
				19	2

ADDRESS

Roark IP
1438 Dahlia Loop
San Jose, CA 95126
UNITED STATES

TITLE

Distributed Software Defined Networking

FILING FEE RECEIVED		
0.00	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	❑ All Fees ❑ 1.16 Fees (Filing) ❑ 1.17 Fees (Processing Ext. of time) ❑ 1.18 Fees (Issue) ❑ Other _____ ❑ Credit

BIB (Rev. 05/07).

PTO/SB/57 (11-23)
Approved for use through 11/30/2027. OMB 0651-0064
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

(Also referred to as FORM PTO-1465)

REQUEST FOR *EX PARTE* REEXAMINATION TRANSMITTAL FORM

Address to:
Mail Stop *Ex Parte* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attorney Docket No.: UNP-16002

Date: May 6, 2025

1. ■ This is a request for *ex parte* reexamination pursuant to 37 CFR 1.510 of patent number 11695823 issued July 4, 2023. The request is made by:
   ☐ patent owner. ■ third party requester.

2. ■ The name and address of the person requesting reexamination is:
   Unified Patents, LLC
   4445 Willard Avenue, Suite 600
   Chevy Case, MD 20815

3. Requester ☐ asserts small entity status (37 CFR 1.27) or ☐ certifies micro entity status (37 CFR 1.29). Only a patent owner requester can certify micro entity status. Form PTO/SB/15A or B must be attached to certify micro entity status.

4. This request is accompanied by payment of the reexamination fee as set forth in:
   ■ 37 CFR 1.20(c)(2); **or**
   ☐ 37 CFR 1.20(c)(1). **In checking this box for payment of the fee set forth in 37 CFR 1.20(c)(1), requester asserts that this request has forty (40) or fewer pages and complies with all other requirements of 37 CFR 1.20(c)(1).**

   Payment of the reexamination fee is made by the method set forth below.
   a. ☐ A check in the amount of $_____ is enclosed to cover the reexamination fee;
   b. ■ The Director is hereby authorized to charge the reexamination fee to Deposit Account No. 506990;
   c. ☐ Payment by credit card. Form PTO-2038 is attached; **or**
   d. ☐ Payment made via USPTO patent electronic filing system.

   ■ In addition, the Director is hereby authorized to charge any fee deficiencies to Deposit Account No. 506990.

5. ■ Any refund should be made by ☐ check or ■ credit to Deposit Account No. 506990. 37 CFR 1.26(c). If payment is made by credit card, refund must be to credit card account.

6. ■ A copy of the patent to be reexamined having a double column format on one side of a separate paper is enclosed. 37 CFR 1.510(b)(4).

7. ☐ CD-ROM or CD-R in duplicate, Computer Program (Appendix) or large table
   ☐ Landscape Table on CD

A Federal agency may not conduct or sponsor, and a person is not required to respond to, nor shall a person be subject to a penalty for failure to comply with an information collection subject to the requirements of the Paperwork Reduction Act of 1995, unless the information collection has a currently valid OMB Control Number. The OMB Control Number for this information collection is 0651-0064. Public burden for this form is estimated to average 55 hours per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the information collection. Send comments regarding this burden estimate or any other aspect of this information collection, including suggestions for reducing this burden to the Chief Administrative Officer, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA 22313-1450 or email InformationCollection@uspto.gov. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **If filing this completed form by mail, send to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

8. ☐ Nucleotide and/or Amino Acid Sequence Submission
If applicable, items a. – c. are required.
   a. ☐ Computer Readable Form (CRF)
   b. Specification Sequence Listing on:
      i. ☐ CD-ROM (2 copies) or CD-R (2 copies) **or**
      ii. ☐ paper
   c. ☐ Statements verifying identity of above copies.

9. ☐ A copy of any disclaimer, certificate of correction or reexamination certificate issued in the patent is included.

10. ■ Reexamination of claim(s) 1, 2, 3, 4, 8, 15 and 19 is requested.

11. ■ A copy of every patent or printed publication relied upon is submitted herewith including a listing thereof on Form PTO/SB/08, PTO-1449, or equivalent.

12. ☐ An English language translation of all necessary and pertinent non-English language patents and/or printed publications is attached.

13. ■ The attached detailed request includes at least the following items:
   a. A statement identifying each substantial new question of patentability based on prior patents and printed publications. 37 CFR 1.510(b)(1).
   b. An identification of every claim for which reexamination is requested, and a detailed explanation of the pertinency and manner of applying the cited art to every claim for which reexamination is requested. 37 CFR1.510(b)(2).

14. ☐ A proposed amendment is included (only where the patent owner is the requester). 37 CFR 1.510(e).

15. ■ It is certified that the statutory estoppel provisions of 35 U.S.C. 315(e)(1) or 35 U.S.C. 325(e)(1) do not prohibit requester from filing this *ex parte* reexamination request. 37 CFR 1.510(b)(6).

16. Service
   a. ■ It is certified that a copy of this request (if filed by other than the patent owner) has been served in its entirety on the patent owner as provided in 37 CFR 1.33(c).

      The name and address of the party served are:

      Roark IP, Attention: Christopher Horgan
      1438 Dahlia Loop
      San Jose, CA 95126

      Date of Service: May 6, 2025

   OR

   b. ☐ A duplicate copy is enclosed since service on patent owner was not possible. An explanation of the efforts made to serve patent owner **is attached**. See MPEP 2220.

17. Correspondence Address: Direct all communication about the reexamination to:

☒ The address associated with Customer Number: 66155

OR

☐ Firm or Individual Name _____
(at the address identified below)

Address

City	State	Zip

Country	

Telephone	Email

18. ☒ The patent is currently the subject of the following concurrent proceeding(s):

a. ☐ Copending reissue Application No. _____

b. ☐ Copending reexamination Control No. _____

c. ☐ Copending Interference No. _____

d. ☒ Copending litigation styled:

Edge Networking Systems, LLC v. Microsoft Corporation, Case No. 1:24cv215 (W.D. Tex. Feb. 29, 2024)

Edge Networking Systems, LLC v. Amazon.com, Inc., et al, Case No: 2:24cv887 (E.D. Tex. Nov. 1, 2024)

IPR 2025-00618 filed by Microsoft Corporation on February 17, 2025.

WARNING : Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

/John T. Pienkos/
Authorized Signature

May 6, 2025
Date

John T. Pienkos
Typed/Printed Name

42,997
Registration No.

☐ For Patent Owner Requester ☒ For Third Party Requester

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. The United States Patent and Trademark Office (USPTO) collects the information in this record under authority of 35 U.S.C. 2. The USPTO's system of records is used to manage all applicant and owner information including name, citizenship, residence, post office address, and other information with respect to inventors and their legal representatives pertaining to the applicant's/owner's activities in connection with the invention for which a patent is sought or has been granted. The applicable Privacy Act System of Records Notice for the information collected in this form is COMMERCE/PAT-TM-7 Patent Application Files, available in the Federal Register at 78 FR 19243 (March 29, 2013), https://www.govinfo.gov/content/pkg/FR-2013-03-29/pdf/2013-07341.pdf.

Routine uses of the information in this record may include disclosure to: 1) law enforcement, in the event that the system of records indicates a violation or potential violation of law; 2) a federal, state, local, or international agency, in response to its request; 3) a contractor of the USPTO having need for the information in order to perform a contract; 4) the Department of Justice for determination of whether the Freedom of Information Act (FOIA) requires disclosure of the record; 5) a Member of Congress submitting a request involving an individual to whom the record pertains, when the individual has requested the Member's assistance with respect to the subject matter of the record; 6) a court, magistrate, or administrative tribunal, in the course of presenting evidence, including disclosures to opposing counsel in the course of settlement negotiations; 7) the Administrator, General Services Administration (GSA), or their designee, during an inspection of records conducted by GSA under authority of 44 U.S.C. 2904 and 2906, in accordance with the GSA regulations and any other relevant (i.e., GSA or Commerce) directive, where such disclosure shall not be used to make determinations about individuals; 8) another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)); 9) the Office of Personnel Management (OPM) for personnel research purposes; and 10) the Office of Management and Budget (OMB) for legislative coordination and clearance.

If you do not furnish the information requested on this form, the USPTO may not be able to process and/or examine your submission, which may result in termination of proceedings, abandonment of the application, and/or expiration of the patent.

Appendix B

Claim Chart for Unpatentability of the ‘823 Patent (Malik)

Claim Chart for U.S. Patent No. 11695823 ('823 Patent)

I. Grounds of Unpatentability

	Claims of '823 Patent	Statute(s)	Prior Art
3.	Claims 1, 2, 3, 4, 15 and 19	35 U.S.C. §102	Malik
4.	Claims 1, 2, 3, 4, 15 and 19	35 U.S.C. §103	Malik

II. Prior Art Relied Upon

Reference
Ex. 1005 U.S. Pub. No. 20130054682 to Malik *et al*. (**Malik**) was filed as U.S. Patent Application No. 13/598256 on August 29, 2012, published on February 28, 2013, and issued on May 19, 2015 as U.S. Patent No. 9034642. Malik is prior art to the '823 Patent at least pursuant to 35 U.S.C. §102(a)(1) and (2).

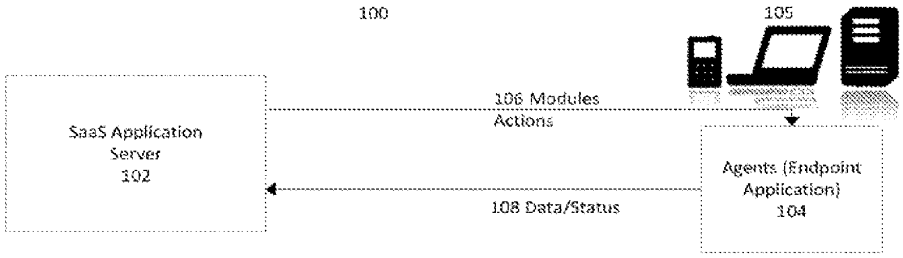
III. Claim Chart

Claims of '823 Patent	Prior Art Disclosure
Claim 1 [P]	
A system comprising:	To the extent that the preamble is limiting, Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation.[1] Malik discloses *a system*.[2] Malik (Ex. 1005) at Abstract and ¶ [0002]. ***See, e.g.*, Malik** **Systems**[3] and methods facilitate the deployment of software modules from a file server to agents on endpoints. Agents can contact a heartbeat server to receive information about a latest manifest file that identifies the software modules that should be installed on the endpoint. Agents can then determine from the manifest information whether new files need to be downloaded and automatically download them. A scripting engine can be used to allow run-time installation of modules without causing significant downtime of the endpoint. The scripting engine can utilize the agent to report back status and data from the software modules to allow modules to be used with a mobile device management platform. Malik (Ex. 1005) at Abstract

[1] The discussion in this Claim Chart is equally applicable to both the anticipation and obviousness grounds for rejection unless otherwise noted.
[2] Language in italics throughout this document denotes claim language of the claims of the '823 Patent.
[3] Unless otherwise noted, all bold used in quotes of the disclosures of the prior art is emphasis added to highlight relevant portions.

	The present invention relates generally to methods and **systems** for providing mobile device management services and more specifically providing deployment services for deploying software modules to a plurality of managed endpoints. Malik (Ex. 1005) at ¶[0002]
Claim 1 [a]	
a programmable network device adapted to host a plurality of network device applications;	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *a programmable network device* (e.g., endpoint) *adapted to host a plurality of network device applications* (e.g., feature modules installed on the endpoint device). Specifically, Malik discloses a system that includes "deployment of software modules from a file server to agents on endpoints." Malik (Ex. 1005) at Abstract. The endpoints are programmable at least in that the agents receive, install and execute feature modules. *Id*. at Abstract, ¶[0057] and Fig. 1. The feature modules can each be an "executable application, script, or data pack that may be selectively deployed to one or many endpoints, providing a specific function in addition to the core functionality of the MDM system", and may be for any purpose. *Id*. at ¶[0032]; *see also Id*. at ¶¶[0030] and [0036]. Feature modules may themselves be applications, or may be modules added to existing applications. Malik (Ex. 1005) at Abstract and ¶[0033]. ***See, e.g.*, Malik** Systems and methods facilitate the **deployment of software modules from a file server to agents on endpoints**. Agents can contact a heartbeat server to receive information about a latest manifest file that identifies the software modules that should be installed on the endpoint. Agents can then determine from the manifest information whether new files need to be downloaded and automatically download them. A scripting engine can be used to allow run-time installation of modules without causing significant downtime of the endpoint. The scripting engine can utilize the agent to report back status and data from the software modules to allow modules to be used with a mobile device management platform Malik (Ex. 1005) at Abstract In some embodiments of the present invention, deployment of applications and data to endpoints can be facilitated by a network-based solution. In some embodiments, a cloud-based approach is used. **Agents can be installed on endpoint devices. These agents can facilitate installation of applications/feature modules on-demand.** This can allow an organization to selectively add features and applications to endpoint devices without a complicated manual rollout. This can allow an organization select features for use with a mobile device management (MDM) platform to target the needs of

	an organization, and allow features to be added or removed as the needs of the organization change. When an MDM system is outsourced to a vendor, or is supplied by vendor, a simplified deployment and distribution approach to applications and features may allow the supplier or operator of the MDM system to offer flexible pricing that is tailored to the features used and the needs of an organization. Malik (Ex. 1005) at ¶[0030] **As used herein, a feature module is an executable application, script, or data pack that may be selectively deployed to one or many endpoints, providing a specific function in addition to the core functionality of the MDM system. A feature module may include the executable code and/or data necessary to implement a feature of an MDM platform.** These can include new standalone applications or scripts, or may include functionality added to existing applications on an endpoint device or updates to these existing applications. Examples of feature modules include patch management for laptops, Exchange ActiveSync managers, Lotus Traveler managers, user visibility applications, such as applications utilizing Active Directory from Microsoft, etc. Other examples of feature modules include those disclosed throughout this application. Malik (Ex. 1005) at ¶[0032] In some embodiments, feature modules may be supplied by the operator or vendor of the MDM platform. In some embodiments, feature modules may also be supplied by third party vendors or the user of the MDM platform itself. **Feature modules may be applications, or modules added to existing MDM applications.** In some embodiments, feature modules are additional applications that can be installed on an endpoint. Feature modules may communicate with components on a network, such as communicating with a server component to receive instructions from the MDM platform or to send information, such as status information about the endpoint device to the server, so that the MDM platform can analyze this information from the endpoint. In some embodiments, feature modules may also use the existing reporting and control channels provided by the MDM system and agents on the endpoint to facilitate visibility and management features provided by the feature module. Malik (Ex. 1005) at ¶[0033] In some embodiments, **feature modules can be used for any purpose**. In some embodiments, these purposes include allowing security and management policies to be enforced on an endpoint device. In some embodiments, these purposes may include providing visibility information about devices to the MDM platform. In some embodiments, this can allow IT managers to gather and analyze information about the health of endpoint devices connecting to a corporate network and the overall health of the network itself. In some embodiments, feature modules may actively participate in controlling or providing access for endpoint devices to corporate resources, such as e-mail,

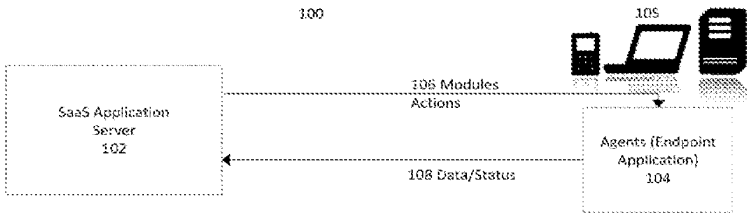
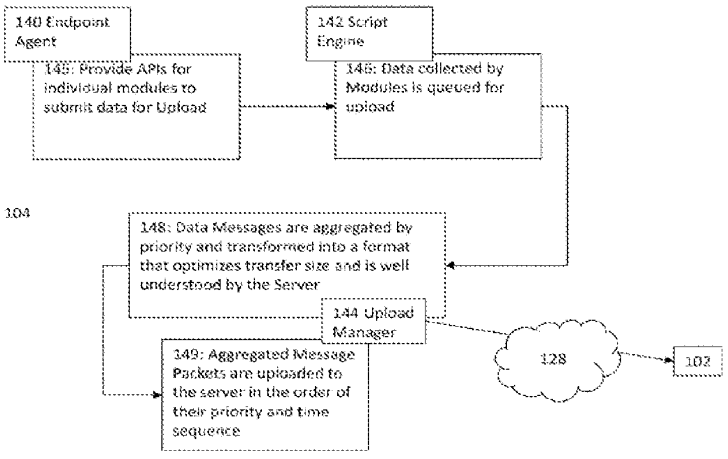
	documents, data, or the like. In some embodiments, feature modules may provide features unrelated to MDM, such as standalone applications that provide non-MDM features that are deployed to endpoints in an organization using the distribution and deployment functionality of the MDM platform. Malik (Ex. 1005) at ¶[0036] FIG. 1 shows a high level architecture of the deployment and distribution system as part of an MDM platform. System 100 contains server components and endpoint components. An application server 102 provides server functionality as described herein. This application server can be a SaaS application server providing server functionality in a cloud-based approach, which include a plurality of servers hosting the application or components thereof. **Agents 104 on endpoints 105 provide core MDM functionality including allowing actions to be executed on the endpoints and allowing data and status information to be returned to the server. Agents 104 can include a plurality of endpoint applications that provide agent functionality.** SaaS application server 102 may transmit actions and modules across a link 106, which may traverse a network, such as the Internet. **Agents 104 may receive these modules and actions and install modules or execute actions accordingly.** Agents 104 can also transmit data to the application server across link 108 which may also traverse a network, such as the Internet. This allows agents to report back status of deployments and data collected or generated by the deployed feature modules to application server 102. Malik (Ex. 1005) at ¶[0057] Malik (Ex. 1005) at Fig. 1:  FIG. 1
Claim 1 [b]	
a programmable cloud device adapted to host a plurality of cloud applications	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *a programmable cloud device* (e.g., server) *adapted to host a plurality of cloud applications* (e.g., feature modules hosted by the server).

	Specifically, Malik discloses a server, or plurality of servers, that are implemented in a cloud based approach. Malik (Ex. 1005) at ¶¶ [0038] and [0048]. The server includes a "file server configured to host a plurality of feature modules." *Id*. at ¶[0008]; *See also Id*. at ¶¶ [0012], [0057], [0087], and [0092]. ***See, e.g.*, Malik** One embodiment is directed to a method for deploying software modules using **device management platform, including providing, at one or more server computers, a file server configured to host a plurality of feature modules**, a module manager configured to maintain the plurality of feature modules, and a manifest server configured to maintain a plurality of manifests. Each manifest conveys the feature module components that at least one endpoint should have installed. The method further includes providing one or more agents to a plurality of endpoints for operating thereon, where the agents are configured to communicate with the one or more server computers to receive at least one manifest and subsequently receive a subset of the plurality of feature modules from the file server. An execution engine configured to execute a plurality of scripts contained in the feature modules is provided to endpoint computers for operating thereon. The agents are configured to facilitate reporting of data from the feature modules to the one or more servers. Malik (Ex. 1005) at ¶[0008] Another embodiment is directed to **a system for providing modules to a plurality of endpoints, having software instructions stored on non-transitory computer memory on at least one server, wherein the software instructions include a file server configured to host a plurality of feature modules, a module manager configured to maintain the plurality of feature modules, and a manifest server configured to maintain a plurality of manifests, wherein each manifest conveys the feature module components that at least one endpoint should have installed**. The software instructions further include a heartbeat server configured to receive requests from a plurality of agents, wherein each agent operates on an endpoint, and to convey information about at least one latest manifest to each agent when that agent has not yet downloaded the latest manifest and an upload server configured to receive data from modules installed on each endpoint, wherein the modules include those modules identified in the latest manifest. Malik (Ex. 1005) at ¶[0012] **While the exemplary MDM systems are described with respect to servers, it should be appreciated that the system can be implemented on a plurality of servers, including cloud-based systems which may deploy the system on a plurality of servers on a dynamic basis, allowing for greater scalability, or a single server that includes many or all of the features disclosed herein**. In some embodiments, the servers described herein can be software modules executing on a common or shared server platform. Any suitable portion of the

	exemplary MDM systems can be executed on one or more processors, which may be separate from the processors executing other portions of the MDM system. A system may be implemented as part of the local IT infrastructure of a corporate environment or as part of a third party solution employed by the organization. Malik (Ex. 1005) at ¶[0038] Agents may also communicate with more than one server. **In some embodiments, a cloud-based approach is used, whereby agents can communicate with any number of servers on the Internet to deploy and manage feature modules.** By using a cloud-based approach, an MDM system and the deployment and distribution system can be scaled to meet demand. For example, where a new update that broadly applies to a great number of endpoints is needed, a cloud-based content delivery system may be used to scale up to meet the temporary demand for the large number of downloads that are required to deploy the feature modules to all endpoints. Malik (Ex. 1005) at ¶[0048] FIG. 1 shows a high level architecture of the deployment and distribution system as part of an MDM platform. System 100 contains server components and endpoint components. An application server 102 provides server functionality as described herein. **This application server can be a SaaS application server providing server functionality in a cloud-based approach, which include a plurality of servers hosting the application or components thereof.** Agents 104 on endpoints 105 provide core MDM functionality including allowing actions to be executed on the endpoints and allowing data and status information to be returned to the server. Agents 104 can include a plurality of endpoint applications that provide agent functionality. SaaS application server 102 may transmit actions and modules across a link 106, which may traverse a network, such as the Internet. Agents 104 may receive these modules and actions and install modules or execute actions accordingly. Agents 104 can also transmit data to the application server across link 108 which may also traverse a network, such as the Internet. This allows agents to report back status of deployments and data collected or generated by the deployed feature modules to application server 102. Malik (Ex. 1005) at ¶[0057] This allows server components of feature modules to correlate collected data with the endpoints and more easily share data with other applications or the MDM platform. **This can allow reporting of data from endpoints to be used for various purposes on the server by applications, policies, or for visibility provided to a dashboard or for other viewing by an administrator using the MDM platform.** Malik (Ex. 1005) at ¶[0087]

	Message handlers 155 or message processors 154 can also store the data from the messages in a shared data store 156 (which may store the uploaded data in native or processed form). When stored, the information can be associated with the individual endpoints that sent the data, allowing the data pertaining to a single endpoint to be gathered from a group of feature modules (which may be unrelated or related) and shared and accessed by an administrator or MDM application. This can allow feature modules to add features to existing MDM applications. **The shared data store can allow a plurality of applications on the server side to access and mine the data returned.** This information be accessed by user interface 159, which can include a dashboard for viewing attributes and data pertaining to groups or individual devices. This interface may be presented to a user via a webpage, allowing an administrator to application server 102 via a browser and manage endpoints and view information about them that is received from feature modules. Malik (Ex. 1005) at ¶[0092]
Claim 1 [c]	
wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications; and	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *wherein the plurality of network device applications* (e.g., feature modules installed on the endpoint device) *and the plurality of cloud applications* (e.g., modules hosted by the server) *are in secure communication with each other to form distributed applications* (e.g., system deployed on a plurality of servers on a dynamic basis). Specifically, Malik discloses agents on the endpoint devices "for downloading updates in a secure manner from any suitable server resources." Malik (Ex. 1005) at ¶[0047]. Malik also discloses that its MDM system can be implemented with a plurality of servers hosting the applications or components thereof, resulting in distributed applications, to provide scalability. *Id*. at ¶¶[0038], [0057], and [0062]. Malik further discloses that software components that reside on and endpoint gather data that is then uploaded to the server components, thus disclosing applications distributed between the endpoint and the server. *Id*. at ¶¶[0062] and [0087]. ***See, e.g.*, Malik** **While the exemplary MDM systems are described with respect to servers, it should be appreciated that the system can be implemented on a plurality of servers, including cloud-based systems which may deploy the system on a plurality of servers on a dynamic basis, allowing for greater scalability, or a single server that includes many or all of the features disclosed herein.** In some embodiments, the servers described herein can be software modules executing on a common or shared server platform. Any suitable portion of the exemplary MDM systems can be executed on one or more processors, which may be separate from the processors executing other portions of the MDM

	system. A system may be implemented as part of the local IT infrastructure of a corporate environment or as part of a third party solution employed by the organization. Malik (Ex. 1005) at ¶[0038] **Agents can be provided for downloading updates in a secure manner from any suitable server resources. For example, communication between agents on endpoints and servers can utilize encryption, such as by utilizing secure socket layer (SSL) or HTTPS communication. The agent can also authenticate any server involved in the delivery of feature modules.** This can prevent attacks whereby untrusted servers provide malicious or unsecure feature modules that may leave an endpoint vulnerable. In some embodiments, encryption and authentication can also provide security to server resources by authenticating agents prior to allowing the agents access to modules. This may allow a vendor of an MDM system to prevent unauthorized access to features to require payment for these features. By providing an encrypted and authenticated system, agents and servers can operate confidently that the endpoints and servers are who they purport to be and are secure. Malik (Ex. 1005) at ¶[0047] **FIG. 1 shows a high level architecture of the deployment and distribution system as part of an MDM platform. System 100 contains server components and endpoint components. An application server 102 provides server functionality as described herein. This application server can be a SaaS application server providing server functionality in a cloud-based approach, which include a plurality of servers hosting the application or components thereof. Agents 104 on endpoints 105 provide core MDM functionality including allowing actions to be executed on the endpoints and allowing data and status information to be returned to the server.** Agents 104 can include a plurality of endpoint applications that provide agent functionality. SaaS application server 102 may transmit actions and modules across a link 106, which may traverse a network, such as the Internet. Agents 104 may receive these modules and actions and install modules or execute actions accordingly. Agents 104 can also transmit data to the application server across link 108 which may also traverse a network, such as the Internet. This allows agents to report back status of deployments and data collected or generated by the deployed feature modules to application server 102. Malik (Ex. 1005) at ¶[0057] Manifest server 114 manages pushing updates to endpoints by creating manifests that associate modules to be distributed to the endpoints with the groups of endpoints to receive the modules. Manifests can be small files that can be easily distributed to endpoints. Malik (Ex. 1005) at ¶[0062]

	The scripting engine 142 passes this collected data to an upload manager. Each of these is a **software component that resides on the endpoint and may be part of the agent.** Upload manager 144 is responsible for aggregating, normalizing, formatting, or other processing of data prior to upload. **Upload manager 144 then forms aggregated message packets that contain information gathered by a plurality of modules to be passed to the server across a network 128, such as the Internet, to server components on server 102**. These packets can include one or more identifiers that associate the collected data with the agent or endpoint. **This allows server components of feature modules to correlate collected data with the endpoints and more easily share data with other applications or the MDM platform.** This can allow reporting of data from endpoints to be used for various purposes on the server by applications, policies, or for visibility provided to a dashboard or for other viewing by an administrator using the MDM platform. Malik (Ex. 1005) at ¶[0087] Malik (Ex. 1005) at Fig. 1:  FIG. 1 Malik (Ex. 1005) at Fig. 4:  FIG. 4

Claim 1 [d]	
wherein the plurality of network device applications and plurality of cloud applications device [*sic*] form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware.	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *wherein the plurality of network device applications* (e.g., feature modules installed on the endpoint device) *and plurality of cloud applications device* [*sic*[4]] (e.g., modules hosted by the server) *form unified capabilities* (e.g., programs on the endpoints and programs on the cloud server work together in a unified manner to perform the full functionality of the applications) *enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device* (e.g., endpoint) *hardware and cloud device* (e.g., server) *hardware*. Requester notes that this limitation recites "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware." The recitation of such "functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the prior art". MPEP §2114; *See also* Request at Section II(D). However, to the extent that such use of APIs is given patentable weight, Malik discloses enabling the use of APIs in its systems. Regarding that the applications on the network device and cloud device *form unified capabilities*, Malik explicitly discloses both applications/feature modules being deployed on endpoints and also an application server hosting applications and providing server functionality in a cloud-based approach. Malik (Ex. 1005) at ¶¶[0030], [0057], [0087], and [0092]. Malik further explicitly discloses that these are distributed applications, including a "server providing server functionality in a cloud-based approach, which include a plurality of servers hosting the application or components thereof" as well as "[a]gents 104 on endpoints 105" that "provide core MDM functionality including allowing actions to be executed on the endpoints and allowing data and status information to be returned to the server." Malik (Ex. 1005) at ¶[0057]; *see also Id.* at ¶¶[0038], [0062] and [0087]. Indeed, Malik discloses both applications/feature modules being deployed on endpoints and also an application server hosting applications and providing server functionality in a cloud-based approach. *Id.* at ¶¶[0030], [0057], [0087], and [0092]. Malik discloses that the use of the feature modules on the server and the endpoints operate in an unified manner in that the feature modules can allow an organization to "selectively add features and applications to endpoint devices without a complicated manual rollout," which may avoid problems that might be caused by changing components that previously worked properly, and which

[4] Requester notes that Claim 1[d] includes the word "device" after the first recitation of "cloud applications," which appears to be a typographical error.

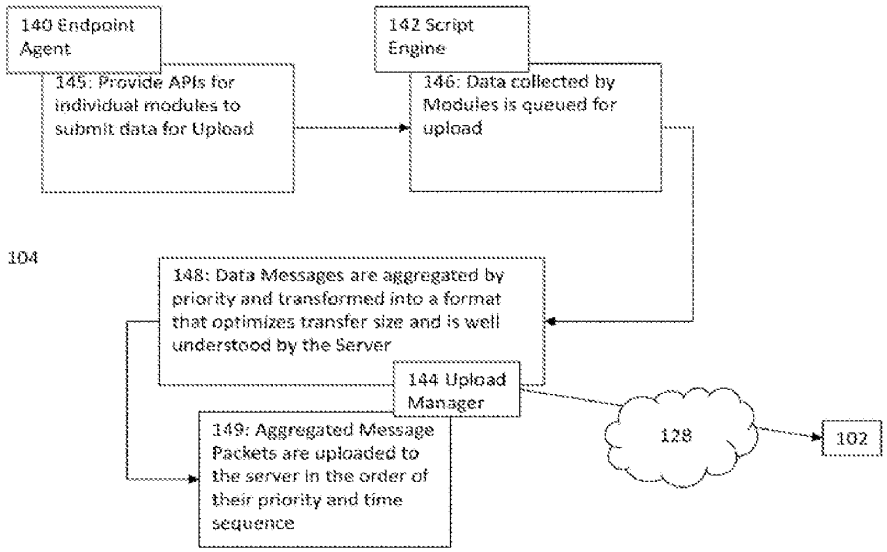
	may be accomplished without "system down time, advance notice, restarting of devices on the network." Malik (Ex. 1005) at ¶¶[0030], [0043]-[0045] and [0084]. Also, Malik discloses that the use of the feature modules on the server and the endpoints operate in an unified manner in that the feature modules "allow reporting of data from endpoints to be used for various purposes on the server by applications, policies, or for visibility provided to a dashboard or for other viewing by an administrator using the MDM platform." Malik (Ex. 1005) at ¶¶[0057], [0087], and [0092]. Further, as a POSITA would understand, because the applications are distributed applications, the applications on the cloud server and the applications on the gateway device necessarily form unified capabilities, because that is the nature and purpose of distributed applications. Geier Decl. at ¶130. Moreover, it is clear from the teachings of Malik that the distributed applications of Malik work together in a unified manner to perform the full functionality of the application, thus forming unified capabilities. Geier Decl. at ¶130. Regarding the element of "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware", Malik also discloses that developers of feature modules can take advantage of APIs as part of Malik's MDM platform, to allow reporting of information gathered by feature modules. Malik (Ex. 1005) at ¶¶ [0084], [0088], and Fig. 4. Malik further discloses use of a web based dashboard and also that developers can define and configure modules for deployment by use of a dashboard. Malik (Ex. 1005) at ¶¶[0094] and [0099]. While Malik does not explicitly state that the applications are programmed by *upper layer* APIs *independent of network device hardware and cloud device hardware,* Malik does teach the use of APIs as discussed above. As would be understood and appreciated by a POSITA, it is the function and purpose of APIs to allow for programing in a manner that is independent of device hardware. Geier Decl. at ¶ 133. It is therefore an inherent characteristic of an API that it can be employed to program applications in a manner that is removed from or "independent of" (and in that sense, in an upper layer with respect to) "hardware." Geier Decl. (Ex. 1003) at ¶ 133. If a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation. MPEP §2114; *See also* Request at Section II(D). For these reasons, Malik either explicitly or inherently discloses *enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware.*

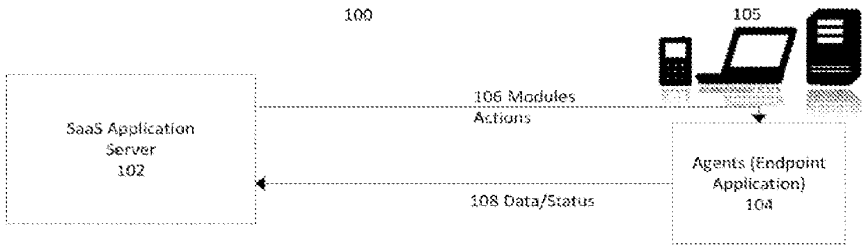
	(**Ground 2, Obviousness**): To the extent that Malik does not anticipate this limitation, it is nonetheless rendered obvious in view of the knowledge of a POSITA as of June 13, 2023 for the reasons detailed in Section IV(E) of the Request. *See also* Geier Decl. at ¶¶ 153-158. As discussed, the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware) was well within the knowledge of a POSITA prior to the June 13, 2023 priority date of the '823 Patent. *Id*. at ¶158. Accordingly, a POSITA would have known about and understood the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware). *Id*. at ¶ 158. Additionally, a POSITA would have understood that the APIs of Malik are "upper layer" APIs within the meaning of the '823 Patent, and that they would be enabled "to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" as recited in independent claims 1 and 19 of the '823 Patent. *Id*. at ¶158. ***See, e.g.*, Malik** In some embodiments of the present invention, deployment of applications and data to endpoints can be facilitated by a network-based solution. In some embodiments, a cloud-based approach is used. Agents can be installed on endpoint devices. These agents can facilitate installation of applications/feature modules on-demand. **This can allow an organization to selectively add features and applications to endpoint devices without a complicated manual rollout.** This can allow an organization select features for use with a mobile device management (MDM) platform to target the needs of an organization, and allow features to be added or removed as the needs of the organization change. When an MDM system is outsourced to a vendor, or is supplied by vendor, a simplified deployment and distribution approach to applications and features may allow the supplier or operator of the MDM system to offer flexible pricing that is tailored to the features used and the needs of an organization. Malik (Ex. 1005) at ¶[0030] **By utilizing a modular approach, in some embodiments, the core software for providing MDM services to the endpoint can remain operational and unaffected by the deployed updates or modules**. For example, deploying a module that updates features related to e-mail, may not require changes to other MDM features, such as reporting status information or maintaining security and compliance with policies on the endpoint. In some embodiments, modules may provide new features to supplement the operation of core MDM software on the endpoint. **Utilizing a modular approach to features may allow an organization to deploy a core MDM component to all devices and then**

	selectively add features to the device, using the deployment system described herein. Furthermore, this may allow an organization to begin with a baseline of MDM features and upgrade the features that the organization uses to meet its needs, without replacing the core MDM software. This can avoid problems that might be caused by changing components that previously worked properly. Malik (Ex. 1005) at ¶[0043] In some embodiments, the feature module may be one or more scripts that can be interpreted by software running on the endpoint. This software may be part of the agent or may be a software module associated or in communication with the agent. This may allow the agent to download new or updated feature modules, while a scripting engine interprets these new modules. In some embodiments, **feature modules may be executed/interpreted or updated without requiring the entire endpoint device or all of the MDM components on that device to be restarted**. This can provide several advantages. Malik (Ex. 1005) at ¶[0044] **First, in embodiments where the device need not be restarted, the end-user of the device may be unaware or not generally inconvenienced by updates. This can allow updates and new modules to be deployed to endpoint without the typical logistic headaches associated with traditional rollouts of software. For example, traditional rollouts may have required system down time, advance notice, restarting of devices on the network, and substantial testing before a rollout. In some embodiments, an update or new module can be deployed to endpoint devices at any time, without requiring system downtime, advance warning to users, or input from users.** Second, in some embodiments where MDM components on the endpoint device need not be restarted, MDM components, such as agents or other feature modules, can continue to operate, allowing endpoint agents and management and security policies to continue to function normally during the update process. This can avoid security risks associated with disabling these MDM components. Furthermore, where updates are unobtrusive to the end-user, and where updates can frequently occur without greatly diminishing the user experience, the MDM system may provide advantages to developers of feature modules. For example, substantial beta testing of updates may not be required before deploying new or updated feature modules. In addition, an update can be provided to a group of endpoints and the group of endpoints can act as a real-world beta testing group for these updates. This may greatly reduce the amount of development time required for producing reliable, new or updated feature modules. Malik (Ex. 1005) at ¶[0045] FIG. 1 shows a high level architecture of the deployment and distribution system as part of an MDM platform. System 100 contains server components and endpoint components. An application server 102 provides server functionality as described herein. **This application server can be a SaaS application server**

	providing server functionality in a cloud-based approach, which include a plurality of servers hosting the application or components thereof. Agents 104 on endpoints 105 provide core MDM functionality including allowing actions to be executed on the endpoints and allowing data and status information to be returned to the server. Agents 104 can include a plurality of endpoint applications that provide agent functionality. SaaS application server 102 may transmit actions and modules across a link 106, which may traverse a network, such as the Internet. Agents 104 may receive these modules and actions and install modules or execute actions accordingly. Agents 104 can also transmit data to the application server across link 108 which may also traverse a network, such as the Internet. This allows agents to report back status of deployments and data collected or generated by the deployed feature modules to application server 102. Malik (Ex. 1005) at ¶[0057] FIG. 4 depicts the end point components of a reporting system that can be used as part of the module distribution and deployment system. **Reporting component can allow deployed modules to report back status related to their installation as well as report back information to server components during normal execution. This can allow developers of modules to take advantage of APIs as part of the MDM platform that allow reporting of information gathered by modules**. For example, modules may be developed that determine status information about an endpoint, apply compliance policies to an endpoint, apply configuration to an endpoint, facilitate access of an endpoint to corporate resources, etc. The MDM platform can be utilized to facilitate many of these functions allowing modules to easily report back data or status information to the MDM platform. In some embodiments, modules may take advantage of these mechanisms during normal operation of the feature module, while in other embodiments certain feature modules may utilize the reporting mechanisms shown in FIGS. 4-5 to report back installation status, allowing feature modules include generic applications that may or may not be related to MDM functionality. In some embodiments, feature modules may wrap basic reporting features around non-MDM applications to allow basic reporting of status of the nation and installation to server components. **For example, where a non-MDM feature module is deployed, such as a sales application, the feature module used to deploy this sales application can include basic reporting functionality that utilizes the APIs of MDM agents on the endpoints to report back successful installation or other status information about the application. Data reported back can include any data that the module is configured to report**. Malik (Ex. 1005) at ¶[0084] **This allows server components of feature modules to correlate collected data with the endpoints** and more easily share data with other applications or the MDM platform. **This can allow reporting of data from endpoints to be used for various purposes on the server by applications**, policies, or for

	visibility provided to a dashboard or for other viewing by an administrator using the MDM platform. Malik (Ex. 1005) at ¶[0087] **FIG. 4 also shows the steps involved in passing data from feature modules executing on and points to the server**. At step 145 **endpoint agent 140 provides APIs that can be accessed by a developer of a feature module to pass data from the endpoint to server**. By invoking these APIs, at step 146, the feature modules executing in the scripting engine can collect data and command the scripting engine to report data back to server. Malik (Ex. 1005) at ¶[0088] Message handlers 155 or message processors 154 can also store the data from the messages in a shared data store 156 (which may store the uploaded data in native or processed form). When stored, the information can be associated with the individual endpoints that sent the data, allowing the data pertaining to a single endpoint to be gathered from a group of feature modules (which may be unrelated or related) and shared and accessed by an administrator or MDM application. This can allow feature modules to add features to existing MDM applications. The shared data store can allow a plurality of applications on the server side to access and mine the data returned. This information be accessed by user interface 159, which can include a dashboard for viewing attributes and data pertaining to groups or individual devices. This interface may be presented to a user via a webpage, allowing an administrator to application server 102 via a browser and manage endpoints and view information about them that is received from feature modules. Malik (Ex. 1005) at ¶[0092] FIG. 6A shows an exemplary screen shot that would be presented to a developer when elevating a module to the server for deploying a feature module to endpoint devices. This dashboard allows the administrator or developer to define to configure the module for deployment. Information added using this screen can be handled by the manifest server and module manager. The interface allows selection of a code name, as well as a name for the module to be deployed. The developer can also select a category and product bundle to associate with the module. This can allow modules to be bundled so that they can be deployed together, allowing a simplified pricing and rollout strategy to be used when deploying modules to customers. The developer can select a version and define SQL or XML statements that should be included in a manifest when the module is deployed to endpoints. Once a module is uploaded, the information input can be used to create the manifest once a bundle or module is selected for deployment to endpoints. Malik (Ex. 1005) at ¶[0094] Batch requester 162 provides a means by which the manifest server may listen to a user interface, such as a web based dashboard that allows an administrator

	to add functionality to groups of devices, and to push those changes in authorization of devices to the deployment and distribution system. Malik (Ex. 1005) at ¶[0099] Malik (Ex. 1005) at Fig. 4:  FIG. 4
Claim 2	
The system of claim 1, wherein the programmable network device and the programmable cloud device each include an access network interface unit configured to send and receive communications and a processor with a memory associated with the access network interface unit	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *[t]he system of claim 1. See* claim 1, *supra*. Malik discloses that *the programmable network device* (e.g., endpoint) *and the programmable cloud device* (e.g., server) *each include an access network interface unit* (e.g., communication device or communication circuit) *configured to send and receive communications and a processor* (e.g., processor) *with a memory* (e.g., memory) *associated with the access network interface unit*. Specifically, Malik discloses that the endpoint devices include a communication circuit and memory for storing data and instructions. Malik (Ex. 1005) at ¶[0035]. Malik also discloses that the components of the MDM system, whether on the server or the feature modules, can each include software operating on one or more processors, and that the computing devices described therein can each include one or more processors, memory, and communication devices. Malik (Ex. 1005) at ¶[0039]. Because "associated with" is a broad term, and

	the entire MDM system of Malik relates to deploying feature modules from the server to an endpoint device and providing data from the feature modules to the server via a network, *see* Malik (Ex. 1005) at Fig. 1, the processor and memory of the server and of the endpoint device are associated with the communication device/circuit that sends and receives the communications in each of the server and the endpoints. ***See, e.g.*, Malik** As used herein, endpoint devices can include any endpoint devices suitable for accessing or interacting with corporate resources. This may include PCs, laptops, mobile devices, smart phones, tablets, network clients, or the like. **Endpoints can be understood to have a processor, memory (which may include magnetic, optical, solid state, or remote memory) for storing data and instructions, and a communication circuit to allow communication with a network, either wirelessly or via a network cable.** Modules and software components, such as agents, described herein may be stored in non-transient memory on the endpoint. Malik (Ex. 1005) at ¶[0035] **The components of an MDM system on a server and the feature modules can include software operating on one or more processors.** This software can include software instructions, written in any suitable manner stored on non-transient computer readable media, such as optical, magnetic, solid-state memory devices, or local or remote storage. **Computing devices, as described herein can be understood to have one or more processors, memory, and communication devices. Software components can be stored locally or remotely, and be stored in non-transient memory**. Malik (Ex. 1005) at ¶[0039] Malik (Ex. 1005) at Fig. 1:  FIG. 1
Claim 3 [P]	

The system of claim 1 further comprising:	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *[t]he system of claim 1. See* claim 1, *supra*.
Claim 3 [a]	
an application management portal capable of managing life cycles of the distributed applications; and	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *an application management portal* (e.g., module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint) *capable of managing life cycles* (e.g., installing, updating and removing modules in accordance with manifests) *of the of the distributed applications* (e.g., feature applications). Requester notes that being "capable of managing life cycles of the distributed applications" is functional language. The recitation of such "functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the prior art", and additionally, if a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation. MPEP §2114; *See also* Request at Section II(D). To the extent that this limitation is given any patentable weight, it is disclosed either expressly or inherently in Malik. Additionally, to the extent that Malik does not explicitly disclose the management server managing life cycles of the plurality of network device applications and plurality of cloud applications, it is clear based on the disclosure of Malik that that the management server of Malik is capable of such management. Geier Decl. at ¶137. Specifically, Malik discloses that its system allows organizations to add, update, or remove features. Malik (Ex. 1005) at ¶¶[0030], [0078] and [0082], thus managing life cycles of the distributed applications. Malik discloses several components of its system that work together to accomplish this life cycle management. With reference to its Figures 2, and 3, Malik discloses a module manager and manifest server as components of the cloud server that keep track of all modules and their components as they are uploaded and maintained and manages pushing updates to endpoints by creating manifests that associate modules to be distributed to the endpoints. Malik (Ex. 1005) at ¶¶[0060]-[0063]. Malik discloses a heartbeat server 116 as a component of the cloud server that provides an interface from the server components to the agents across the network, and allows the transfer of manifests and modules. Malik (Ex. 1005) at ¶[0064]. Malik also discloses a module manager on the endpoint that receives, downloads and decrypts a waiting manifest, and in response to changes

	in the manifest also downloads or removes modules. Malik (Ex. 1005) at ¶¶[0076], [0078], and [0082]. ***See, e.g.*, Malik** In some embodiments of the present invention, deployment of applications and data to endpoints can be facilitated by a network-based solution. In some embodiments, a cloud-based approach is used. Agents can be installed on endpoint devices. These agents can facilitate installation of applications/feature modules on-demand. This can allow an organization to selectively add features and applications to endpoint devices without a complicated manual rollout. **This can allow an organization select features for use with a mobile device management (MDM) platform to target the needs of an organization, and allow features to be added or removed as the needs of the organization change**. When an MDM system is outsourced to a vendor, or is supplied by vendor, a simplified deployment and distribution approach to applications and features may allow the supplier or operator of the MDM system to offer flexible pricing that is tailored to the features used and the needs of an organization. Malik (Ex. 1005) at ¶[0030] Server 102 also includes module manager 112. **Module manager 112 can be a software component that keeps track of all modules and their components as they are uploaded and maintained. Module manager 112 can map the locations (e.g., URLs or other network locations) of these components. In some embodiments, it may also participate in determining which endpoints should receive feature modules that are uploaded to file server 110, while in other embodiments, this aspect may be handled by manifest server 114.** In some embodiments, by separating module manager 112 and manifest server 114 from file server 110, greater scalability can be achieved. File server 110 can be an off-the-shelf CDN component, while module manager 112 and manifest server 114 can be specialized software components developed to facilitate the deployment and distribution system of an MDM platform. In some embodiments module manager 112 can facilitate maintaining modules on a multitenant database. Multiple customers can share server resources for an MDM platform. Module manager 112 can include different or shared modules for different customers and or groups. **Module manager 112 can also keep track of multiple versions of modules to assist in operating endpoints to new versions of modules.** In some embodiments, multiple versions of a module can be used to allow a select group of endpoints to participate in a beta test of updated versions of a module. Malik (Ex. 1005) at ¶[0060] **Manifest server 114 manages pushing updates to endpoints by creating manifests that associate modules to be distributed to the endpoints with the groups of endpoints to receive the modules**. Manifests can be small files that can be easily distributed to endpoints. A manifest can associate the locations of module components with an identification of the group of endpoints that should

	receive the update. In some embodiments, the agent makes the decision of whether each individual endpoint belongs to that group, allowing greater scalability, as the manifest server 114 may not need identify the individual endpoints to receive the update. In other embodiments, the manifest may identify the specific endpoints to receive a module. In some embodiments, a manifest is a file that has XML or metadata that identifies the locations of the components of modules to install and the attributes of a group of endpoints (or the individual endpoints) for which the modules should be installed. **Endpoints can work with heartbeat server 116 to determine which manifests apply to each endpoint. Manifests may also include any security credentials, such as encryption keys, necessary for an endpoint to access the module components. Manifests can be distributed to agents across the network in a secure manner to protect these credentials. Manifests may also include a timestamp identifying the last change to the manifest or creation date and time**. Malik (Ex. 1005) at ¶[0062] When a **manifest server has created a new manifest, it may upload the manifest to file server 110 to allow agents to subsequently download the manifest. In addition, manifest server 114 will inform heartbeat server 116 of the new manifest**, and provide the timestamp of the manifest and, in some embodiments, decryption credentials to allow an agent to decrypt encrypted manifest stored on the file server. AES encryption may be used. Malik (Ex. 1005) at ¶[0063] **Heartbeat server 116 provides an interface from the server components to the agents 104 (and specifically file request manager 130, within each agent) across the network 128, which may include the Internet.** In some embodiments, heartbeat server 116 uses a lightweight protocol, whereby each time an agent checks in with the heartbeat server (which may be at a regular interval, such as every 15 minutes) the heartbeat server will identify any manifests that may relate to the endpoint and/or identify a timestamp of the last change to that manifest. The agent may keep track of the manifest or manifests that it has utilized in the past and compare the timestamps of the past manifest to determine if a new manifest is waiting. In some embodiments, the heartbeat server reports the timestamp or ID of the last manifest to the agent, allowing the agent to make the determination of whether a new manifest should be downloaded. Malik (Ex. 1005) at ¶[0064] **Module manager 132 resides on the agent and is responsible for managing the components of feature modules. In some embodiments, in response to a new timestamp on the manifest, module manager 132 can reach out to file server 110 to download and decrypt the waiting manifest**. In some embodiments, the module manager can receive the new manifest from file

request manager 130. Module manager 132 can then evaluate the contents of the manifest.
Malik (Ex. 1005) at ¶[0076]

In response to a changed module component identified in the manifest, module manager 132 can communicate with file server 110 across a network, such as Internet, to download these modules and utilize encryption keys stored in the manifest to decrypt the contents. In some embodiments, module manager 132 may need to negotiate with file server 110 to determine which module content has changed and should be downloaded, while in others, this information is provided in the manifest. In some embodiments, module manager 132 only downloads changed content to save on network bandwidth, allowing file server 110 to be more scalable and to reduce the overhead data used by each endpoint.
Malik (Ex. 1005) at ¶[0078]

In some embodiments, **the manifest can also be used to remove modules**. In some embodiments, a manifest is a complete list of all modules that should be installed on an endpoint. This can be conveyed either through a single manifest for an organization that allows agents to identify the group to which the receiving endpoints belongs, allowing different endpoints within an organization to have different modules, or through a manifest file that includes only the modules that should be installed on the receiving endpoint. **The manifest can include specific instructions to delete modules, or may simply exclude modules that are no longer to be used. The module manager can respond to this information by removing any modules that are not identified in the manifest. This can provide a simple way of controlling the modules that are used by a given endpoint**.
Malik (Ex. 1005) at ¶[0082]

Malik (Ex. 1005) at Fig. 2:

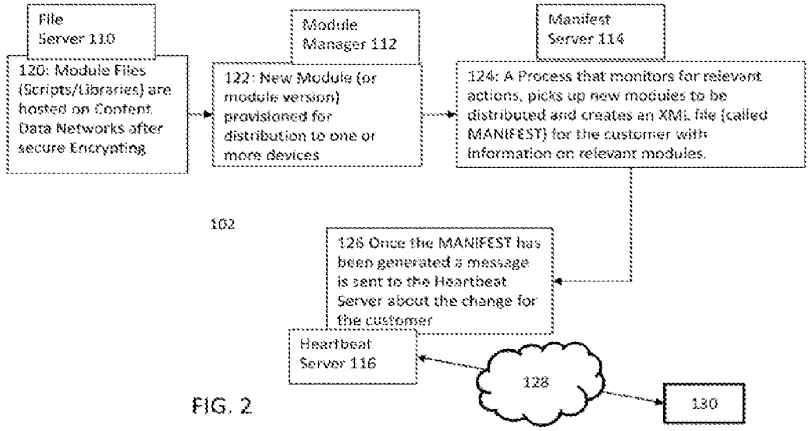
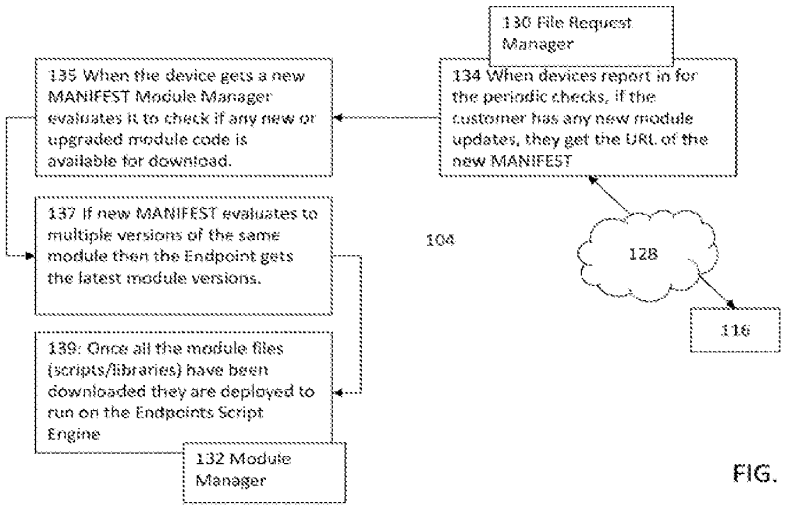


FIG. 2

	Malik (Ex. 1005) at Fig. 3:  FIG. 3
Claim 3 [b]	
an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested and certified distributed applications to the application management portal.	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *an infrastructure application marketplace* (e.g. web-based portal and file server) *in communication with the application management portal* (e.g., module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint)*, said infrastructure marketplace capable of providing tested and certified distributed applications to the application management portal* (e.g., modules can be tested before deploying new or updated feature modules). Requester notes that being "capable of providing tested and certified distributed applications to the application management portal" is functional language. The recitation of such "functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the prior art", and additionally, if a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation. MPEP §2114; *See also* Request at Section II(D). To the extent that this limitation is given any patentable weight, it is disclosed either expressly or inherently in Malik. Additionally, to the extent that Malik does not disclose this element though explicit disclosure of providing such applications, based on the disclosure of Malik it is clear that the application store of Malik would be "capable of" providing such applications. Geier Decl. at ¶84. Specifically, Malik discloses a web-based portal that allows for deployment of feature modules, discloses the selective purchasing of features, and also

	discloses a file server that acts as a repository to store the data/code components that comprise the feature modules. Malik (Ex. 1005) at ¶¶ [0009], [0031], [0059] and [0067]. The file server allows modules to be elevated to the server infrastructure and later downloaded by agents on endpoints. Malik (Ex. 1005) at ¶ [0059]. The file server is in communication with the components of the cloud server that make up the application management portal. Malik (Ex. 1005) at ¶¶ [0060], [0067] and Fig 2. Malik also discloses that the applications can be tested prior to being made available in the file server. Although Malik discloses that "less-tested" modules may be used and that robust testing may not be required, that is a disclosure that modules can be tested prior to being provided to the file server. Malik (Ex. 1005) at ¶¶[0045] and [0054]. ***See, e.g.*, Malik** One aspect of some embodiments can include providing a web-based portal that allows a user to deploy the subset of feature modules to the endpoint computers. Another aspect of some embodiments can include providing at one or more servers, a heartbeat server configured to receive requests for updates from the plurality of endpoint on a periodic basis. Malik (Ex. 1005) at ¶[0009] In addition to allowing customers of an MDM platform to selectively purchase and deploy features, in some embodiments, a user of MDM platform may be able to deploy third-party applications and feature packs or develop its own features and applications. Malik (Ex. 1005) at ¶[0031] First, in embodiments where the device need not be restarted, the end-user of the device may be unaware or not generally inconvenienced by updates. This can allow updates and new modules to be deployed to endpoint without the typical logistic headaches associated with traditional rollouts of software. For example, traditional rollouts may have required system down time, advance notice, restarting of devices on the network, and substantial testing before a rollout. In some embodiments, an update or new module can be deployed to endpoint devices at any time, without requiring system downtime, advance warning to users, or input from users. Second, in some embodiments where MDM components on the endpoint device need not be restarted, MDM components, such as agents or other feature modules, can continue to operate, allowing endpoint agents and management and security policies to continue to function normally during the update process. This can avoid security risks associated with disabling these MDM components. Furthermore, where updates are unobtrusive to the end-user, and where updates can frequently occur without greatly diminishing the user experience, the MDM system may provide advantages to developers of feature modules. For example, **substantial beta**

	testing of updates may not be required before deploying new or updated feature modules. In addition, an update can be provided to a group of endpoints and the group of endpoints can act as a real-world beta testing group for these updates. This may greatly reduce the amount of development time required for producing reliable, new or updated feature modules. Malik (Ex. 1005) at ¶[0045] In some embodiments, the end-to-end deployment and reporting of installation status of feature modules can provide further advantages. For example, in some embodiments, the deployment process can be substantially automated using the deployment and distribution system. Once a request is submitted, the feature module will be deployed to a plurality of endpoints and the status of the deployment can be checked at any time by administrator. Furthermore, information about unexpected status of deployments, such as failed installations, can be reported to an administrator as an alert. **Because the status of a device and the individual feature modules can be reported, an administrator may be able to monitor status of deployed modules, allowing less-tested modules to be freely distributed without traditional headaches associated with robustly testing modules.** For example, an administrator may be able to recall and install feature modules from a plurality of devices if a bug is discovered. This can reduce the development cost of feature modules. Furthermore, because the deployment and distribution system may be used to deploy updates relatively easily, version maintenance of applications on the endpoints can be greatly simplified. An administrator may be able to quickly tell which endpoints are not running updated versions and use the deployment system to send updated versions to these noncompliant endpoints. Malik (Ex. 1005) at ¶[0054] **FIG. 2 depicts the server components for facilitating deployment and distribution of feature modules. FIG. 2 illustrates the various server components and the steps that they perform to facilitate distribution of features. File server 110 acts as a repository to store the data/code components that comprise the feature modules. File server 110 allows modules to be elevated to the server infrastructure and later downloaded by agents on endpoints.** In some embodiments, file server 110 may be part of a content distribution network (CDN), such as a vendor network (e.g., Akamai). File server 110 may use a cloud-based approach allowing mirroring and dynamic load balancing to facilitate upload and download of module content. Files stored on file server 110 can include the data and instructions that constitute a feature module. Malik (Ex. 1005) at ¶[0059] Server 102 also includes module manager 112 . Module manager 112 can be a software component that keeps track of all modules and their components as they are uploaded and maintained. Module manager 112 can map the locations (e.g., URLs or other network locations) of these components. In some

	embodiments, it may also participate in determining which endpoints should receive feature modules that are uploaded to file server 110 , while in other embodiments, this aspect may be handled by manifest server 114 . In some embodiments, by separating module manager 112 and manifest server 114 from file server 110 , greater scalability can be achieved. File server 110 can be an off-the-shelf CDN component, while module manager 112 and manifest server 114 can be specialized software components developed to facilitate the deployment and distribution system of an MDM platform. In some embodiments module manager 112 can facilitate maintaining modules on a multitenant database. Multiple customers can share server resources for an MDM platform. Module manager 112 can include different or shared modules for different customers and or groups. Module manager 112 can also keep track of multiple versions of modules to assist in operating endpoints to new versions of modules. In some embodiments, multiple versions of a module can be used to allow a select group of endpoints to participate in a beta test of updated versions of a module. Malik (Ex. 1005) at ¶[0060] FIG. 2 also depicts the steps used in uploading and distributing new modules. Steps 120 through 126 are carried out by respective components 110 through 116. **At step 120, module files, including scripts, libraries, data, or other code components are uploaded to a file server 110.** This can include uploading to a CDN. These components may be encrypted. At step 122 when a module is created (or an update or new version is created), module manager 112 will provision the new module by associating the various components and identifying the location of these components on the CDN. Module manager 112 can make this information available to manifest server 114. At step 124, manifest server 114 monitors requests to distribute modules. These requests may come from administrators explicitly or may be responsive to updates to existing modules as identified by module manager 112. Manifest server 114 then creates a manifest file that associates module files with groups of endpoints to receive the module. At step 126 once a manifest has been generated manifest server 114 informs heartbeat server 116. This informs the heartbeat server 116 that the timestamp of a manifest has changed. The next time relevant endpoints check in with the heartbeat server, heartbeat server 116 will inform these endpoints that a new manifest is waiting, allowing agents on these endpoints to seek out and download new module components. Malik (Ex. 1005) at ¶[0067] Malik (Ex. 1005) at Fig. 2:

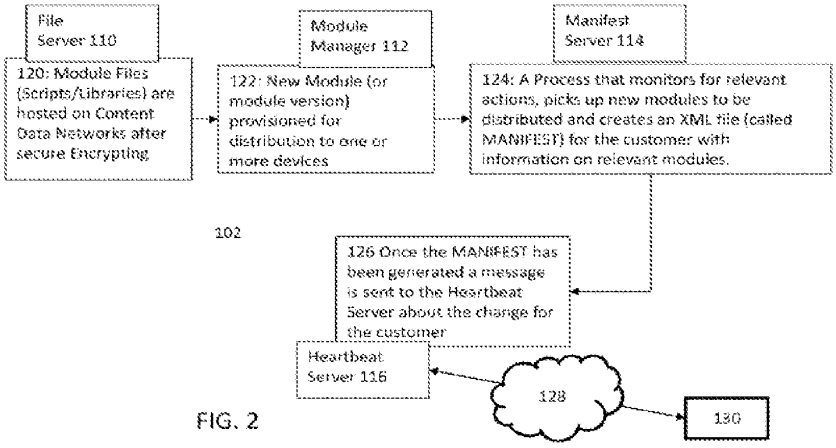


FIG. 2

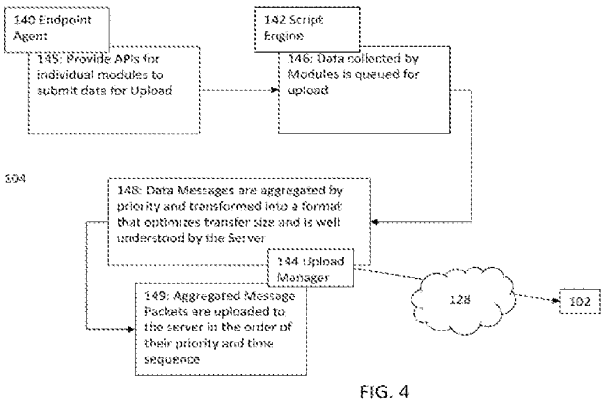
Claim 4	
The system of claim 3, wherein the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications.	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *[t]he system of claim 3. See* claim 3, *supra*. Malik discloses that *the application management portal* (e.g., module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint) *is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace* (e.g. web-based portal and file server); *testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications.* Requester notes that this limitation recites a Markush group. MPEP §2117. Accordingly, disclosure of any one of the options within the group anticipates the limitation Requester also notes that being "capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications" is functional language. The recitation of such "functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the prior art", and additionally, if a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation. MPEP §2114; *See also* Request at Section II(D).

	To the extent that this limitation is given any patentable weight, it is disclosed either expressly or inherently in Malik. Specifically, Malik discloses that its system allows organizations to add, update, or remove features, Malik (Ex. 1005) at ¶¶[0030], [0078] and [0082], thus performing each of: receiving new applications from the infrastructure application marketplace; provisioning said distributed applications; and deprovisioning said distributed applications. With reference to its Figures 2, and 3, Malik discloses a module manager and manifest server as components of the cloud server that keep track of all modules and their components as they are uploaded and maintained in the file server, and that manage pushing updates to endpoints by creating manifests that associate modules to be distributed to the endpoints. Malik (Ex. 1005) at ¶¶[0060]-[0063]. Malik discloses a heartbeat server 116 as a component of the cloud server that provides an interface from the server components to the agents across the network, and allows the transfer of manifests and modules. Malik (Ex. 1005) at ¶[0064]. Malik also discloses a module manager on the endpoint that receives, downloads and decrypts a waiting manifest, and in response to changes in the manifest also downloads or removes modules. Malik (Ex. 1005) at ¶¶[0076], [0078], and [0082]. Thus, Malik discloses that the module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint perform (are and capable of performing) at least *receiving new applications from the infrastructure application marketplace; provisioning said distributed applications; and deprovisioning said distributed applications.* To the extent that Malik does not explicitly disclose the application management portal performing at least one of the recited actions, it is clear based on the disclosure in Malik that the management server of Malik is capable of at least *receiving new applications from the infrastructure application marketplace; provisioning said distributed applications; and deprovisioning said distributed applications*. Geier Decl. at ¶93. ***See, e.g.*, Malik** In some embodiments of the present invention, deployment of applications and data to endpoints can be facilitated by a network-based solution. In some embodiments, a cloud-based approach is used. Agents can be installed on endpoint devices. These agents can facilitate installation of applications/feature modules on-demand. This can allow an organization to selectively add features and applications to endpoint devices without a complicated manual rollout. **This can allow an organization select features for use with a mobile device management (MDM) platform to target the needs of an organization, and allow features to be added or removed as the needs of the organization change**. When an MDM system is outsourced to a vendor, or is supplied by vendor, a simplified deployment and distribution approach to applications and features may allow the supplier or operator of the MDM system to offer flexible pricing that is tailored to the features used and the needs of an organization.

	Malik (Ex. 1005) at ¶[0030] Server 102 also includes module manager 112. **Module manager 112 can be a software component that keeps track of all modules and their components as they are uploaded and maintained. Module manager 112 can map the locations (e.g., URLs or other network locations) of these components. In some embodiments, it may also participate in determining which endpoints should receive feature modules that are uploaded to file server 110, while in other embodiments, this aspect may be handled by manifest server 114.** In some embodiments, by separating module manager 112 and manifest server 114 from file server 110, greater scalability can be achieved. File server 110 can be an off-the-shelf CDN component, while module manager 112 and manifest server 114 can be specialized software components developed to facilitate the deployment and distribution system of an MDM platform. In some embodiments module manager 112 can facilitate maintaining modules on a multitenant database. Multiple customers can share server resources for an MDM platform. Module manager 112 can include different or shared modules for different customers and or groups. **Module manager 112 can also keep track of multiple versions of modules to assist in operating endpoints to new versions of modules.** In some embodiments, multiple versions of a module can be used to allow a select group of endpoints to participate in a beta test of updated versions of a module. Malik (Ex. 1005) at ¶[0060] **Manifest server 114 manages pushing updates to endpoints by creating manifests that associate modules to be distributed to the endpoints with the groups of endpoints to receive the modules**. Manifests can be small files that can be easily distributed to endpoints. A manifest can associate the locations of module components with an identification of the group of endpoints that should receive the update. In some embodiments, the agent makes the decision of whether each individual endpoint belongs to that group, allowing greater scalability, as the manifest server 114 may not need identify the individual endpoints to receive the update. In other embodiments, the manifest may identify the specific endpoints to receive a module. In some embodiments, a manifest is a file that has XML or metadata that identifies the locations of the components of modules to install and the attributes of a group of endpoints (or the individual endpoints) for which the modules should be installed. **Endpoints can work with heartbeat server 116 to determine which manifests apply to each endpoint. Manifests may also include any security credentials, such as encryption keys, necessary for an endpoint to access the module components. Manifests can be distributed to agents across the network in a secure manner to protect these credentials. Manifests may also include a timestamp identifying the last change to the manifest or creation date and time**. Malik (Ex. 1005) at ¶[0062]

	When a **manifest server has created a new manifest, it may upload the manifest to file server 110 to allow agents to subsequently download the manifest. In addition, manifest server 114 will inform heartbeat server 116 of the new manifest**, and provide the timestamp of the manifest and, in some embodiments, decryption credentials to allow an agent to decrypt encrypted manifest stored on the file server. AES encryption may be used. Malik (Ex. 1005) at ¶[0063] **Heartbeat server 116 provides an interface from the server components to the agents 104 (and specifically file request manager 130, within each agent) across the network 128, which may include the Internet.** In some embodiments, heartbeat server 116 uses a lightweight protocol, whereby each time an agent checks in with the heartbeat server (which may be at a regular interval, such as every 15 minutes) the heartbeat server will identify any manifests that may relate to the endpoint and/or identify a timestamp of the last change to that manifest. The agent may keep track of the manifest or manifests that it has utilized in the past and compare the timestamps of the past manifest to determine if a new manifest is waiting. In some embodiments, the heartbeat server reports the timestamp or ID of the last manifest to the agent, allowing the agent to make the determination of whether a new manifest should be downloaded. Malik (Ex. 1005) at ¶[0064] **Module manager 132 resides on the agent and is responsible for managing the components of feature modules. In some embodiments, in response to a new timestamp on the manifest, module manager 132 can reach out to file server 110 to download and decrypt the waiting manifest**. In some embodiments, the module manager can receive the new manifest from file request manager 130. Module manager 132 can then evaluate the contents of the manifest. Malik (Ex. 1005) at ¶[0076] **In response to a changed module component identified in the manifest, module manager 132 can communicate with file server 110 across a network, such as Internet, to download these modules and utilize encryption keys stored in the manifest to decrypt the contents.** In some embodiments, module manager 132 may need to negotiate with file server 110 to determine which module content has changed and should be downloaded, while in others, this information is provided in the manifest. In some embodiments, module manager 132 only downloads changed content to save on network bandwidth, allowing file server 110 to be more scalable and to reduce the overhead data used by each endpoint. Malik (Ex. 1005) at ¶[0078] In some embodiments, **the manifest can also be used to remove modules**. In some embodiments, a manifest is a complete list of all modules that should be

	installed on an endpoint. This can be conveyed either through a single manifest for an organization that allows agents to identify the group to which the receiving endpoints belongs, allowing different endpoints within an organization to have different modules, or through a manifest file that includes only the modules that should be installed on the receiving endpoint. **The manifest can include specific instructions to delete modules, or may simply exclude modules that are no longer to be used. The module manager can respond to this information by removing any modules that are not identified in the manifest. This can provide a simple way of controlling the modules that are used by a given endpoint**. Malik (Ex. 1005) at ¶[0082]
Claim 15	
The system of claim 1 further comprising: a distributed content service wherein contents generated by the at least one of the plurality of network device applications are shared with at least one of the plurality of cloud applications.	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *[t]he system of claim 1. See* claim 1, *supra*. Malik discloses *a distributed notification service* (e.g. reporting system) *wherein contents* (e.g., status and information) *generated by the at least one of the plurality of network device applications* (e.g., feature modules installed on the endpoint device) *are shared with at least one of the plurality of cloud applications* (e.g., feature modules hosted by the server). Specifically, with respect to its Figure 4, Malik discloses the end point components of a reporting system that can be used as part of the module distribution and deployment system to allow deployed modules to "report back status related to their installation as well as report back information to server components during normal execution." Malik (Ex. 1005) at ¶[0087]; *see also Id*. at ¶[0057] and Fig. 4. ***See, e.g.*, Malik** FIG. 1 shows a high level architecture of the deployment and distribution system as part of an MDM platform. System 100 contains server components and endpoint components. An application server 102 provides server functionality as described herein. This application server can be a SaaS application server providing server functionality in a cloud-based approach, which include a plurality of servers hosting the application or components thereof. **Agents 104 on endpoints 105 provide core MDM functionality including allowing actions to be executed on the endpoints and allowing data and status information to be returned to the server.** Agents 104 can include a plurality of endpoint applications that provide agent functionality. SaaS application server 102 may transmit actions and modules across a link 106 , which may traverse a network, such as the Internet. Agents 104 may receive these modules and actions and install modules or execute actions accordingly. Agents 104 can also transmit data to the application server across link 108 which may also

	traverse a network, such as the Internet. This allows agents to report back status of deployments and data collected or generated by the deployed feature modules to application server 102. Malik (Ex. 1005) at ¶[0057] [0084] **FIG. 4 depicts the end point components of a reporting system that can be used as part of the module distribution and deployment system. Reporting component can allow deployed modules to report back status related to their installation as well as report back information to server components during normal execution. This can allow developers of modules to take advantage of APIs as part of the MDM platform that allow reporting of information gathered by modules.** For example, modules may be developed that determine status information about an endpoint, apply compliance policies to an endpoint, apply configuration to an endpoint, facilitate access of an endpoint to corporate resources, etc. The MDM platform can be utilized to facilitate many of these functions allowing modules to easily report back data or status information to the MDM platform. In some embodiments, modules may take advantage of these mechanisms during normal operation of the feature module, while in other embodiments certain feature modules may utilize the reporting mechanisms shown in FIGS. 4-5 to report back installation status, allowing feature modules include generic applications that may or may not be related to MDM functionality. In some embodiments, feature modules may wrap basic reporting features around non-MDM applications to allow basic reporting of status of the nation and installation to server components. For example, where a non-MDM feature module is deployed, such as a sales application, the feature module used to deploy this sales application can include basic reporting functionality that utilizes the APIs of MDM agents on the endpoints to report back successful installation or other status information about the application. Data reported back can include any data that the module is configured to report. Malik (Ex. 1005) at ¶[0084] Malik (Ex. 1005) at Fig. 4  FIG. 4

Claim 19 [P]	
A system comprising:	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *a system*. *See* Claim 1[P], *supra.*
Claim 19 [a]	
a programmable network device adapted to host a plurality of network device applications;	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *a programmable network device* (e.g., endpoint) *adapted to host a plurality of network device applications* (e.g., feature modules installed on the endpoint device). *See* Claim 1[a], *supra.*
Claim 19 [b]	
a programmable cloud device adapted to host a plurality of cloud applications,	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *a programmable cloud device* (e.g., server) *adapted to host a plurality of cloud applications* (e.g., feature modules hosted by the server). *See* Claim 1[b], *supra.*
Claim 19[c]	
wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications;	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses that *the plurality of network device applications* (e.g., feature modules installed on the endpoint device) *and the plurality of cloud applications* (e.g., feature modules hosted by the server) *are in secure communication with each other to form distributed applications*. *See* Claim 1[c], *supra.*
Claim 19 [d]	
wherein the plurality of network device applications and plurality of cloud applications form unified capabilities enabling a plurality of upper layer application	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses that *the plurality of network device applications* (e.g., feature modules installed on the endpoint device) *and plurality of cloud applications* (e.g., feature modules hosted by the server) *form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device* (e.g., endpoint) *hardware and cloud device* (e.g., server) *hardware*. *See* Claim 1[d], *supra.*

programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware;	Requester notes that the only difference between Claim 1[d], *supra*, and Claim 19[d] is that Claim 1[d], *supra*, includes the word "device" after the first recitation of "cloud applications", which appears to be a typographical error.
Claim 19 [e]	
an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications; and	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Requester notes that the only difference between Claim 3[a], supra, and Claim 19[e] is that Claim 3[a], supra, recites "distributed applications" while 19[e] recites "the plurality of network device applications and plurality of cloud applications." However, because "the plurality of network device applications and plurality of cloud applications" are what forms the "distributed applications", the same disclosures in Stewart are equally applicable to both limitations. Geier Decl. at ¶136. Malik discloses *an application management portal* (e.g., module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint) *capable of managing life cycles of the plurality of network device applications* (e.g., feature modules installed on the endpoint device) *and plurality of cloud applications* (e.g., feature modules hosted by the server). *See Claim 3[a], supra.*
Claim 19 [f]	
an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested	Malik anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Malik discloses *an infrastructure application marketplace* (e.g. web-based portal and file server) *in communication with the application management portal* (e.g., module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint), *said infrastructure marketplace capable of providing tested distributed applications to the application management portal*. *See* Claim 3[b], *supra.*

distributed applications to the application management portal.	

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
134

Ex. 1003

Declaration of James T. Geier
("Geier Decl.")

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

U.S. Patent No. 11,695,823	)	Attorney Docket No. UNP-16002
	)	
Issued: July 4, 2023	)	
	)	
Filed: January 6, 2021	)	
	)	
Title: DISTRIBUTED SOFTWARE DEFINED NETWORKING	)	
	)	
	)	
	)	

DECLARATION OF JAMES T. GEIER
IN SUPPORT OF REQUEST FOR EX PARTE REEXAMINATION OF U.S. PATENT NO. 11,695,823

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
136

TABLE OF CONTENTS

SECTION	PAGE
I. Background and Qualifications	1
II. Legal Framework	1
A. Anticipation	1
B. Obviousness	2
i. General Framework	2
ii. Analogous Art	3
iii. Combining Known Elements in the Art, Common Sense	3
iv. Teaching Away	5
v. Secondary Considerations	5
C. Claim Construction	6
III. Opinion	9
A. Level of Skill of a POSITA	9
B. Summary of My Opinions	10
C. Summary of the '823 Patent	10
D. Ground 1: Stewart Anticipates the Challenged Claims	14
1. Overview of Stewart	14
2. Application of Stewart to the Challenged Claims	18
a) '823 Patent at Independent Claims 1 and 19: Stewart Discloses a "*system*"	18
b) '823 Patent at Independent Claims 1 and 19: Stewart Discloses "a programmable network device adapted to host a plurality of network device applications"	18
c) '823 Patent at Independent Claims 1 and 19: Stewart Discloses "a programmable cloud device adapted to host a plurality of cloud applications"	18
d) '823 Patent at Independent Claims 1 and 19: Stewart Discloses "wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications"	19
e) '823 Patent at Independent Claims 1 and 19: Stewart Discloses "wherein the plurality of network device applications and plurality of cloud applications device [sic] form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware	20
f) '823 Patent at Independent Claim 19 and Dependent Claim 3: Stewart Discloses "an application management portal capable of managing life cycles of the plurality of network device	

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
137

applications and plurality of cloud applications" and "an application management portal capable of managing life cycles of the distributed applications" 22
g) '823 Patent at Independent Claim 19 and Dependent Claim 3: Stewart Discloses "an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested distributed applications to the application management portal"..... 24
h) '823 Patent at Dependent Claim 2: Stewart Discloses "wherein the programmable network device and the programmable cloud device each include an access network interface unit configured to send and receive communications and a processor with a memory associated with the access network interface unit" 25
i) '823 Patent at Claim 4: Stewart Discloses "wherein the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications"..... 26
j) '823 Patent at Dependent Claim 8: Stewart Discloses "a distributed notification service wherein notice of a predetermined event is sent from at least one of the plurality of network device applications to at least one of the plurality of cloud applications" 27
k) '823 Patent at Dependent Claim 15: Stewart Discloses "a distributed content service wherein contents generated by the at least one of the plurality of network device applications are shared with at least one of the plurality of cloud applications"..... 28
E. Ground 2: Stewart Renders Obvious the Challenged Claims..... 29
1. Overview of the Knowledge of a POSITA..... 29
2. Stewart in View of the Knowledge of a POSITA..... 30
F. Ground 3: Malik Anticipates Claims 1, 2, 3, 4, 15 and 19 of the '823 Patent... 31
1. Overview of Malik..... 31
2. Application of Malik to Claims 1, 2, 3, 4, 15 and 19 of the '823 Patent 35
a) '823 Patent at Independent Claims 1 and 19: Malik Discloses a "*system*" 35
b) '823 Patent at Independent Claims 1 and 19: Malik Discloses "a programmable network device adapted to host a plurality of network device applications" 35
c) '823 Patent at Independent Claims 1 and 19: Malik Discloses "a programmable cloud device adapted to host a plurality of cloud applications" 36
d) '823 Patent at Independent Claims 1 and 19: Malik Discloses "wherein the plurality of network device applications and the plurality of cloud applications are in secure communication

with each other to form distributed applications" 36
e) '823 Patent at Independent Claims 1 and 19: Malik Discloses "wherein the plurality of network device applications and plurality of cloud applications device [sic] form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware..... 36
f) '823 Patent at Independent Claim 19 and Dependent Claim 3: Malik Discloses "an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications" and "an application management portal capable of managing life cycles of the distributed applications" 39
g) '823 Patent at Independent Claim 19 and Dependent Claim 3: Malik Discloses "an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested distributed applications to the application management portal"..... 40
h) '823 Patent at Dependent Claim 2: Malik Discloses "wherein the programmable network device and the programmable cloud device each include an access network interface unit configured to send and receive communications and a processor with a memory associated with the access network interface unit" 41
i) '823 Patent at Claim 4: Malik Discloses "wherein the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications"..... 42
j) '823 Patent at Dependent Claim 15: Malik Discloses "a distributed content service wherein contents generated by the at least one of the plurality of network device applications are shared with at least one of the plurality of cloud applications"..... 43
G. Ground 4: Malik Renders Obvious Claims 1, 2, 3, 4, 15 and 19 of the '823 Patent 43
1. Malik in View of the Knowledge of a POSITA.....44
H. Conclusion.....45

I. Background and Qualifications

1. In addition to the following, my qualifications are further set forth in my curriculum vitae, a copy of which is submitted as Appendix A to this Declaration.

2. I received a B.S. in Electrical Engineering from California State University in 1985, and an M.S. in Electrical Engineering from the Air Force Institute of Technology in 1990.

3. I am currently, and have been for about 25 years, the principal consultant and founder for Wireless-Nets, Ltd., which is a company that, among other things, designs large-scale wireless networks for municipalities, hospitals, airports, and manufacturing facilities for supporting voice, video and data applications.

4. I have over 40 years of experience in the communications industry designing, analyzing, and implementing communications systems, wireless networks, and mobile wireless devices.

5. I have authored over a dozen books on mobile and wireless topics, including Designing and Deploying 802.11 Wireless Networks (Cisco Press), Implementing 802.1X Security Solutions (Wiley), Wireless Networking Handbook (New Riders) and Network Re-engineering (McGraw-Hill).

6. I have been an active participant within standards organizations, such as the IEEE 802.11 Working Group and the Wi-Fi Alliance, and have served as Chairman of the IEEE Computer Society, Dayton Section, and various conferences dealing with wireless networking.

II. Legal Framework

7. I am not an attorney; I am a technical expert and do not offer any legal opinions. However, for the purposes of this Declaration, I have been informed about certain legal principles regarding patentability and related matters under United States patent law by Requestor's counsel, which I have applied in performing my analysis and arriving at my technical opinions in this matter.

A. Anticipation

8. I am informed that, pursuant to 35 U.S.C. §102 a person cannot obtain a patent on an invention if: the claimed invention was patented, described in a printed publication, or in public use, on sale, or otherwise available to the public before the effective filing date of the claimed invention; or (2) the claimed invention was described in a patent issued under 35 U.S.C. §151, or in an application for patent published or deemed published under 35 U.S.C. §122(b), in

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
140

which the patent or application, as the case may be, names another inventor and was effectively filed before the effective filing date of the claimed invention.

9. In considering whether certain prior art anticipates a claim, I am informed that a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference. I am further informed that, in order to be anticipatory, the identical invention must be shown in the prior art reference in as complete detail as is contained in the claim, and that the elements must be arranged as required by the claim, but that this is not an *ipsissimis verbis test*, i.e., identity of terminology is not required.

B. Obviousness

1. General Framework

10. I am informed that a person cannot obtain a patent on an invention if the differences between the invention and the prior art are such that the subject matter as a whole would have been obvious to a person having ordinary skill in the art (a "POSITA") at the time of the invention. I am informed that a conclusion of obviousness may be founded upon more than a single item of prior art. I am also informed that obviousness is determined by evaluating the following factors: (1) the scope and content of the prior art, (2) the differences between the prior art and the claim at issue, (3) the level of ordinary skill in the pertinent art, and (4) secondary considerations of non-obviousness. In addition, the obviousness inquiry should not be done in hindsight. Instead, the obviousness inquiry should be done through the eyes of a person having ordinary skill in the art at the time of the alleged invention.

11. In considering whether certain prior art renders a particular patent claim obvious, I am informed that one must consider the scope and content of the prior art, examples of which include disclosures in patents, trade publications, journal articles, conference papers, industry standards, product literature and documentation, texts describing competitive technologies, requests for comment published by standard setting organizations, and materials from industry conferences.

12. I am informed that for a patent claim to be found obvious, the proper perspective to consider is that of a person having ordinary skill in the art at the time of the alleged invention. I am informed that the factors to consider in determining the level of ordinary skill in the art include (1) the educational level and experience of people working in the field at the time the

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
141

invention was made, (2) the types of problems faced in the art and the solutions found to those problems, and (3) the sophistication of the technology in the field.

2. Analogous Art

13. I am informed that for a prior art reference to be proper for use in an obviousness analysis, the reference must be "analogous art" to the claimed invention. I am informed that a reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention). In order for a reference to be "reasonably pertinent" to the problem, it must logically have commended itself to an inventor's attention in considering his/her problem. In determining whether a reference is reasonably pertinent, one should consider the problem faced by the inventor, as reflected either explicitly or implicitly, in the specification. I believe that all of the references I considered in forming my opinions in support of the Request are well within the range of references a person having ordinary skill in the art would have consulted by the time of the invention to address the type of problems described in the Challenged Claims.

3. Combining Known Elements in the Art, Common Sense

14. I am informed that, in order to establish that a claimed invention was obvious based on a combination of prior art elements, a clear articulation of the reason(s) why a claimed invention would have been obvious must be provided. Specifically, I am informed that, under the U.S. Supreme Court's KSR decision, a combination of multiple items of prior art renders a patent claim obvious when there was an apparent reason for a person having ordinary skill at the time of the invention to combine or modify the prior art. I am informed that such reasons may include, but is not limited to, one or more of the following rationales: (A) combining prior art methods according to known methods to yield predictable results; (B) substituting one known element for another to obtain predictable results; (C) using a known technique to improve a similar device in the same way; (D) applying a known technique to a known device ready for improvement to yield predictable results; (E) trying a finite number of identified, predictable potential solutions, with a reasonable expectation of success; (F) identifying that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill

in the art; or (G) identifying an explicit teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine the prior art references to arrive at the claimed invention.

15. I am informed that the test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference, but rather whether one of skill would have been motivated to combine the teachings of the prior art references to achieve the claimed invention. Nevertheless, I am informed that an invention that is a combination of prior art must do more than yield predictable results to be non-obvious and that, even where one or more motivations to combine exist, claims are not necessarily obvious unless a person having ordinary skill in the art would have had a reasonable expectation of success in implementing or applying the proposed combination at the time of the invention.

16. I also am informed that in conducting an obviousness analysis, a precise teaching directed to the specific subject matter of the challenged claim need not be sought out because it is appropriate to take into account the inferences and creative steps that a POSITA would employ. In other words, the prior art need not be directed towards solving the same specific problem as the problem addressed by the patent. Further, the individual prior art references themselves need not all be directed towards solving the same problem. The prior art considered can be directed to any need or problem known in the field of endeavor at the time of invention and can provide a reason for combining the elements of the prior art in the manner claimed. Common sense teaches that familiar items may have obvious uses beyond their primary purposes, and a POSITA will often have been able to fit the teachings of multiple prior art references like pieces of a puzzle. I am informed that, under the KSR standard, common sense is important and should be considered in assessing obviousness.

17. I also am informed that the fact that a particular combination of prior art elements was "obvious to try" may indicate that the combination was obvious even if no one attempted the combination. If the combination was obvious to try (regardless of whether it was actually tried) or leads to anticipated success, then it is likely the result of ordinary skill and common sense rather than innovation. I am further informed that in many fields it may be that there is little discussion of obvious techniques or combinations, and it often may be the case that market demand, rather than scientific literature or knowledge, will drive the design of an invention. When there is a design need or market pressure to solve a problem and there are a finite number

of identified, predictable solutions, a POSITA has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation, but of ordinary skill and common sense.

4. Teaching Away

18. I am informed that it is improper to combine references where the references teach away from their combination. I am informed that a reference may be said to teach away when a POSITA, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the patent applicant. In general, a reference will teach away if it suggests that the line of development flowing from the reference's disclosure is unlikely to be productive of the result sought by the patentee. I am informed that a reference teaches away, for example, if (1) the combination would produce a seemingly inoperative device, or (2) the references leave the impression that the product would not have the property sought by the patentee. However, I also am informed, that a reference does not teach away if it merely expresses a general preference for an alternative invention but does not criticize, discredit, or otherwise discourage investigation into the invention claimed. Further, a POSITA will often appreciate that different courses of action often have simultaneous advantages and disadvantages, and I am informed the existence of such does not necessarily obviate a motivation to combine.

5. Secondary Considerations

19. I am informed that even if a prima facie case of obviousness is established, the final determination of obviousness must also consider "secondary considerations" if presented. In most instances, the patentee raises these secondary considerations of non-obviousness. In that context, the patentee argues an invention would not have been obvious in view of these considerations, which include: (a) commercial success of a product due to the merits of the claimed invention; (b) a long-felt, but unsatisfied need for the invention; (c) failure of others to find the solution provided by the claimed invention; (d) deliberate copying of the invention by others; (e) unexpected results achieved by the invention; (f) praise of the invention by others skilled in the art; (g) lack of independent simultaneous invention within a comparatively short space of time; (h) teaching away from the invention in the prior art.

20. I am further informed that secondary considerations evidence is only relevant if the offering party establishes a connection, or nexus, between the evidence and the claimed

invention. The nexus cannot be based on prior art features. The establishment of a nexus is a question of fact. For example, if commercial success of products allegedly embodying the claims is presented as a secondary indicium of non-obviousness, the patent owner must show that such success is due to features of the claimed invention, and not features unrelated to the claimed invention or that already existed in the prior art.

21. It does not appear that Patent Owner has previously offered any secondary considerations of non-obviousness during prosecution, and I am unaware of any offered since that time. However, in my opinion the clear teachings of the prior art herein and discussed in the Request would overcome such "secondary considerations," if any were offered.

C. Claim Construction

22. I am informed that in performing an obviousness analysis, it is necessary to understand the scope of the claims. I am also informed that the first step in an unpatentability analysis, therefore, involves construing the claims. Second, the construed claim language is then compared to the disclosures of the prior art.

23. I am informed that during reexamination, claim terms are to be given the broadest reasonable interpretation consistent with the specification, and limitations in the specification are not to be read into the claims. I am informed that the broadest reasonable interpretation does not mean the broadest possible interpretation, but that instead the meaning must be consistent with the ordinary and customary meaning of the term (unless the term has been given a special definition in the specification), and must be consistent with the use of the claim term in the specification and drawings. I am informed that the broadest reasonable interpretation of the claims must also be consistent with the interpretation that those skilled in the art would reach. I am informed that, for the purposes of claim construction, while expert testimony may be helpful, the testimony from a technical expert is generally less reliable than the patent itself and its prosecution history in determining the meaning of the claim terms, and it is not the role of technical experts to supplant courts and PTAB judges in applying principles of claim interpretation. Accordingly, for the purposes of this proceeding, I have construed the claims in accordance with these requirements.

24. I note that claims 1 and 19 of the '823 Patent use the phrase "wherein the plurality of network device applications and plurality of cloud applications device form **unified**

capabilities. . . ." . '823 Patent (Ex. 1001) at claims 1 and 19. However, the Specification of the '823 Patent does not use the term "unified capabilities", and I have been informed that the term was not specifically addressed during the prosecution history. The only description provided in the Specification of the '823 Patent involving the use of the term "unified" states that "As used in this disclosure in describing the dSDN, a database may be either a single unified system or a distributed system wherein certain database elements are located upon different systems or servers which may be in different physical locations, acting in harmony to appear as one unified database." *Id.* at 4:43-47.

25. Accordingly, in the context of claims 1 and 19, which recites the combination of a plurality of network device applications and a plurality of cloud applications, the broadest reasonable interpretation of the term "unified capabilities", in my opinion, should at least encompass network device applications and cloud applications that, although distributed, act in a unified manner, that is, act in harmony or in concert so as to appear as unified application(s).

26. Claims 1 and 19 of the '823 Patent also recite "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" (the "Enabling Clause"). *Id.* at claims 1 and 19. The '823 Patent only mentions the term "upper layer" on one occasion when the '823 Patent states, with reference to FIG. 4, as follows:

> **FIG. 4 demonstrates an examplary view of the software layers** in the fxOS 302a. Each layer of this software stack is modified and redefined to optimize for embedded networking equipment of wireless and wireline nature. Starting from the bottom of the stack, the kernel also connects to the hardware resources via the device drivers. Examples of such resources are wireless baseband SoC 402, hardware packet processor 404, Wide Area Network (WAN) uplink interface 406, cache and storage 408, and multi-core CPU 410. The OS kernel 412 is enhanced with the FastPath function 414 to accelerate packet processing (by help of hardware features such as Intel's Data Plane Development Kit (DPDK) or pure software enhancements) and other related data path functions. The enhanced kernel with FastPath 412, 414 would enable visibility into the data flows, sniffing the Non-Access Stratum (NAS) signaling (which is the signaling between the UE and the Core Network and the RAN is supposed to be transparent to), policy enforcements, regular packet forwarding functions, collection of flow statistics into Flow Information Base (FIB), and encryption/decryption functions.
>
> Above the kernel, there are **several tools 416**, native daemons 418, native libraries 420 and Hardware Abstraction Layers (HAL) 422. **These tools abstract**

> **out the hardware dependencies for the upper layers and programmers**. The Runtime 424 is an embedded virtual machine capable of securely isolating and executing the applications. System Services 426 are a set of services always running and available to the developers (e.g., timing and messaging services).

Id. at 11:16-12:26 (emphasis added).

27. Additionally, regarding the requirement of being "independent of network device hardware and cloud device hardware", the Specification of the '823 Patent also states on several occasions that "The dSDN also harmonizes application/fxApp deployment across the network independent of the hardware vendor." *Id.* at 2:35-37 and 7:61-62 (also see the Abstract).

28. Accordingly, in my opinion, with respect to the proper interpretation of the Enabling Clause as recited in claims 1 and 19, this clause should be interpreted as encompassing that the implementation of the network device applications and cloud applications in a "unified manner" (interpreted as discussed above) allows for network device applications and cloud device applications to be programmed via APIs. Additionally, the Enabling Clause should be interpreted as encompassing that the aforementioned APIs are "upper layer" APIs in that the APIs constitute or interact with software layers that are abstracted out from or only indirectly in contact with a hardware-interactive layer such as a kernel (or operating system), and therefore independent of or removed from any particular hardware on which the network device applications and cloud applications are implemented.

29. Finally, each of the claims of the '823 Patent are system claims, and are thus appratus claims. However, several of the claims include funtional limitatations as set forth below:

- "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" in claims 1 and 19.
- " an application management portal capable of managing life cycles of the distributed applications" in claim 3.
- "said infrastructure marketplace capable of providing tested and certified distributed applications to the application management portal" in claims 3 and 19.
- "the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said

distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications" in claim 4.

- "an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications" in claim 19.

30. I am informed that the analysis of whether a prior art reference anticipates or renders obvious a given claim can take into account additional considerations when the claim language is functional language rather structural language. Specifically, I am informed that "[A]pparatus claims cover what a device is, not what a device does." MPEP §2114. I am further informed that a claim containing a "recitation with respect to the manner in which a claimed apparatus is intended to be employed does not differentiate the claimed apparatus from a prior art apparatus" if the prior art apparatus teaches all the structural limitations of the claim. *Id.* Additionally, I am informed that if a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation. *Id.*

III. Opinion

A. Level of Skill of a POSITA

31. I was asked to provide my opinion as to the level of skill of a person having ordinary skill in the field of the '823 Patent at the time of the claimed invention, which I have been instructed to assume is June 13, 2013. In determining the characteristics of a hypothetical POSITA, I was told to consider several factors, including the type of problems encountered in the art, the solutions to those problems, the rapidity with which innovations are made in the field, the sophistication of the technology, and the education level of active workers in the field. I also placed myself back in the time frame of the claimed invention and considered the colleagues with whom I had worked at that time.

32. In my opinion, the level of ordinary skill in the art that one would need in order to have the capability of understanding the scientific and engineering principles applicable to the '823 Patent is (i) a Bachelor degree (or higher degree) in electrical engineering, computer science, or computer engineering or equivalent and (ii) at least two years of experience working in network architecture and solutions, communication systems, wireless networks, or mobile wireless devices. Additional industry experience or technical training may offset less formal

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
148

education, while advanced degrees or additional formal education may offset lesser levels of industry experience.

33. Based on my education, training, and professional experience, I am familiar with the level and abilities of a POSITA at the time of the claimed invention. Additionally, I met at least these minimum qualifications to be a person having ordinary skill in the art at the time of the claimed invention. Further, although my qualifications may exceed those of the hypothetical POSITA defined above, my analysis and opinions regarding the '823 Patent have been rendered from the perspective of a POSITA at the time of the claimed invention.

B. Summary of My Opinions

34. I have reviewed the claims of the '823 Patent, the prior art cited in the Request, and the mapping of the prior art charted against the Challenged Claims in Appendix A and Appendix B of the Request.

35. I agree with Requestor that Stewart (U.S. Pub. No. 20120278454) anticipates or renders obvious claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent.

36. I also agree with Requestor that Malik (U.S. Pub. No. 20130054682) anticipates or renders obvious claims 1, 2, 3, 4, 15 and 19 of the '823 Patent.

C. Summary of the '823 Patent

37. The '823 Patent generally relates to "a network architecture that facilitates secure and flexible programmability between a user device and across a network," '823 Patent (Ex. 1001) at 1:14-16, and more specifically to a "Distributed Software Defined Network (dSDN)" that purports to be "an architecture that enables secure and flexible programmability across a network with full lifecycle management of services and infrastructure applications (fxDeviceApp)" that "also harmonizes application deployment across the network independent of the hardware vendor." *Id.* at Abstract and 2:31-39. The disclosure of the '823 Patent is applicable to "many different types of networks," although is specifically disclosed with reference to a "cellular network." *Id.* at 8:62-67.

38. Claims 1 and 19 are the independent claims of the '823 Patent, while claims 2-18 each depend directly or indirectly from independent claim 1. Claims 1 and 19 are reproduced below:

1. A system comprising:

a programmable network device adapted to host a plurality of network device applications;

a programmable cloud device adapted to host a plurality of cloud applications, wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications; and

wherein the plurality of network device applications and plurality of cloud applications device form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware.

19. A system comprising:

a programmable network device adapted to host a plurality of network device applications;

a programmable cloud device adapted to host a plurality of cloud applications, wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications;

wherein the plurality of network device applications and plurality of cloud applications form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware;

an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications; and

an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested distributed applications to the application management portal.

39. Independent claims 1 and 19 are both directed to systems. The system of claim 19 includes all of the elements of the system of claim 1, as well as two additional elements, namely the application management portal and the infrastructure application marketplace. The Specification of the ’823 Patent describes examples of systems as recited in claims 1 and 19. For example, a system that includes the elements recited in both claims 1 and 19 is illustrated in Fig. 3 of the ’823 Patent, as shown below:

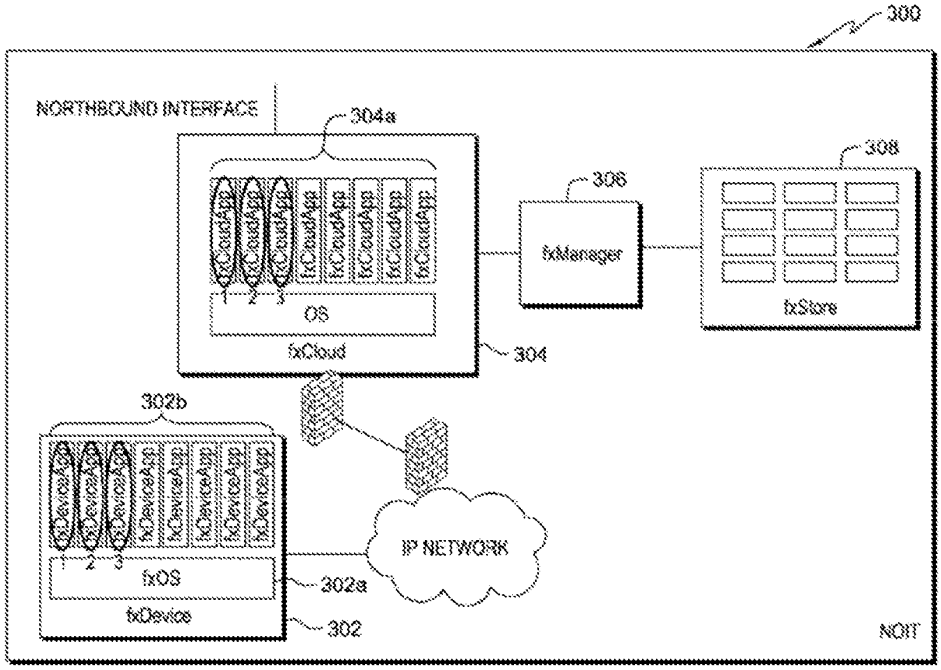


FIG. 3

40. As shown in Fig. 3 of the '823 Patent, "[t]he dSDN system 300 consists of a flexible network device (fxDevice) 302, a flexible cloud platform (fxCloud) 304, an application management portal (fxManager) 306, and an infrastructure application market place (fxStore) 308." '823 Patent (Ex. 1001) at 10:9-13. With respect to Fig. 3, the '823 Patent explains that "[t]he fxDevice 302 may host several independent and securely isolated applications (on top of fxOS 302 a) named here fxDeviceApp 302 b" and that "fxDeviceApp 302 b may have a sister app in the backend cloud infrastructure (i.e., flexible cloud platform 304) referenced here as fxCloudApp 304 a." *Id.* at 10:21-27. The '823 Patent also states that "[t]he fxCloudApp 304 a in the cloud is paired with its fxDeviceApp 302 b in the fxDevice 302" and that they "collectively form a distributed application (dApp or fxApp)." *Id.* at 10:27-31. Thus, in the '823 Patent, "when fxDeviceApp is referenced it shall mean any application that may have software components in the fxDevice, fxCloud, or both." *Id.* at 10:31-33.

41. Further, the specification of the '823 Patent states that "[A]s used in this disclosure in describing the dSDN, a database may be either a single unified system or a distributed system wherein certain database elements are located upon different systems or

servers which may be in different physical locations, acting in harmony to appear as one unified database." *Id*. at 4:43-47.

42. With reference to Fig. 4, reproduced below, which illustrates an examplary view of the software layers in the operating system (fxOS 302a) of the network device 302, the '823 Patent explains that "[a]bove the kernel, there are several tools 416, native daemons 418, native libraries 420 and Hardware Abstraction Layers (HAL) 422" and that "these tools abstract out the hardware dependencies for the upper layers and programmers." *Id*. at 16:38-51.

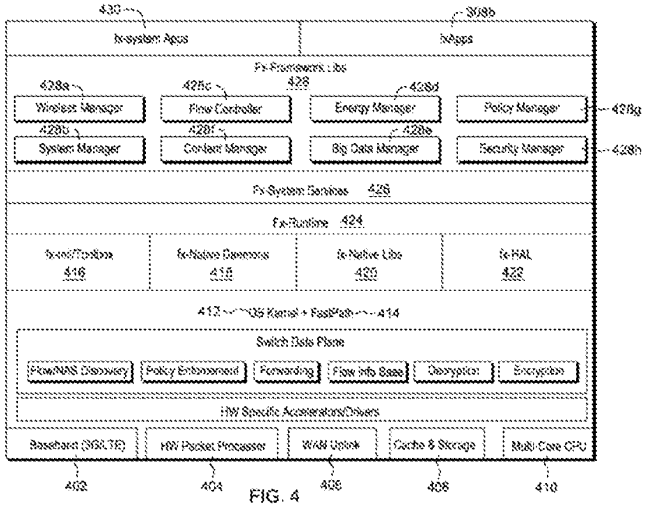


FIG. 4

43. With respect to application programming interfaces (APIs), the Specification of the '823 Patent discloses a flexible Software Development Kit (fxSDK) and explains that:

> The fxSDK is unique in several aspects exemplified as follows. First, since the dSDN applications may have a device component (fxDevice 302) and cloud component (fxCloud 304), the fxSDK allows the developer to develop the components together to simplify the development and testing. Second, the dRMS offers various system level APIs with remote accessibility. In other words, the APIs in the fxDevice 302 could be accessed via the applications in the fxCloud (fxCloudApps) and vice versa.

Id. at 16:38-51. The '823 Patent also discloses an "Extensible API Framework" that "allows the developer to create custom APIs and make it accessible for other applications (in the fxDevice or in the fxCloud)" and that "[i]n turn, the fxCloud could present these APIs using e.g. REST technologies (via the fxCloud Northbound Interface) to the developer outside the dSND system." *Id*. at 28:49-54.

D. Ground 1: Stewart Anticipates the Challenged Claims

1. Overview of Stewart

44. Stewart "generally relates to software application development and deployment on remote devices, and more specifically to systems and methods for configuring or managing gateway devices at various locations." Stewart (Ex. 1004) at ¶[0002]. FIGS. 1A and 1B of Stewart, produced below, illustrate various exemplary systems of Stewart. *Id*. at ¶[0043].

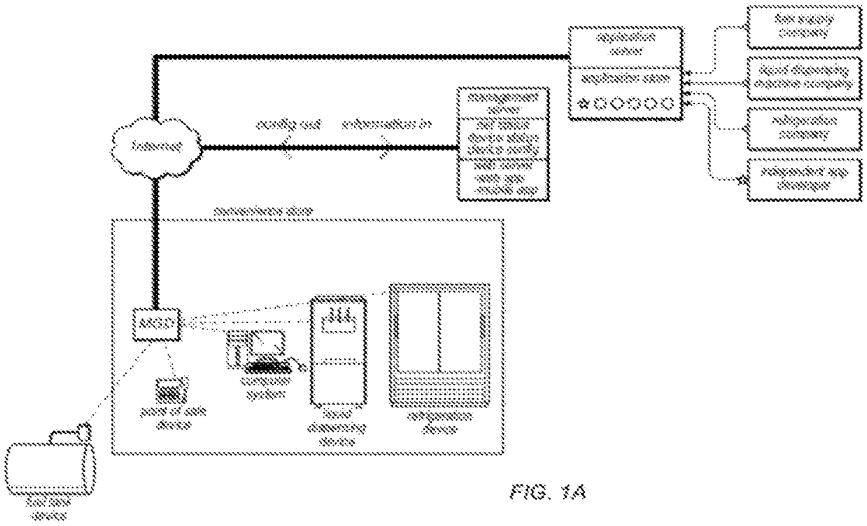


FIG. 1A

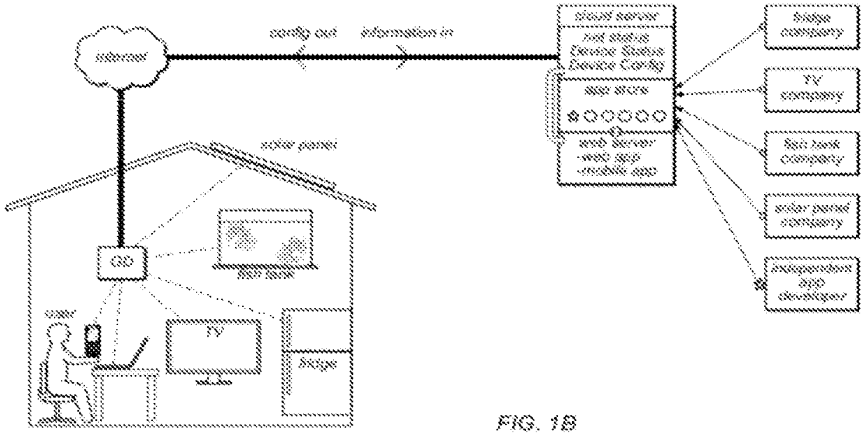


FIG. 1B

45. The systems of Stewart as shown in Figs. 1A and 1B are very similar, except that the system of Fig. 1A is implemented in a merchant location and the system of Fig. 1B is implemented in a home. *Id*. at ¶[0044] and [0076]-[0078].

46. Stewart discloses that its systems include a gateway device (MGD in Fig. 1A and GD in Fig. 1B), which is a device that is "configured to couple to a wide area network, such as the Internet, and also couple (either wired or wirelessly) to one or more devices, and which is configurable with one or more application programs to monitor and/or control the one or more devices." *Id*. at ¶ [0042]. Stewart further discloses that the gateway device may install (and thus host) applications that may be provided by a cloud server in order to manage various devices coupled to the gateway device. *Id*. at Abstract.

47. Stewart discloses that the gateway device is in communication with one or more servers, which may be a cloud server or cloud servers. *Id*. at ¶¶[0054]-[0055] and [0078]. The one or more servers include a management server and an application server, which may be the same server or different servers, and which are illustrated in Fig. 1A as being separate servers and in Fig. 1B as being provided by a single cloud server. *Id*. at ¶¶[0068] and [0078].

48. Various different application developers (or "package developers") may create various different applications that may be uploaded to the application server for purchase by customers (e.g., merchants) who are operating gateway devices. *Id*. at ¶[0012]. The applications may be executed in a distributed fashion, e.g., among the gateway device, the server, and/or other devices, such as user devices. *Id.* at ¶[0016].

49. "A user may be able to configure the gateway device directly (e.g., via a web site hosted by the gateway device or via an application executing on the computer system) or may be able to configure the gateway device over the Internet, via the management server." *Id*. at ¶[0054]. "For example, a manager may browse the application server and select from a plurality of different applications that may be installed and executed by the gateway device and/or the management server (and possibly by devices associated with the convenience store). *Id*. at ¶[0069].

50. Stewart discloses that a software development kit (SDK) and/or application programming interface (API) may be used to allow a developer to develop an application to be stored on the application server that is compatible with the gateway device, the management

server, and/or other devices that may execute portions of the application. *Id*. at ¶¶[0074] and [0105]-[0106].

51. Additionally, with reference to Fig. 6, reproduced below, Stewart discloses that APIs can also be used in relation to developed applications.

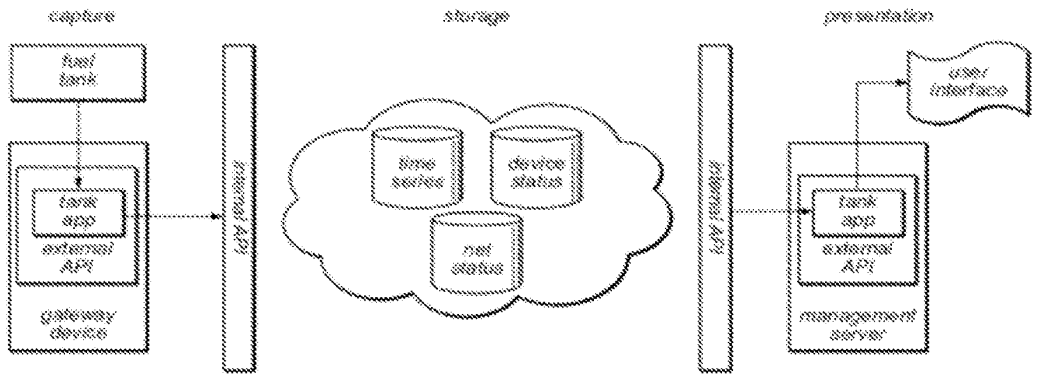


FIG. 6

52. Specifically, Fig. 6 of Stewart shows a tank application that is configured to determine the fuel level in the fuel tank and present the fuel level information in a user interface. *Id*. at ¶¶[0111]. The portion of the tank application executing on the gateway device receives information from the fuel tank using a communication protocol supported by the fuel tank device (e.g., the device monitoring the fuel level in the fuel tank). *Id*. at ¶¶[0112]. The gateway device is configured to receive the reported information from the tank application and provide that information for storage via an internal API. Id. Additionally, the tank application is configured to provide the reported information for storage, e.g., in cloud storage on the Internet, and executes an external API to perform that functionality. *Id*. at ¶¶[0112].

53. Further, a second portion of the tank application may execute on the management server, and may use a presentation external API to request information stored in the cloud storage. *Id*. at ¶[0113]. "Thus, FIG. 6 illustrates an exemplary embodiment of a distributed application using internal and external APIs." *Id.*

54. Stewart is analogous art to the '823 Patent. The field of endeavor of the '823 Patent relates to a network architecture that facilitates "secure and flexible programmability between a user device and across a network…" '823 Patent (Ex. 1001) at Abstract. Further, the '823 Patent relates to a network architecture that includes one or more cloud device (e.g., a cloud

server) that interact with one or more network devices. *See e.g.*, '823 Patent (Ex. 1001) at Abstract and at 10:9-13. The '823 Patent relates to network technology that "is not limited to wireless networks embodiments" but rather "could be applied to many other network types including enterprise, wireline service providers, data centers, and Internet of Things (IoT)." '823 Patent (Ex. 1001) at 7:67-8:3.

55. Stewart similarly relates to networks, and for example provides disclosure regarding how a "gateway device may also be coupled to a server (e.g., a cloud server) over a wide area network…." Stewart (Ex. 1004) at ¶¶[0009]. Stewart also addresses the programming of devices on a network, stating for example that a "default program configuration may include a management application that may be installed and executed by the gateway device and/or the server to allow users to interact with the gateway device…." Stewart (Ex. 1004) at ¶¶[0139]. Because Stewart relates to networks with involve both cloud device(s) and other network device(s) as well as programming of such devices, Stewart is within the field of endeavor of the '823 Patent.

56. Further, Stewart is reasonably pertinent to at least one problem considered by the inventor of the '823 Patent. The '823 Patent concerns addressing problems associated with the inflexibility of configuring a network, for example, stating that "[e]xisting network solutions are built on proprietary hardware and software" and additionally that "[n]etwork Operators & Information Technology (NOIT) can only configure the proprietary network infrastructure provided by vendors and are unable to add new and customized features or capabilities." '823 Patent (Ex. 1001) at 1:22-26. Stewart describes a networked system that achieves enhanced configuration of devices on the network, for example, describing how a "management server may also allow for configuration of the gateway device or premise devices managed by the gateway device in the convenience store. Stewart (Ex. 1004) at ¶¶[0066]. Stewart further describes a network system having a framework that may allow "for easy installation of applications in response to user selection of applications to install." Stewart (Ex. 1004) at ¶¶[0070]. Stewart further recognizes that, in at least some embodiments, it is significant that configurations be achieved that are not tied to specific devices, stating for example that "while the above described applications are specific to individual types (or even models) of premise devices in the convenience store, there may be applications that are not device specific." Stewart (Ex. 1004) at ¶¶[0075]. Thus, Stewart closely relates to the aforementioned problem identified

by the '823 Patent, in that Stewart concerns network systems that, in at least some embodiments, are designed to facilitate flexible configuration of devices associated with the network and, in at least some circumstances in a manner that is independent of specific characteristics of those devices.

2. Application of Stewart to the Challenged Claims

57. Stewart discloses each and every element of claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent for at least the reasons set forth in Appendix A to the Request.

a) '823 Patent at Independent Claims 1 and 19: Stewart Discloses a "*system*"

58. I am informed that the preamble of a claim may not be given patentable weight. However, in the event that the preamble is given patentable weight, Stewart discloses systems as recited in claims 1 and 19 of the '823 Patent. More specifically, Stewart discloses "systems and methods for configuring or managing gateway devices at various locations". Stewart (Ex. 1004) at ¶ [0003]. Figures 1A and 1B of Stewart illustrate "exemplary systems that may implement embodiments described" in Stewart. *Id.* at ¶[0043].

b) '823 Patent at Independent Claims 1 and 19: Stewart Discloses "*a programmable network device adapted to host a plurality of network device applications*"

59. Stewart discloses *a programmable* (e.g., configurable) *network device* (e.g., a gateway device) *adapted to host a plurality of network device applications* (e.g., applications or application programs on the gateway device) as recited in claims 1 and 19 of the '823 Patent.

60. Stewart discloses a system that includes a gateway device that is "configurable with one or more application programs." Stewart (Ex. 1004) at ¶ [0042]. Stewart further discloses that "the gateway device may install [and thus host] applications provided by the cloud server to manage the devices coupled to the gateway device." Stewart (Ex. 1004) at Abstract.

c) '823 Patent at Independent Claims 1 and 19: Stewart Discloses "*a programmable cloud device adapted to host a plurality of cloud applications*"

61. Stewart discloses *a programmable* (e.g., configurable) *cloud device* (e.g., cloud server) *adapted to host a plurality of cloud applications* (e.g., applications or application programs on the cloud server) as recited in claims 1 and 19 of the '823 Patent.

62. Stewart discloses a system that includes a "cloud server" that hosts "applications and may facilitate selection and deployment of these applications on the gateway device." Stewart (Ex. 1004) at ¶¶ [0010]; *See also Id.* at Abstract and ¶¶ [0012], [0067], and [0115].

Indeed, Stewart discloses that its system may include one or more servers, such as an application server and a management server, each of which may be (together or separately), the cloud server. Stewart (Ex. 1004) at ¶¶ ¶¶[0054]-[0055], [0068], [0078], and Figs. 1A and 1B. The cloud server is programmable at least in that (1) applications may be executed on the cloud server, and (2) the cloud server includes an application server and applications may be developed and uploaded to the cloud server. Stewart (Ex. 1004) at ¶¶ [0011]-[0012], [0067], and Figs. 1A and 1B.

d) '823 Patent at Independent Claims 1 and 19: Stewart Discloses "*wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications*"

63. Stewart discloses *wherein the plurality of network device applications* (e.g., applications or application programs on the gateway device) *and the plurality of cloud applications* (e.g., applications or application programs on the cloud server) *are in secure communication with each other to form distributed applications* (e.g., executing applications "in a distributed fashion") as recited in claims 1 and 19 of the '823 Patent.

64. Stewart discloses generally that the applications described therein may be executed in a distributed fashion, and thus constitute distributed applications, and also provides a specific example with respect to Fig. 6 of a distributed application. Stewart (Ex. 1004) at ¶¶ [0016], [0111], [0126] and Fig. 6. Stewart explicitly discloses distributed applications, including the application components executed by the gateway device and the application components executed by the cloud server. See Stewart (Ex. 1004) at ¶¶ [0016], [0111], [0126]. Stewart provides a specific example of such a distributed application with respect to its Figure 6, which is "a tank application that is configured to determine the fuel level in the fuel tank and present the fuel level information in a user interface" where "the tank application is executed in a distributed fashion, by both the gateway device (on the capture side) and the management server (on the presentation side)." *Id.* at ¶[0111] and Fig. 6. As discussed with respect to elements 1[a] and 1[b] above, the applications on the gateway device correspond to the *network device applications*, and applications on the management server (cloud server) correspond to *cloud applications*.

65. Stewart also discloses that the gateway device is in secure communication with the server, thus resulting in the applications on the gateway device being in secure communication with the applications on the cloud server. Stewart (Ex. 1004) at ¶¶ [0051], [0085], [0227]-[0230]. For example, "the gateway device may include security software or

circuitry, e.g., to provide a firewall for the Internet connection, perform communication encryption, protect credit card information provided over the network connection, etc." Stewart at ¶[0051]. Additionally, "an application may request to communicate with a server or other entity over a network, such as the Internet. The gateway device may determine whether the communication is allowed using a security policy". Stewart at ¶[0228].

e) '823 Patent at Independent Claims 1 and 19: Stewart Discloses "*wherein the plurality of network device applications and plurality of cloud applications device [sic] form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware*"

66. Stewart discloses *wherein the plurality of network device applications* (e.g., applications or application programs on the gateway device) *and plurality of cloud applications device* [*sic*] (e.g., applications or application programs on the cloud server) *form unified capabilities* (e.g., programs on the gateway device and programs on the cloud server work together in a unified manner to perform the full functionality of the applications) *enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications* (e.g., the use of APIs to program the applications on both the gateway device and cloud server) *independent of network device hardware and cloud device hardware* (e.g., developed applications are compatible with the gateway device, the management server, and/or other devices that may execute portions of the application).

67. I note that there is a slight difference in the wording of this limitation between claims 1 and 19 of the '823 Patent. Specifically, claim 1 recites "cloud applications device" while claim 19 recites "cloud applications". However, the word "device" after the recitation of "cloud applications" in claim 1 appears to be a typographical error that can, and should be, appropriately disregarded.

68. I also note that this limitation recites the functional element of "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware." I am informed that the recitation of such functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the

prior art. However, to the extent that this limitation is given patentable weight, Stewart discloses enabling the use of APIs in its systems.

69. Regarding that the applications on the network device and cloud device form *unified capabilities*, Stewart explicitly discloses distributed applications, including the application components executed by the gateway device (corresponding to the "plurality of network device applications" recited in claims 1 and 19 of the '823 Patent) and the application components executed by the cloud server (corresponding to the "plurality of cloud applications" recited in claims 1 and 19 of the '823 Patent). *See Stewart* (Ex. 1004) at ¶¶ [0016], [0111], [0126]. Stewart also discloses that the applications can be distributed applications. *Id*. at ¶¶ [0016], [0111], [0126] and Fig. 6. Stewart provides a specific example of such a distributed application with respect to its Figure 6, which is "a tank application that is configured to determine the fuel level in the fuel tank and present the fuel level information in a user interface" where "the tank application is executed in a distributed fashion, by both the gateway device (on the capture side) and the management server (on the presentation side)." *Id*. at ¶[0111] and Fig. 6.

70. While Stewart does not use the term "unified capabilities", I am informed that identity of terminology is not required. Moreover, as a POSITA would understand, because the applications are distributed applications, the applications on the cloud server and the applications on the gateway device necessarily form unified capabilities, because that is the nature and purpose of distributed applications.

71. Further, it is clear from the teachings of Stewart that the distributed applications of Stewart work together in a unified manner to perform the full functionality of the applications, thus forming *unified capabilities*.

72. Regarding the element of "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware", Stewart also discloses that developers can use an application programming interface (API) to develop an application that is compatible with and can be installed on the gateway device, the management server, and/or other devices that may execute portions of the application. Stewart (Ex. 1004) at ¶¶ [0011], [0067], [0071]-[0072], [0074], [0106], [0118]-[0119]. Additionally, as shown in Figure 6 of Stewart, the distributed applications of Stewart

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
160

can use APIs to facilitate the performance of the applications, and to program the gateway and server portions of the applications in the form of receiving and transferring information between them. Stewart (Ex. 1004) at ¶¶ [0111]-[0113] and Fig. 6.

73. I am informed that if a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation.

74. Thus, while Stewart does not explicitly state that the applications are programmed by upper layer APIs independent of network device hardware and cloud device hardware, Stewart does teach that the developed applications "are compatible with the gateway device, the management server, and/or other devices that may execute portions of the application". Stewart (Ex. 1004) at ¶¶ [0106]. As would be understood and appreciated by a POSITA, it is the function and purpose of APIs to allow for programing in a manner that is independent of device hardware. It is therefore an inherent characteristic of an API that it can be employed to program applications in a manner that is removed from or "independent of" (and in that sense, in an upper layer with respect to) "hardware."

75. Accordingly, Stewart either explicitly or inherently discloses "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware."

f) '823 Patent at Independent Claim 19 and Dependent Claim 3: Stewart Discloses "*an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications*" and "*an application management portal capable of managing life cycles of the distributed applications*"

76. Stewart discloses *an application management portal* (e.g., a management server) *capable of managing life cycles* (e.g., installing, modifying, and removing or replacing applications) *of the plurality of network device applications* (e.g., applications or application programs on the gateway device) *and plurality of cloud applications* (e.g., applications or application programs on the cloud server) as recited in claim 19, and "*an application management portal* (e.g., a management server) *capable of managing life cycles* (e.g., installing, modifying, and removing or replacing applications) *of the of the distributed applications* (e.g., applications or application programs)" as recited in dependent claim 3. Although the

terminology of this limitation varies with respect to what the object of the lifecycle management is, the teachings of Stewart are equally applicable to each limitation because the "distributed applications" as recited in claim 3 include the "plurality of network device applications and plurality of cloud applications" as recited in claim 19.

77. I note that the elements of being "capable of managing life cycles of the plurality of network device applications and plurality of cloud applications" and "capable of managing life cycles of the of the distributed applications" are each functional language, which may not be given patentable weight. However, to the extent that these limitation are given any patentable weight, they are disclosed either expressly or inherently in Stewart.

78. Additionally, to the extent that Stewart does not explicitly disclose the management server managing life cycles of the plurality of network device applications and plurality of cloud applications, it is clear based on the disclosure of Stewart that the management server of Stewart would be capable of such management.

79. Stewart discloses a management server and an application server that are in communication with the gateway device, and which may be the same server or different servers, and may each (or together) be the cloud server. Stewart (Ex. 1004) at ¶¶ ¶¶[0054]-[0055], [0068], [0078]. and Figs. 1A and 1B.

80. As to *managing life cycles of the of the distributed applications* in particular, Stewart discloses that the management server can be used to "perform control actions" and "configure the gateway device," including installing, modifying, and removing or replacing applications. Stewart (Ex. 1004) at ¶[0054]. Stewart discloses that the management server can be used to select one or more applications to be installed and executed by "the gateway device and/or the management server (and possibly by devices associated with the convenience store)". Stewart (Ex. 1004) at ¶¶[0066]-[0069]. For example, Stewart discloses that "a manager may browse the application server and select from a plurality of different applications that may be installed." Stewart (Ex. 1004) at ¶[0069]. Stewart also generally discloses that the installed applications can be modified using the management server. Stewart (Ex. 1004) at ¶¶[0066]. Stewart also discloses that some applications may not be removable (thus indicating that other applications are removable), but that such applications can be replaced. Stewart (Ex. 1004) at ¶¶[0223]. Accordingly, Stewart discloses the management server managing life cycles of the plurality of network device applications and plurality of cloud applications.

g) '823 Patent at Independent Claim 19 and Dependent Claim 3: Stewart Discloses "*an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested (and certified) distributed applications to the application management portal*"

81. Stewart discloses *an infrastructure application marketplace* (e.g., application server operating as an application store) *in communication with* (e.g., the application server operating as an application store communicates over the Internet with the management server) *the application management portal* (e.g., management server)*, said infrastructure marketplace capable of providing tested (and certified) distributed applications to the application management portal* as recited in independent claim 19 and dependent claim 3.

82. I note that claim 3 recites providing "tested and certified" distributed applications, while claim 19 recites providing "tested" distributed applications. However, the difference does not provide any distinction, and the same disclosure in Stewart is applicable to both terms.

83. I note that the element of being "capable of providing tested and certified distributed applications to the application management portal" is functional language that may not be given patentable weight. However, to the extent that this limitation is given any patentable weight, it is disclosed either expressly or inherently in Stewart.

84. Moreover, to the extent that Stewart does not disclose this element though explicit disclosure of providing such applications, based on the disclosure of Stewart discussed below, it is clear that the application store of Stewart would be "capable of" providing such applications.

85. Stewart discloses that "applications may be developed and uploaded to a server, referred to as an application server. The application server may execute software to operate as an 'application store'." Stewart (Ex. 1004) at ¶[0012]. Stewart also discloses a website with respect to which a user may log on and choose from a plurality of applications for installation. *Id*. at [0013]. Applications can be selected manually or automatically from the application store and be installed and executed by the gateway device and/or the management server (and by other devices associated with the gateway device). *Id*. at ¶[0013]; *See also Id*. at ¶¶[0067]-[0069], and [0116] and Figs. 1A and 1B.

86. As to the *infrastructure application marketplace* being *in communication with the application management portal*, Stewart discloses that the application server and the management server can both be cloud servers and that they communicate with each other via the

Internet, and even that "The application server may be the same server as the management server". Stewart (Ex. 1004) at ¶[0068], Fig. 1A and Fig. 1B. Stewart also discloses that applications selected in the application store are then installed by the management server. *Id*. at ¶[0069]..

87. As to *said infrastructure marketplace capable of providing tested and certified distributed applications to the application management portal*, Stewart also discloses that the applications can be tested, both by the developer and by an administrator of the application server, prior to being made available in the application store. Stewart (Ex. 1004) at ¶¶[0106] and [0108]. For example, referring to its Figure 5, Stewart discloses a method for developing an application that includes an approval process in which "an administrator of the application server (or other authorized user) may review the source code, test execution of the application, etc." prior to the application being made available for installation. Stewart (Ex. 1004) at ¶[0108]. Such an approval process would constitute both testing and certifying of the applications. After the approval process, the tested application is then made available via the application server operating as an application store for installation on a device. Stewart (Ex. 1004) at ¶[0110]. Accordingly, in view of Stewart's explicit disclosure of testing an application, Stewart's application server is *capable of providing tested and certified distributed applications.*

h) '823 Patent at Dependent Claim 2: Stewart Discloses "*wherein the programmable network device and the programmable cloud device each include an access network interface unit configured to send and receive communications and a processor with a memory associated with the access network interface unit*"

88. Stewart discloses *wherein the programmable* (e.g., configurable) *network device* (e.g., gateway device) *and the programmable* (e.g., configurable) *cloud device* (e.g., cloud server) *each include an access network interface unit configured to send and receive communications* (e.g., suitable hardware and/or firmware to permit communication over the Internet) *and a processor* (e.g., processor) *with a memory* (e.g., memory or cloud storage) *associated with the access network interface unit* as recited in dependent claim 2 of the '823 Patent.

89. Stewart discloses that the "gateway device may be in communication with one or more servers… over the Internet connection." Stewart (Ex. 1004) at ¶[0055]; *See also Id.* at ¶¶[0017], and Figs. 3 and 4A. Accordingly, Stewart teaches that the gateway device and the cloud server each include an access network interface, since such a component would be

necessary in order to connect to and communicate via the Internet. Additionally, the '823 Patent discloses that the network interface "may include any suitable hardware and/or firmware" such as "a wired network adapter, a wireless network adapter, a telephone modem, and/or a wireless modem." '823 Patent at 9:41-51. Likewise, Stewart discloses that the gateway device may include a modem or other suitable hardware or firmware to provide an Internet connection. Stewart (Ex. 1004) at ¶¶[0050], [0094] and Fig. 4A. Stewart also discloses that the gateway device and the cloud server each have a memory associated with the access network interface. For example, Stewart discloses that the gateway device may include "circuitry such as an ASIC or FPGA and/or a processor and memory storing program instructions to provide Internet connection", Stewart (Ex. 1004) at ¶[0050], and that the gateway device includes both communication logic and applications logic, each of which include memory. Stewart (Ex. 1004) at ¶¶[0094], [0096]. [0098]-[0099] and Figs. 4A and 4B. Stewart also discloses that the cloud server includes cloud based storage, which is memory, which, among other things, may store information received from the gateway device via the Internet. Stewart (Ex. 1004) at ¶[0085] and Fig. 3.

i) '823 Patent at Claim 4: Stewart Discloses "*wherein the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications*"

90. Stewart discloses that *the application management portal* (e.g., management server) *is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace* (e.g., application server operating as an application store); *testing said distributed applications prior to deployment; provisioning said distributed applications* (e.g., management server installing applications)*; and deprovisioning said distributed applications* as recited in claim 4 of the '823 Patent.

91. I am informed that claim language listing alternatives in the format set forth in claim 4 of the '823 patent is anticipated if at least one of the listed alternatives is disclosed in the prior art.

92. I note that the element of being "capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications" is functional language, which may not be given any

patentable weight. However, To the extent that this limitation is given any patentable weight, it is disclosed either expressly or inherently in Stewart. Additionally, to the extent that Stewart does not disclose this element though explicit disclosure of providing such applications, based on the disclosure of Stewart discussed below, it is clear that the application store of Stewart would be "capable of" providing such applications.

93. Stewart discloses that applications may be developed and uploaded to the application server, and that the management server may obtain and execute one or more of the applications. Stewart (Ex. 1004) at ¶¶ [0012]-[0013] and [0067]-[0068]. Applications can be selected manually or automatically from the application store and be installed and executed by the gateway device and/or the management server (and by other devices associated with the gateway device). Id. at ¶[0013]. Thus, the management server of Stewart is capable of both receiving new applications from the application store and provisioning said distributed applications.

j) '823 Patent at Dependent Claim 8: Stewart Discloses "*a distributed notification service wherein notice of a predetermined event is sent from at least one of the plurality of network device applications to at least one of the plurality of cloud applications*"

94. Stewart discloses *a distributed notification service* (e.g., gateway device sending messages to the management server) *wherein notice of a predetermined event* (e.g., messages indicating specific alarms or events associated with premise devices) *is sent from at least one of the plurality of network device applications* (e.g., applications or application programs on the gateway device, such as the "first portion" of the example fuel tank application) *to at least one of the plurality of cloud applications* (e.g., applications or application programs on the cloud server, such as the "second portion" of the example fuel tank application) as recited in claim 8 of the '823 Patent.

95. Stewart discloses that the management server executes applications, and that once the gateway device is configured with one or more applications it may provide information reported by the premise devices to the management server. Stewart (Ex. 1004) at ¶¶[0016], [0067]. For example, the gateway device may send to the management server messages indicating specific alarms or events associated with the premise devices managed by the gateway device. Stewart (Ex. 1004) at ¶[0057].

96. Stewart discloses an example of such communication between the gateway device and the management server in which a distributed application includes "a first portion (provided to the gateway device) to adequately perform management of the fuel tank device" and "a second portion for user interaction" where "the second portion of the application may be executed by the server to provide the received reported information to various other devices for presentation, provide a user interface (e.g., a graphical user interface or "GUI") to control or modify settings associated with the fuel tank device, provide a user interface to set up fuel level thresholds (for when the fuel tank should be refilled), set up contact information associated with the fuel tank (e.g., whom to contact when the fuel level is below a threshold), purchase order information for requisitioning more fuel when the fuel tank is depleted, etc." Stewart at ¶[0072]. As noted in Stewart, the server can be a cloud server. Stewart at ¶[0068]. Accordingly, in the example above, the first portion of the distributed application on the gateway device sends information to the second portion of the distributed application on the cloud server, such as notifications regarding fuel level.

k) '823 Patent at Dependent Claim 15: Stewart Discloses "*a distributed content service wherein contents generated by the at least one of the plurality of network device applications are shared with at least one of the plurality of cloud applications*"

97. Stewart discloses *a distributed notification service* (e.g., gateway device sending messages to the management server messages) *wherein contents* (e.g., information) *generated by the at least one of the plurality of network device applications* (e.g., applications or application programs on the gateway device, such as the "first portion" of the example fuel tank application)) *are shared with at least one of the plurality of cloud applications* (e.g., applications or application programs on the cloud server, such as the "second portion" of the example fuel tank application) as recited in claim 15 of the '823 Patent.

98. Stewart discloses that the management server executes applications, and that once the gateway device is configured with one or more applications it may provide information reported by the premise devices to the management server. Stewart (Ex. 1004) at ¶¶[0016], [0067]. For example, the gateway device may send to the management server messages comprising information gathered from each of (or at least a subset of) the plurality of premise devices managed by the gateway device. Stewart (Ex. 1004) at ¶[0057].

99. Stewart discloses an example of such communication between the gateway device and the management server in which a distributed application includes "a first portion (provided to the gateway device) to adequately perform management of the fuel tank device" and "a second portion for user interaction" where "the second portion of the application may be executed by the server to provide the received reported information to various other devices for presentation, provide a user interface (e.g., a graphical user interface or "GUI") to control or modify settings associated with the fuel tank device, provide a user interface to set up fuel level thresholds (for when the fuel tank should be refilled), set up contact information associated with the fuel tank (e.g., whom to contact when the fuel level is below a threshold), purchase order information for requisitioning more fuel when the fuel tank is depleted, etc." Stewart at ¶[0072]. As noted in Stewart, the server can be a cloud server. Stewart at ¶[0068]. Accordingly, in the example above, the first portion of the distributed application on the gateway device sends information to the second portion of the distributed application on the cloud server, such as content regarding fuel level.

E. Ground 2: Stewart Renders Obvious the Challenged Claims

100. Because Stewart discloses each and every element of claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent for at least the reasons described above, and as set forth in Appendix A to the Request, Stewart also renders obvious those claims.

101. However, with respect to claims 1 and 19, to the extent that Stewart does not explicitly or inherently disclose "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware," these elements are rendered obvious based on Stewart, particularly in view of the knowledge of a POSITA as of June 13, 2023.

1. Overview of the Knowledge of a POSITA

102. The use of application programming interfaces (APIs) to program applications independently of device hardware has been known for decades.

103. Generally, APIs provide a set of rules or protocols that enable software applications to communicate with each other to exchange data, features and functionality independently of the hardware implementation of those applications.

104. The term "API" was introduced at least as early as 1974, in a paper called "The Relational and Network Approaches: Comparison of the Application Programming Interface."

105. With the growth of the Internet in the 1990's, APIs were used as a way for applications to exchange data. Prior to May 30, 1995, Microsoft had developed a software model referred to as WOSA, which was defined for use in the Windows programming environment. The WOSA model isolated "application programmers from the complexities of programming to different service providers by providing an API layer that is independent of an underlying hardware or service." U.S. Patent No. 5691897 at 2:55-58.

106. Accordingly, the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware) was well within the knowledge of a POSITA prior to the June 13, 2023 priority date of the '823 Patent.

2. Stewart in View of the Knowledge of a POSITA

107. As discussed above, a POSITA as of June 13, 2023 would have known about and understood the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware).

108. As discussed in detail in Appendix A, Stewart explicitly discloses the use of APIs to program the plurality of network device applications and plurality of cloud applications as recited in independent claims 1 and 19 of the '823 Patent. Appendix A at pp. 10-16. However, Stewart does not explicitly state that the APIs are upper layer APIs or that the programming is independent of network device hardware and cloud device hardware. To the extent that those elements are not deemed to be inherent, they would have been obvious based on the teachings of Stewart in view of the knowledge of a POSITA as of June 13, 2023.

109. Specifically, a POSITA as of June 13, 2023 would have known about and understood the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware). Accordingly, a POSITA as of June 13, 2023 would have understood that the APIs of Stewart are "upper layer" APIs within the meaning of the '823 Patent, and that they would be enabled "to program the plurality of network device applications and plurality of cloud applications independent of

network device hardware and cloud device hardware" as recited in independent claims 1 and 19 of the '823 Patent.

F. Ground 3: Malik Anticipates Claims 1, 2, 3, 4, 15 and 19 of the '823 Patent

1. Overview of Malik

110. Malik generally relates to "methods and systems for providing mobile device management services and more specifically providing deployment services for deploying software modules to a plurality of managed endpoints." Malik (Ex. 1005) at ¶[0002]. Malik teaches a mobile device management (MDM) platform that can be used to allow an organization to selectively add or remove applications/feature modules on-demand. Malik (Ex. 1005) at ¶[0030]. Malik teaches that "deployment of applications and data to endpoints can be facilitated by a network-based solution" using "a cloud-based approach". Malik (Ex. 1005) at ¶[0030]. With reference to Fig. 1 of Malik, reproduced below:

> FIG. 1 shows a high level architecture of the deployment and distribution system as part of an MDM platform. System 100 contains server components and endpoint components. An application server 102 provides server functionality as described herein. This application server can be a SaaS application server providing server functionality in a cloud-based approach, which include a plurality of servers hosting the application or components thereof. Agents 104 on endpoints 105 provide core MDM functionality including allowing actions to be executed on the endpoints and allowing data and status information to be returned to the server. Agents 104 can include a plurality of endpoint applications that provide agent functionality. SaaS application server 102 may transmit actions and modules across a link 106, which may traverse a network, such as the Internet. Agents 104 may receive these modules and actions and install modules or execute actions accordingly. Agents 104 can also transmit data to the application server across link 108 which may also traverse a network, such as the Internet. This allows agents to report back status of deployments and data collected or generated by the deployed feature modules to application server 102.

Malik (Ex. 1005) at ¶[0057].

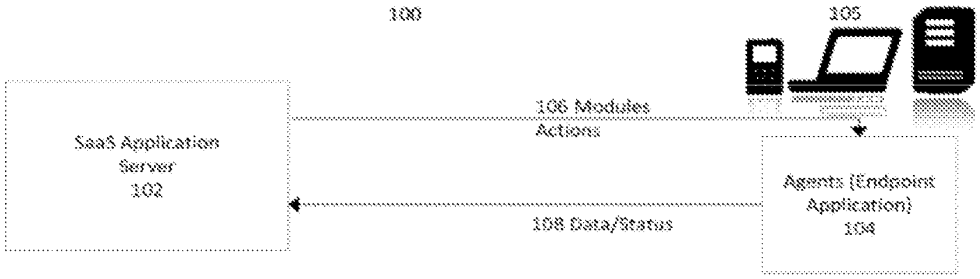


FIG. 1

111. The endpoint devices in Malik "can include any endpoint devices suitable for accessing or interacting with corporate resources" such as "PCs, laptops, mobile devices, smart phones, tablets, network clients, or the like." Malik (Ex. 1005) at ¶[0035]. Endpoint devices have a processor, memory, and a communication circuit to allow communication with a network, either wirelessly or via a network cable. Id. Modules and software components, such as agents, are stored in non-transient memory on the endpoint. Id.

112. Figures 2 and 3 of Malik, reproduced below illustrate system and flow diagrams of "exemplary server components for deploying feature modules to endpoint devices" (Fig. 2) and "exemplary client components for deploying feature modules to endpoint devices" (Fig. 3).

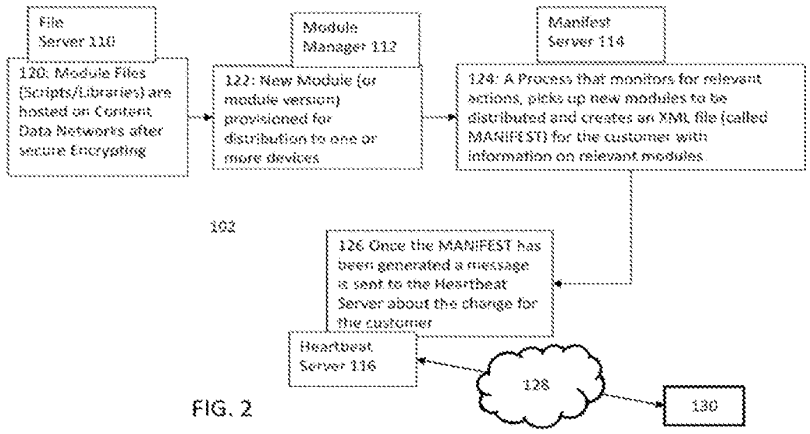


FIG. 2

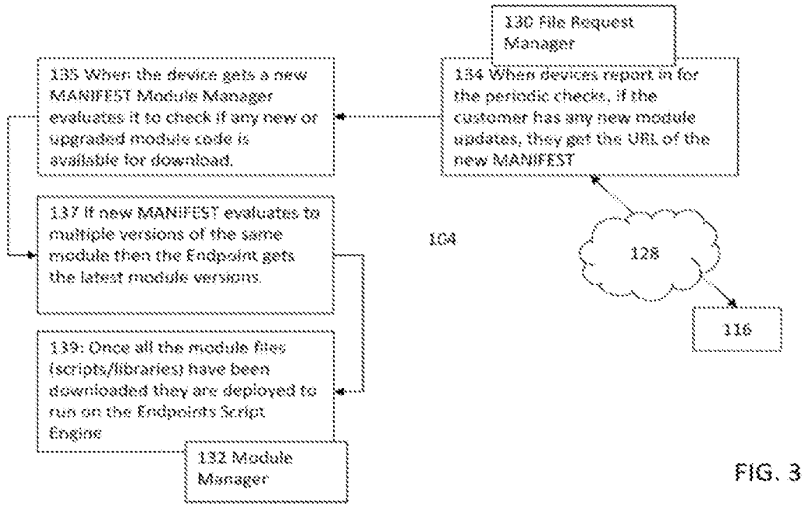


FIG. 3

113. As shown in Fig. 2, the server of Malik includes "a file server configured to host a plurality of feature modules, a module manager configured to maintain the plurality of feature modules, and a manifest server configured to maintain a plurality of manifests, wherein each manifest conveys the feature module components that at least one endpoint should have installed." Malik (Ex. 1005) at ¶[0012]. As shown in Figs. 2 and 3, "in response to a changed module component identified in the manifest, module manager 132 (residing on the endpoint device) can communicate with file server 110 across a network, such as Internet, to download these modules and utilize encryption keys stored in the manifest to decrypt the contents." Id. at ¶[0078].

114. Malik also discloses that its MDM system can be implemented with a plurality of servers hosting the applications or components thereof, resulting in distributed applications, to provide scalability. Malik at ¶¶[0038], [0057], and [0062]. Malik further discloses that software components that reside on and endpoint gather data that is then uploaded to the server components, thus disclosing applications distributed between the endpoint and the server. Malik at ¶¶[0062] and [0087].

115. Malik teaches that developers of feature modules can take advantage of APIs as part of Malik's MDM platform, to allow reporting of information gathered by feature modules. Malik at ¶¶ [0084], [0088], and Fig. 4. Malik further discloses use of a web based dashboard and

also that developers can define and configure modules for deployment by use of a dashboard. Malik at ¶¶[0094] and [0099].

116. Malik is analogous art to the '823 Patent. The field of endeavor of the '823 Patent relates to a network architecture that facilitates "secure and flexible programmability between a user device and across a network…" '823 Patent (Ex. 1001) at Abstract; Further, the '823 Patent relates to a network architecture that includes one or more cloud device (e.g., a cloud server) that interact with one or more network devices. See e.g., '823 Patent (Ex. 1001) at Abstract and at 10:9-13; The '823 Patent relates to network technology that "is not limited to wireless networks embodiments" but rather "could be applied to many other network types including enterprise, wireline service providers, data centers, and Internet of Things (IoT)." '823 Patent (Ex. 1001) at 7:67-8:3; Malik similarly relates to networks, for example, as shown in Figure 1 thereof, which shows a system that "contains server components and endpoint components." Malik (Ex. 1005) at ¶¶[0057]; Malik also addresses the programming of devices on a network, stating for example that the "deployment of applications and data to endpoints can be facilitated by a network-based solution" and that, "[i]n some embodiments, a cloud-based approach is used…." Malik (Ex. 1005) at ¶¶[0030]; Because Malik relates to networks involving server components and other components and that involve a cloud-based approach, as well as relates to the programming of network devices, Malik is within the field of endeavor of the '823 Patent.

117. Further, Malik is reasonably pertinent to at least one problem considered by the inventor of the '823 Patent. The '823 Patent concerns addressing problems associated with the inflexibility of configuring a network, for example, stating that "[e]xisting network solutions are built on proprietary hardware and software" and additionally that "[n]etwork Operators & Information Technology (NOIT) can only configure the proprietary network infrastructure provided by vendors and are unable to add new and customized features or capabilities." '823 Patent (Ex. 1001) at 1:22-26. As already discussed above, Malik relates to a network system having "server components and endpoint components." Malik (Ex. 1005) at ¶¶[0057].

118. Further, Malik discloses a system that is directed toward the deployment of applications/feature modules, where the system particularly includes agents that are "installed on endpoint devices, where the "agents can facilitate installation of applications/feature modules on-demand." Malik (Ex. 1005) at ¶¶[0030]. Malik further describes at least some embodiments the envision the implementation of genericized applications, stating for example that in some

embodiments "certain feature modules may utilize the reporting mechanisms shown in FIGS. 4-5 to report back installation status, allowing feature modules include generic applications that may or may not be related to MDM functionality" and also that in some embodiments "feature modules may wrap basic reporting features around non-MDM applications to allow basic reporting of status of the nation and installation to server components." Malik (Ex. 1005) at ¶¶[0084]. Thus, Malik closely relates to the aforementioned problem identified by the '823 Patent, in that Malik concerns network systems that, in at least some embodiments, are designed to facilitate flexible configuration of devices associated with the network and, in at least some circumstances in a genericized manner.

2. Application of Malik to Claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent

119. Malik discloses each and every element of claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent for at least the reasons set forth in Appendix B to the Request.

a) '823 Patent at Independent Claims 1 and 19: Malik Discloses a "*system*"

120. I am informed that the preamble of a claim may not be given patentable weight. However, in the event that the preamble is given patentable weight, Malik discloses systems as recited in claims 1 and 19 of the '823 Patent. More specifically, Malik discloses "methods and systems for providing mobile device management services and more specifically providing deployment services for deploying software modules to a plurality of managed endpoints." Malik (Ex. 1005) at ¶ [0002]; *See also* Malik at Abstract.

b) '823 Patent at Independent Claims 1 and 19: Malik Discloses "*a programmable network device adapted to host a plurality of network device applications*"

121. Malik discloses *a programmable network device* (e.g., endpoint) *adapted to host a plurality of network device applications* (e.g., feature modules installed on the endpoint device) as recited in claims 1 and 19 of the '823 Patent.

122. Malik discloses a system that includes "deployment of software modules from a file server to agents on endpoints". Malik (Ex. 1005) at Abstract. The endpoints are programmable at least in that the agents receive, install and execute feature modules. *Id.* at Abstract, ¶[0057] and Fig. 1. The feature modules can each be an "executable application, script, or data pack that may be selectively deployed to one or many endpoints, providing a specific function in addition to the core functionality of the MDM system", and may be for any purpose. *Id.* at ¶[0032]; *See also Id.* at ¶¶[0030], and [0036]. Feature modules may themselves be

applications, or may be modules added to existing applications. Malik (Ex. 1005) at Abstract and ¶[0033].

c) '823 Patent at Independent Claims 1 and 19: Malik Discloses "*a programmable cloud device adapted to host a plurality of cloud applications*"

123. Malik discloses *a programmable cloud device* (e.g., server) *adapted to host a plurality of cloud applications* (e.g., feature modules hosted by the server) as recited in claims 1 and 19 of the '823 Patent.

124. Malik discloses Malik discloses a server, or plurality of servers, that are implemented in a cloud based approach. Malik (Ex. 1005) at ¶¶ [0038] and [0048]. The server includes a "file server configured to host a plurality of feature modules." *Id.* at ¶[0008]; *See also Id.* at ¶¶ [0012], [0057], [0087], and [0092].

d) '823 Patent at Independent Claims 1 and 19: Malik Discloses "*wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications*"

125. Malik discloses *wherein the plurality of network device applications* (e.g., feature modules installed on the endpoint device) *and the plurality of cloud applications* (e.g., modules hosted by the server) *are in secure communication with each other to form distributed applications* (e.g., system deployed on a plurality of servers on a dynamic basis) as recited in claims 1 and 19 of the '823 Patent.

126. Malik discloses agents on the endpoint devices "for downloading updates in a secure manner from any suitable server resources." Malik (Ex. 1005) at ¶[0047]. Malik also discloses that its MDM system can be implemented with a plurality of servers hosting the applications or components thereof, resulting in distributed applications, to provide scalability. *Id.* at ¶¶[0038], [0057], and [0062]. Malik further discloses that software components that reside on and endpoint gather data that is then uploaded to the server components, thus disclosing applications distributed between the endpoint and the server. *Id.* at ¶¶[0062] and [0087].

e) '823 Patent at Independent Claims 1 and 19: Malik Discloses "*wherein the plurality of network device applications and plurality of cloud applications device [sic] form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware*"

127. Malik discloses *wherein the plurality of network device applications* (e.g., feature modules installed on the endpoint device) *and plurality of cloud applications device* [*sic*] (e.g.,

modules hosted by the server) *form unified capabilities* (e.g., programs on the endpoints and programs on the cloud server work together in a unified manner to perform the full functionality of the applications) *enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device* (e.g., endpoint) *hardware and cloud device* (e.g., server) *hardware* as recited in claims 1 and 19 of the '823 Patent.

128. I note that there is a slight difference in the wording of this limitation between claims 1 and 19 of the '823 Patent. Specifically, claim 1 recites "cloud applications device" while claim 19 recites "cloud applications". However, the word "device" after the recitation of "cloud applications" in claim 1 appears to be a typographical error that can, and should be, appropriately disregarded.

129. I also note that this limitation recites the functional element of "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware." I am informed that the recitation of such functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the prior art. However, to the extent that the functional language is given patentable weight, Malik discloses enabling the use of APIs in its systems.

130. Regarding that the applications on the network device and cloud device form *unified capabilities*, Malik explicitly discloses both applications/feature modules being deployed on endpoints and also an application server hosting applications and providing server functionality in a cloud-based approach. Malik (Ex. 1005) at ¶¶[0030], [0057], [0087], and [0092]. Malik further explicitly discloses that these are distributed applications, including a "server providing server functionality in a cloud-based approach, which include a plurality of servers hosting the application or components thereof" as well as "[a]gents 104 on endpoints 105" that "provide core MDM functionality including allowing actions to be executed on the endpoints and allowing data and status information to be returned to the server." Malik (Ex. 1005) at ¶[0057]; see also *Id.* at ¶¶[0038], [0062] and [0087]. Indeed, Malik discloses both applications/feature modules being deployed on endpoints and also an application server hosting applications and providing server functionality in a cloud-based approach. *Id.* at ¶¶[0030], [0057], [0087], and [0092]. Malik discloses that the use of the feature modules on the server and

the endpoints operate in an unified manner in that the feature modules can allow an organization to "selectively add features and applications to endpoint devices without a complicated manual rollout," which may avoid problems that might be caused by changing components that previously worked properly, and which may be accomplished without "system down time, advance notice, restarting of devices on the network." Malik (Ex. 1005) at ¶¶[0030], [0043]-[0045] and [0084]. Also, Malik discloses that the use of the feature modules on the server and the endpoints operate in an unified manner in that the feature modules "allow reporting of data from endpoints to be used for various purposes on the server by applications, policies, or for visibility provided to a dashboard or for other viewing by an administrator using the MDM platform." Malik (Ex. 1005) at ¶¶[0057], [0087], and [0092]. Further, as a POSITA would understand, because the applications are distributed applications, the applications on the cloud server and the applications on the gateway device necessarily form unified capabilities, because that is the nature and purpose of distributed applications. Moreover, it is clear from the teachings of Malik that the distributed applications of Malik work together in a unified manner to perform the full functionality of the application, thus forming unified capabilities.

131. Regarding the element of "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware", Malik also discloses that developers of feature modules can take advantage of APIs as part of Malik's MDM platform, to allow reporting of information gathered by feature modules. Malik (Ex. 1005) at ¶¶ [0084], [0088], and Fig. 4. Malik further discloses use of a web based dashboard and also that developers can define and configure modules for deployment by use of a dashboard. Malik (Ex. 1005) at ¶¶[0094] and [0099].

132. I am informed that if a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation.

133. While Malik does not explicitly state that the applications are programmed by *upper layer* APIs *independent of network device hardware and cloud device hardware*, Malik does teach the use of APIs as discussed above. As would be understood and appreciated by a POSITA, it is the function and purpose of APIs to allow for programing in a manner that is

independent of device hardware. It is therefore an inherent characteristic of an API that it can be employed to program applications in a manner that is removed from or "independent of" (and in that sense, in an upper layer with respect to) "hardware."

134. Accordingly, Malik either explicitly or inherently discloses "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware."

f) '823 Patent at Independent Claim 19 and Dependent Claim 3: Malik Discloses "*an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications*" and "*an application management portal capable of managing life cycles of the distributed applications*"

135. Malik discloses *an application management portal* (e.g., module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint) *capable of managing life cycles* (e.g., installing, updating and removing modules in accordance with manifests) *of the plurality of network device applications* (e.g., feature modules installed on the endpoint device) *and plurality of cloud applications* (e.g., feature modules hosted by the server) as recited in claim 19, and discloses *an application management portal* (e.g., module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint) *capable of managing life cycles* (e.g., installing, updating and removing modules in accordance with manifests) *of the of the distributed applications* (e.g., feature applications) as recited in dependent claim 3.

136. Although the terminology of this limitation varies with respect to what the object of the lifecycle management is, the teachings of Malik are equally applicable to each limitation because the "distributed applications" as recited in claim 3 include the "plurality of network device applications and plurality of cloud applications" as recited in claim 19.

137. I note that the elements of being "capable of managing life cycles of the plurality of network device applications and plurality of cloud applications" and "capable of managing life cycles of the of the distributed applications" are each functional language, which may not be given patentable weight. However, to the extent that these limitation are given any patentable weight, they are disclosed either expressly or inherently in Malik. Additionally, to the extent that Malik does not explicitly disclose the management server managing life cycles of the plurality of

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
178

network device applications and plurality of cloud applications, it is clear based on the disclosure of Malik that that the management server of Malik is capable of such management.

138. Malik discloses that its system allows organizations to add, update, or remove features, Malik (Ex. 1005) at ¶¶[0030], [0078] and [0082], thus managing life cycles of the distributed applications. Malik discloses several components of its system that work together to accomplish this life cycle management. With reference to its Figures 2, and 3, Malik discloses a module manager and manifest server as components of the cloud server that keep track of all modules and their components as they are uploaded and maintained and manages pushing updates to endpoints by creating manifests that associate modules to be distributed to the endpoints. Malik (Ex. 1005) at¶¶[0060]-[0063]. Malik discloses a heartbeat server 116 as a component of the cloud server that provides an interface from the server components to the agents across the network, and allows the transfer of manifests and modules. Malik (Ex. 1005) at¶[0064]. Malik also discloses a module manager on the endpoint that receives, downloads and decrypts a waiting manifest, and in response to changes in the manifest also downloads or removes modules. Malik (Ex. 1005) at¶¶[0076], [0078], and [0082].

g) '823 Patent at Independent Claim 19 and Dependent Claim 3: Malik Discloses "*an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested distributed applications to the application management portal*"

139. Malik discloses *an infrastructure application marketplace* (e.g. web-based portal and file server) *in communication with the application management portal* (e.g., module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint)*, said infrastructure marketplace capable of providing tested (and certified) distributed applications to the application management portal* (e.g., modules can be tested before deploying new or updated feature modules) as recited in independent claim 19 and dependent claim 3.

140. I note that claim 3 recites providing "tested and certified" distributed applications, while claim 19 recites providing "tested" distributed applications. However, the difference does not provide any distinction, and the same disclosure in Malik is applicable to both terms.

141. I note that the element of being "capable of providing tested and certified distributed applications to the application management portal" is functional language that may not be given patentable weight. However, to the extent that this limitation is given any patentable weight, it is disclosed either expressly or inherently in Malik. Additionally, to the

extent that Malik does not disclose this element though explicit disclosure of providing such applications, based on the disclosure of Malik it is clear that the application store of Malik would be "capable of" providing such applications.

142. Malik discloses a web-based portal that allows for deployment of feature modules, discloses the selective purchasing of features, and also discloses a file server that acts as a repository to store the data/code components that comprise the feature modules. Malik (Ex. 1005) at¶¶ [0009], [0031], [0059] and [0067]. The file server allows modules to be elevated to the server infrastructure and later downloaded by agents on endpoints. Malik (Ex. 1005) at¶ [0059]. The file server is in communication with the components of the cloud server that make up the application management portal. Malik (Ex. 1005) at¶¶ [0060], [0067] and Fig 2,

143. Malik also discloses that the applications can be tested prior to being made available in the file server. Although Malik discloses that "less-tested" modules may be used and that robust testing may not be required, that is a disclosure that modules can be tested prior to being provided to the file server. Malik (Ex. 1005) at¶¶[0045] and [0054].

h) '823 Patent at Dependent Claim 2: Malik Discloses "*wherein the programmable network device and the programmable cloud device each include an access network interface unit configured to send and receive communications and a processor with a memory associated with the access network interface unit*"

144. Malik discloses that *the programmable network device* (e.g., endpoint) *and the programmable cloud device* (e.g., server) *each include an access network interface unit* (e.g., communication device or communication circuit) *configured to send and receive communications and a processor* (e.g., processor) *with a memory* (e.g., memory) *associated with the access network interface unit* as recited in dependent claim 2 of the '823 Patent.

145. Malik discloses that the endpoint devices include a communication circuit and memory for storing data and instructions. Malik (Ex. 1005) at¶[0035]. Malik also discloses that the components of the MDM system, whether on the server or the feature modules, can each include software operating on one or more processors, and that the computing devices described therein can each include one or more processors, memory, and communication devices. Malik (Ex. 1005) at¶[0039]. Because "associated with" is a broad term, and the entire MDM system of Malik relates to deploying feature modules from the server to an endpoint device and providing data from the feature modules to the server via a network, *see* Malik (Ex. 1005) at Fig. 1, the processor and memory of the server and of the endpoint device are associated with the

communication device/circuit that sends and receives the communications in each of the server and the endpoints.

i) '823 Patent at Claim 4: Malik Discloses "*wherein the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications*"

146. Malik discloses that *the application management portal* (e.g., module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint) *is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace* (e.g. web-based portal and file server); *testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications* as recited in claim 4 of the '823 Patent.

147. I am informed that claim language listing alternatives in the format set forth in claim 4 of the '823 patent is anticipated if at least one of the listed alternatives is disclosed in the prior art.

148. I note that the element of being "capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications" is functional language, which may not be given any patentable weight. However, to the extent that this limitation is given any patentable weight, it is disclosed either expressly or inherently in Malik.

149. Malik discloses that its system allows organizations to add, update, or remove features, Malik (Ex. 1005) at ¶¶[0030], [0078] and [0082], thus performing each of: receiving new applications from the infrastructure application marketplace; provisioning said distributed applications; and deprovisioning said distributed applications. With reference to its Figures 2, and 3, Malik discloses a module manager and manifest server as components of the cloud server that keep track of all modules and their components as they are uploaded and maintained in the file server, and that manage pushing updates to endpoints by creating manifests that associate modules to be distributed to the endpoints. Malik (Ex. 1005) at ¶¶[0060]-[0063]. Malik discloses a heartbeat server 116 as a component of the cloud server that provides an interface from the server components to the agents across the network, and allows the transfer of manifests and modules. Malik (Ex. 1005) at¶[0064]. Malik also discloses a module manager on the

endpoint that receives, downloads and decrypts a waiting manifest, and in response to changes in the manifest also downloads or removes modules. Malik (Ex. 1005) at ¶¶[0076], [0078], and [0082]. Thus, Malik discloses that the module manager, manifest server and heartbeat server of cloud server and module manager of agent on endpoint perform (are and capable of performing) at least *receiving new applications from the infrastructure application marketplace; provisioning said distributed applications; and deprovisioning said distributed applications.*

150. To the extent that Malik does not explicitly disclose the application management portal performing at least one of the recited actions, it is clear based on the disclosure in Malik that the management server of Malik is capable of at least receiving new applications from the infrastructure application marketplace; provisioning said distributed applications; and deprovisioning said distributed applications.

j) '823 Patent at Dependent Claim 15: Malik Discloses "*a distributed content service wherein contents generated by the at least one of the plurality of network device applications are shared with at least one of the plurality of cloud applications*"

151. Malik discloses *a distributed notification service* (e.g. reporting system) *wherein contents generated by the at least one of the plurality of network device applications* (e.g., feature modules installed on the endpoint device) *are shared with at least one of the plurality of cloud applications* (e.g., feature modules hosted by the server) as recited in claim 15 of the '823 Patent.

152. Malik discloses, with respect to its Figure 4, that the end point components of a reporting system that can be used as part of the module distribution and deployment system to allow deployed modules to report back status related to their installation as well as report back information to server components during normal execution. Malik (Ex. 1005) at¶¶[0057], [0087] and Fig. 4.

G. Ground 4: Malik Renders Obvious Claims 1, 2, 3, 4, 15 and 19 of the '823 Patent

153. Because Malik discloses each and every element of claims 1, 2, 3, 4, 15 and 19 of the '823 Patent for at least the reasons described above, and as set forth in Appendix B to the Request, Malik also renders obvious those claims.

154. However, with respect to claims 1 and 19, to the extent that Malik does not explicitly or inherently disclose "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud

applications independent of network device hardware and cloud device hardware," these elements are rendered obvious based on Malik, particularly in view of the knowledge of a POSITA as of June 13, 2023.

1. Malik in View of the Knowledge of a POSITA

155. As discussed above, a POSITA as of June 13, 2023 would have known about and understood the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware).

156. A POSITA as of June 13, 2023 would have understood that the APIs of Malik are "upper layer" APIs within the meaning of the '823 Patent, and that they would be enabled "to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" as recited in independent claims 1 and 19 of the '823 Patent.

157. Accordingly, to the extent that Malik does not explicitly or inherently disclose "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware," these elements are rendered obvious based on Malik in view of the knowledge of a POSITA as of June 13, 2023.

158. As discussed in the Request, the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware) was well within the knowledge of a POSITA prior to the June 13, 2023 priority date of the '823 Patent. Accordingly, a POSITA would have known about and understood the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware). Additionally, a POSITA would have understood that the APIs of Malik are "upper layer" APIs within the meaning of the '823 Patent, and that they would be enabled "to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" as recited in independent claims 1 and 19 of the '823 Patent

IV. Conclusion

159. I declare that all statements made herein of my knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Executed this 5th day of May, 2025.

By: James T. Geier

James T. Geier

Appendix A

CV of James T. Geier

James T. Geier

CONTACT INFORMATION

Name: James T. Geier
Title: Principal Consultant
Company: Wireless-Nets, Ltd.
Address: 2468 Locust Hill Blvd., Beavercreek, Ohio 45431 U.S.A.
Email: jimgeier@wireless-nets.com
Cell Phone: +1 937-829-0008
Website: www.wireless-nets.com

PROFILE

James Geier has 40 years of experience in the communications industry designing, analyzing, and implementing communications systems, wireless networks, and mobile wireless devices. He has authored over a dozen books on mobile and wireless topics, including Designing and Deploying 802.11 Wireless Networks (Cisco Press), Implementing 802.1X Security Solutions (Wiley), Wireless Networking Handbook (New Riders) and Network Re-engineering (McGraw-Hill). He had been an active participant within standards organizations, such as the IEEE 802.11 Working Group and the Wi-Fi Alliance. He has served as Chairman of the IEEE Computer Society, Dayton Section, and various conferences dealing with wireless networking.

PROFESSIONAL EXPERIENCE

Principal Consultant and Founder – *Wireless-Nets, Ltd. (Apr 2000 - present)*

- Designs mobile devices and implements corresponding software / firmware.
- Designs and integrates Wi-Fi, Bluetooth, cellular, Ethernet and USB transceivers into mobile devices, such as smart phones, hospital patient monitors, cable T.V. boxes, and bar code scanners.
- Designs large-scale wireless networks for municipalities, hospitals, airports, and manufacturing facilities for supporting voice, video and data applications.

President, CEO, and Cofounder – *Health Grade Networks LLC. (Aug 2014 – Dec 2015)*

- Provided strategic direction and management of the company for implementing wireless network solutions.

Product Engineer / Manager - *Monarch Marking Systems (Aug 1996 – Mar 2000)*

- Designed and implemented RF radios for Monarch's bar code scanners and printers.
- Integrated Ethernet and USB transceivers into tabletop printers.
- Designed and implemented wireless middleware software for improving performance between mobile wireless devices and application servers.
- Designed and implemented wireless network infrastructures for wireless bar code scanners and portable printers used in retail and manufacturing applications.

Senior Systems Engineer - *TASC, Inc. (Mar 1994 to Jul 1996)*

- Designed and implemented an enterprise-wide Ethernet / IEEE 802.3 and wireless network for Dayton Power and Light to support the migration from mainframe to client/server systems.
- Designed an information system architecture that supports internal and external communications for the U.S. Navy's NSSN attack submarine.
- Analyzed requirements for hardware, software, and support of the Joint Logistics Systems Center (JLSC) Materiel Management Standard System (MMSS) for the combined U.S. militaries.

Senior Systems Engineer - *Adroit Systems, Inc. (Aug 1992 to Feb 1994)*

- Researched and analyzed emerging wireless network technologies as part of the Department of Defense Airborne Reconnaissance Data Link Architecture (ARDA) study, supporting communications for airborne systems.
- Designed a software tool that aids network engineers in planning, upgrading and maintaining shipboard computer networks - based on a Small Business Innovative Research (SBIR) government grant obtained from the U.S. Navy.

Systems Design Engineer - *Information Systems Center, Captain U.S. Air Force (Sep 1990 – Jun 1992)*

- Evaluated the effectiveness of wireless LAN technology for use in mobile and portable military environments.
- Represented the Air Force as part of the IEEE 802.11 Wireless LAN standards development.
- Designed and implemented large-scale Ethernet / IEEE 802.3 LANs and WANs for various government organizations.

Communications Test Engineer - *AFCC Operational Test and Evaluation Center, Lieutenant U.S. Air Force (Sep 1986 - May 1989)*

- Performed analog, digital, and protocol tests on various government wireless computer networks and long-haul communications systems.
- Developed approaches and methods for testing computer networks.

Radar Technician - *75th TCF, U.S. Air Force (Dec 1977 - Jun 1983)*

- Performed acceptance testing of newly designed radar systems.
- Maintained automatic tracking radar systems in support of tactical Air Force operations worldwide.

EDUCATION

M.S., Electrical Engineering, *Air Force Institute of Technology (1990)* – thesis involved designing and implementing a wireless mesh network for the U.S. Department of Defense.

B.S., Electrical Engineering, *California State University (1985)*.

MILITARY EXPERIENCE

U.S. Air Force, Dec 1977 – Jun 1992 (Commissioned Officer Mar 1986 – Jun 1992).

BOOK PUBLICATIONS

- *Designing and Deploying 802.11 Wireless LANs (2nd Edition)*, Cisco Press, 2015.
- *Designing and Deploying 802.11n Wireless LANs*, Cisco Press, 2010.
- *Implementing 802.1x Security Solutions*, Wiley, 2008.
- *Deploying Voice over Wireless LANs*, Cisco Press, 2007.
- *Computer Security*, Wiley, 2007.
- *Computer Transfer and Backup*, Wiley, 2007.
- *CCIE Routing and Switching* – Official Exam Cortication Guide, 2nd Edition, Cisco Press, 2006.
- *Wireless Networks – 5-minute Fixes*, Wiley, 2006.
- *PCs – 5-minute Fixes*, Wiley, 2006.
- *Wireless Networks – First Step, Cisco Press*, 2005 (translated to Chinese, French, Hungarian, Italian, Korean, Polish, Portuguese, and Romanian).
- *Certified Wireless Analysis Professional - Official Study Guide*, McGraw-Hill, 2004.

- *Wireless LANs, 2nd Edition*, SAMs, 2001.
- *Wireless LANs*, Macmillan Technical Publishing, 1999.
- *Wireless Networking Handbook*, Macmillan (New Riders) Publishing, 1996.
- *Network Re-Engineering*, McGraw-Hill, 1996.

INDUSTRY AFFILIATIONS

Chairman, IEEE Computer Society - Dayton Section:
- Managed the 900-member organization and established a continuing education program.

Chairman, IEEE International Conference on Wireless LAN Implementation:
- Managed all aspects of the conference from 5/91 to 12/92.

Member, Wi-Fi Alliance
- Voting member of the Wireless ISP for Roaming (WISPr) committee.

IEEE 802.11 Wireless LAN Working Group
- Represented interests of the Department of Defense for dealing with applications and frequency allocations.

TEACHING EXPERIENCE

U.S. Naval Post Graduate School
- Developed and regularly instructed a course on wireless network design and security to students and faculty.

Wright State University
- Periodically instructed graduate courses on computer communications.

USAF Test and Evaluation School
- Developed and instructed a 240 hour training course on wireless system test and evaluation.

Conferences
- Regularly gives presentations at international conferences, including Supercomm (Asia), Scantech (Germany), and IBC (England).

Infocomm Solutions
- Developed and instructed workshops in India, Singapore and Malaysia on wireless network implementation.

Technology Training Corporation (TTC)
- Developed and instructed international training courses in Mexico and South America on wireless networking and network re-engineering.

Educational Services Institute
- Developed and instructed courses on software project management, software testing, system integration and network re-engineering as part of the Project Management Institute (PMI) Project Management Professional (PMP) certification program.

Onsite Training
- Regularly instructs workshops on wireless network design and deployment for product developers, system integrators, hospitals, and enterprises worldwide.

TESTIFYING EXPERIENCE

Cases where James Geier has testified at depositions and trials:

Ericsson v. Apple

Law firm: Fish & Richardson

Expert on behalf of the defendant (Apple) during 2022 regarding patent litigation within the International Trade Commission. Analyzed patents, products, prior art and other case documents associated with wireless roaming algorithms; analyzed product firmware; wrote expert reports regarding patent invalidity and non-infringement; testified at deposition; testified at the International Trade Commission in 2022 (Judge Elliot).

IOENGINE v. Ingenico, Inc.

Law firm: Sunstein Kann Murphy & Timbers

Expert on behalf of Ingenico during 2021-2022 regarding patent litigation within the U.S. District Court – Delaware. Analyzed patents, products, prior art and other case documents associated with wireless credit card processing systems; analyzed product firmware; wrote expert reports regarding patent non-infringement and invalidity; testified at deposition; testified at a jury trial within the District Court of Delaware in 2022 (Judge Bryson).

AIG Specialty Insurance Company v. Pegatron Corporation

Law Firm: Gordon Rees

Expert on behalf of the plaintiff (AIG Specialty Insurance Company) during 2020-2021 regarding Wi-Fi / cellular interference issues within the Northern District of Georgia. Analyzed test reports and case documents; wrote a declaration; testified at deposition.

NTT Corporation, et al. v. MediaTek, Acer, and Texas Instruments

Law Firm: Dickinson Wright

Expert on behalf of the plaintiff (NTT Corporation) during 2020-2021 regarding claim construction within district court of the Western District of Texas. Analyzed patents and case documents related to wireless communications; wrote a declaration regarding claim construction; testified at deposition.

Bertram Communications, LLC v. Simon Westlake, et al.

Law Firm: Kilpatrick Townsend

Expert on behalf of the plaintiff (Bertram Communications) during 2019-2020 regarding trade secret litigation within the State of Wisconsin, Circuit Court, Sheboygan County. Analyzed wireless designs and other case documents associated with wireless ISP systems in relation to trade secret analysis; wrote expert reports regarding associated analysis; testified at deposition.

Ingenico, Inc.

Law firm: Sunstein Kann Murphy & Timbers

Expert on behalf of Ingenico regarding an Inter Partes Review (IPR) with the USPTO Patent Trial and Appeal Board during 2018-2020. Analyzed patents, prior art and other case documents associated with wireless credit card processing systems; wrote declarations regarding patent invalidity; testified at depositions.

Next Caller, Inc.

Law firm: McDermott Will & Emery

Expert on behalf of Next Caller regarding an Inter Partes Review (IPR) with the USPTO Patent Trial and Appeal Board during 2018-2019. Analyzed patents, prior art and other case documents associated with caller profile systems; wrote a declaration regarding patent validity; testified at deposition.

Samsung, Inc.

Law firm: Kirkland & Ellis

Expert on behalf of Samsung regarding an Inter Partes Review (IPR) with the USPTO Patent Trial and Appeal Board during 2018-2019. Analyzed patents, prior art and other case documents associated with battery charging systems for mobile devices; wrote declarations regarding patent invalidity; testified at deposition.

Foxconn Technology Group

Law firm: McDermott Will & Emery

Expert on behalf of Foxconn regarding an Inter Partes Review (IPR) within the USPTO Patent Trial and Appeal Board during 2018. Analyzed patents, prior art and other case documents associated with short circuit protection for reversible connectors; wrote a declaration regarding patent invalidity; testified at deposition.

ZTE, Inc.

Law firm: McDermott Will & Emery

Expert on behalf of ZTE, Inc. regarding an Inter Partes Review (IPR) within the USPTO Patent Trial and Appeal Board during 2018. Analyzed patents, prior art and other case documents associated with supplying a power source to mobile devices over standardized ports; wrote a declaration regarding patent invalidity; testified at deposition.

Certain Wireless Headsets

Law firm: Haynes Boone

Expert on behalf of defendants (BlueAnt, Creative, GN Netcom, Jawbone and Sony) during 2015-2018 regarding patent litigation within the International Trade Commission (ITC). Analyzed patents and other case documents associated with wireless audio headsets; wrote expert reports regarding patent written description and inception date; testified at deposition.

Camsoft Data Systems, Inc. v. Southern Electronics Supply and Active Solutions, LLC

Law firm: Melancon & Rimes

Expert on behalf of the plaintiff (Camsoft Data Systems) during 2014-2018 regarding trade secret litigation within the 19th Judicial District Court – State of Louisiana. Analyzed wireless designs and other case documents associated with wireless video surveillance systems in relation to trade secret analysis; wrote expert reports regarding associated analysis; testified at deposition.

United States of America v. IBM Corporation

Law firm: Jones Day

Expert on behalf of the defendant (IBM Corporation) during 2017 regarding litigation within the U.S. District Court, Northern District of Illinois - Eastern Division. Analyzed designs, contracts, and other case documents associated with wireless networks and integration methods; wrote an expert report regarding associated analysis; testified at deposition.

Emerson Electric Co.

Law firm: Ropes & Gray

Expert on behalf of Emerson Electric Co. regarding an Inter Partes Reviews (IPRs) within the USPTO Patent Trial and Appeal Board during 2016-2017. Analyzed patents, prior art and other case documents associated with optimum route determination over computer networks; wrote declarations regarding patent invalidity; testified at multiple depositions.

Emerson Electric Co.

Law firm: Ropes & Gray

Expert on behalf of Emerson Electric Co. regarding a Covered Business Method Patent Review (CBMPR) within the USPTO Patent Trial and Appeal Board during 2016-2017. Analyzed patents, prior art and other case documents associated with systems for communicating information; wrote declarations regarding patent invalidity; testified at multiple depositions.

Chrimar v. Aerohive

Law firm: Wilson, Sonsini, Goodrich & Rosati

Expert on behalf of the defendant (Aerohive) during 2016-2017 regarding patent litigation within the U.S. District Court – Eastern Division of Texas (Judge Gilstrap). Analyzed patents, products, and other case documents associated with power-over-Ethernet technologies; performed testing of power-over-Ethernet products; wrote an expert report regarding patent non-infringement; testified at deposition; testified at a jury trial within the District Court of Eastern Texas in 2017 (Judge Gilstrap).

Intellectual Ventures v. AT&T

Law firm: Dechert

Expert on behalf of the plaintiff (Intellectual Ventures) during 2014-2017 regarding patent litigation within the U.S. District Court - Delaware. Analyzed patents, prior art and other case documents associated with IEEE 802.11 encryption and quality of service (QoS) technologies; performed testing and protocol analysis of Wi-Fi products; wrote expert reports regarding patent infringement and validity; testified at deposition.

Chrimar v. AMX

Law firm: McDermott, Will & Emery

Expert on behalf of the defendant (AMX) during 2016 regarding patent litigation within the U.S. District Court – Eastern Division of Texas. Analyzed patents, products, prior art and other case documents associated with power-over-Ethernet technologies; performed testing of power-over-Ethernet products; wrote expert reports regarding patent non-infringement and invalidity; testified at deposition.

Universal Remote

Law firm: Ostrolenk Faber

Expert on behalf of Universal Remote regarding an Inter Partes Review (IPR) within the USPTO Patent Trial and Appeal Board filed during 2015. Analyzed patents, prior art and other case documents associated with TV remote control technologies; wrote a declaration regarding patent invalidity; testified at deposition.

Skyhook v. Google

Law firm: Tensegrity

Expert on behalf of the plaintiff (Skyhook) during 2012-2015 regarding patent litigation within the U.S. District Court - Boston. Analyzed patents, software, and other case documents associated with Wi-Fi-based location systems; performed in-depth review and analysis of associated software; wrote an expert report regarding patent infringement; testified at deposition.

MTEL v. UPS

Law firm: Reed & Scardino

Expert on behalf of the plaintiff (MTEL) during 2014 regarding patent litigation within the U.S. District Court – Atlanta. Analyzed patents, prior art and other case documents associated with mail delivery notifications; wrote an expert report regarding patent validity; testified at deposition.

Motorola v. Microsoft

Law firm: Sidley Austin

Expert on behalf of the defendant (Microsoft) during 2012 regarding patent litigation within the International Trade Commission. Analyzed patents, products, prior art and other case documents associated with data addressing and data update mechanisms; wrote expert reports on patent invalidity and indirect non-infringement; testified at deposition; testified at the International Trade Commission in 2012 (Judge Shaw).

Motorola v. Microsoft

Law firm: Sidley Austin

Expert on behalf of the defendant (Microsoft) during 2011 regarding patent litigation within the International Trade Commission. Analyzed patents, products, prior art and other case documents associated with security and encryption protocols; analyzed product firmware; wrote expert reports regarding patent invalidity and non-infringement; testified at deposition; testified at the International Trade Commission in 2011 (Judge Shaw).

Motorola v. Microsoft

Law firm: Sidley Austin

Expert on behalf of the defendant (Microsoft) during 2011 regarding patent litigation within the U.S. District Court – Southern Florida. Analyzed patents, products, prior art and other case documents associated with application registry and data update technologies; wrote expert reports regarding patent invalidity and non-infringement; testified at deposition.

Autocell v. Cisco Systems

Law Firm: Hanify & King

Expert on behalf of the plaintiff (Autocell) during 2009-2011 regarding patent litigation within the U.S. District Court - Delaware. Analyzed patents, products, prior art and other case documents associated with transmit power control of 802.11 radios and access points; performed laboratory and field testing of associated products; analyzed product firmware; wrote expert reports regarding patent infringement and validity; testified at deposition.

Truckstop.Net v. Sprint Communications

Law Firm: Holland & Hart

Expert on behalf of the plaintiff (Truckstop.Net) during 2004-2010 regarding contract litigation within the U.S. District Court - Idaho. Analyzed case documents including wireless network designs and contracts associated with wireless networks installed at truck stops; performed in-depth field testing and analysis of wireless signal coverage and performance at truck stops located throughout the U.S.; wrote expert reports regarding test results and review of case documents; testified at deposition.

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC PAYMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
90/019,948	**05/06/2025 02:50:18 PM Z ET**	**UNP-16002**

Title of Invention

Distributed Software Defined Networking

Application Information

APPLICATION TYPE	Utility / ex parte reexam	PATENT #	11695823
CONFIRMATION #	2943	FILED BY	Jennifer Lacroix
PATENT CENTER #	70310235	AUTHORIZED BY	-
CUSTOMER #	112449	FILING DATE	-
CORRESPONDENCE ADDRESS	-	FIRST NAMED INVENTOR	Pouya Taaghol

Payment Information

PAYMENT METHOD	PAYMENT TRANSACTION ID	PAYMENT AUTHORIZED BY
DA / 506990	**E202556E51146848**	**Jennifer Lacroix**

FEE CODE	DESCRIPTION	ITEM PRICE($)	QUANTITY	ITEM TOTAL($)
1812	REQUEST FOR EX PARTE REEXAM- NON-STREAMLINED	13545.00	1	13545.00
			TOTAL AMOUNT:	**$13,545.00**

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Ex. 1001

U.S. Patent No. 11,695,823

US011695823B1

(12) **United States Patent**
Taaghol et al.

(10) **Patent No.: US 11,695,823 B1**
(45) **Date of Patent: *Jul. 4, 2023**

(54) **DISTRIBUTED SOFTWARE DEFINED NETWORKING**

(71) Applicant: **Big Data Federation, Inc.**, Santa Clara, CA (US)

(72) Inventors: **Pouya Taaghol**, San Jose, CA (US); **Vivek Ramanna**, San Jose, CA (US)

(73) Assignee: **Edge Networking Systems, LLC**, Austin, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **17/142,983**

(22) Filed: **Jan. 6, 2021**

Related U.S. Application Data

(63) Continuation of application No. 16/900,963, filed on Jun. 14, 2020, now Pat. No. 10,893,095, which is a (Continued)

(51) **Int. Cl.**
H04L 67/10 (2022.01)
H04L 67/00 (2022.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04L 67/10*** (2013.01); ***G06F 21/44*** (2013.01); ***G06F 21/53*** (2013.01); ***G06F 21/57*** (2013.01); ***H04L 9/0822*** (2013.01); ***H04L 41/0803*** (2013.01); ***H04L 41/0893*** (2013.01); ***H04L 41/0895*** (2022.05); ***H04L 45/563*** (2013.01); ***H04L 45/64*** (2013.01); ***H04L 63/0227*** (2013.01); ***H04L 67/1076*** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC H04L 9/08–0838; H04L 29/08135; H04L 29/08981; H04L 41/08–082; H04L 41/0866–0876; H04L 63/0218; H04L 67/10; H04L 67/34; H04L 41/0895; H04L 41/22; H04L 45/00; H04L 45/56–563; H04L 45/586; H04L 45/64; H04L 45/645; H04L 45/655; H04L 45/76; H04L 41/12–122; H04L 45/036
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,114,021 B2 9/2006 Seshadri
7,630,706 B2 12/2009 Weigand
(Continued)

OTHER PUBLICATIONS

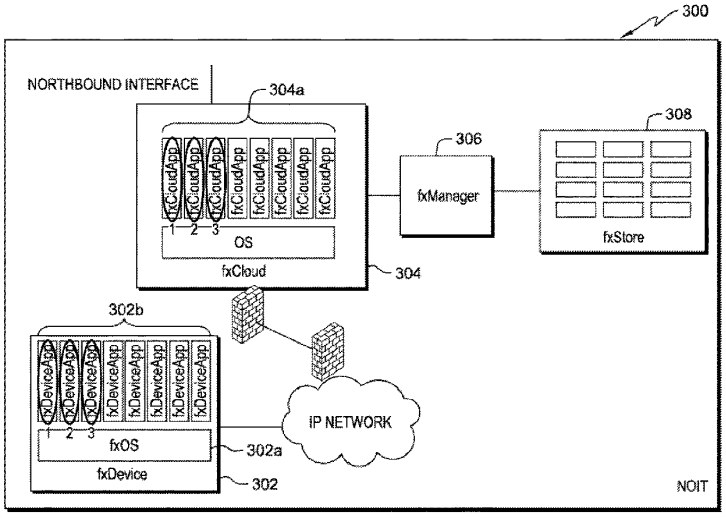
McKeown, et al. "OpenFlow: Enabling Innovation in Campus Networks", Mar. 14, 2008.
(Continued)

Primary Examiner — Brendan Y Higa
(74) *Attorney, Agent, or Firm* — Roark IP

(57) **ABSTRACT**

The Distributed Software Defined Network (dSDN) disclosed herein is an end-to-end architecture that enables secure and flexible programmability across a network with full lifecycle management of services and infrastructure applications (fxDeviceApp). The dSDN also harmonizes application deployment across the network independent of the hardware vendor. As a result, the dSDN simplifies the network deployment lifecycle from concept to design to implementation to decommissioning.

19 Claims, 18 Drawing Sheets



Related U.S. Application Data

continuation of application No. 15/836,824, filed on Dec. 9, 2017, now Pat. No. 10,686,871, which is a continuation of application No. 14/295,331, filed on Jun. 4, 2014, now Pat. No. 9,843,624.

(60) Provisional application No. 61/834,807, filed on Jun. 13, 2013.

(51) **Int. Cl.**

H04L 41/0803	(2022.01)
H04L 9/40	(2022.01)
H04L 9/08	(2006.01)
H04L 41/0893	(2022.01)
H04L 45/00	(2022.01)
H04L 45/64	(2022.01)
H04L 67/1074	(2022.01)
H04L 41/0895	(2022.01)
H04W 4/00	(2018.01)
H04L 41/082	(2022.01)
G06F 21/57	(2013.01)
G06F 21/44	(2013.01)
G06F 21/53	(2013.01)
G06F 8/60	(2018.01)
G06F 8/70	(2018.01)
H04L 41/5019	(2022.01)
G06F 9/455	(2018.01)

(52) **U.S. Cl.**

CPC ***H04L 67/34*** (2013.01); *G06F 8/60* (2013.01); *G06F 8/70* (2013.01); *G06F 2009/45587* (2013.01); *H04L 41/082* (2013.01); *H04L 41/5019* (2013.01); *H04L 63/0218* (2013.01); *H04W 4/00* (2013.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,132,166	B2	3/2012	DeHaan	
8,495,611	B2	7/2013	McCarthy	
8,528,108	B2	9/2013	Ammer	
8,650,299	B1	2/2014	Huang	
8,756,595	B2	6/2014	Arcilla	
8,762,501	B2 *	6/2014	Kempf	H04L 47/825 709/228
8,843,933	B1	9/2014	Holler	
9,226,333	B2	12/2015	Steiner	
2002/0016954	A1	2/2002	Charisius	
2004/0107417	A1	6/2004	Chia	
2004/0237081	A1	11/2004	Homiller	
2005/0246703	A1	11/2005	Ahonen	
2006/0271341	A1	11/2006	Brown	
2007/0078988	A1	4/2007	Miloushev	
2008/0209491	A1	8/2008	Hasek	
2009/0007091	A1	1/2009	Appiah	
2009/0113327	A1	4/2009	Pogrebinsky	
2009/0276771	A1	11/2009	Nickolov	
2009/0300057	A1	12/2009	Friedman	
2011/0075611	A1	3/2011	Choi	
2011/0265077	A1	10/2011	Collison	
2012/0066665	A1	3/2012	Kunze	
2012/0096071	A1	4/2012	Murphey	
2012/0266156	A1	10/2012	Spivak	
2012/0300615	A1	11/2012	Kempf	
2013/0007254	A1	1/2013	Fries	
2013/0054763	A1 *	2/2013	Van der Merwe	H04W 8/12 709/220
2013/0160147	A1	6/2013	Draluk	
2013/0223442	A1	8/2013	Narayanan	
2013/0263104	A1	10/2013	Basel	
2014/0052976	A1	2/2014	Marino	
2014/0082156	A1 *	3/2014	Jagtap	G06F 9/5027 709/220
2014/0241158	A1	8/2014	Anthony, Jr.	
2014/0254373	A1	9/2014	Varma	
2014/0259012	A1	9/2014	Nandlall	
2014/0317293	A1	10/2014	Shatzkamer	
2014/0351428	A1 *	11/2014	Madani	H04L 47/76 709/224
2015/0071053	A1 *	3/2015	Kempf	H04W 24/02 370/216
2015/0160877	A1	6/2015	Larkin	
2015/0173121	A1	6/2015	Mklos	
2015/0242237	A1	8/2015	Spivak	
2016/0224460	A1	8/2016	Bryant	
2017/0257432	A1 *	9/2017	Fu	H04L 67/1014
2018/0309747	A1 *	10/2018	Sweet	H04L 63/20
2019/0165997	A1	5/2019	Shaikh	

OTHER PUBLICATIONS

Li et al. "Toward Software-Defined Cellular Networks", "2012 European Workshop on Software Defined Networking".

"Network Functions Virtualisation", "SDN and OpenFlow World Congress", Darmstadt-Germany, Oct. 22-24, 2012.

Yap et al., "OpenRoads: Empowering Research in Mobile Networks", ACM SIGCOMM Computer Communication Review, 2010.

Bansal et al., "OpenRadio: A Programmable WirelessDataplane", HotSDN'12, Aug. 13, 2012, Helsinki, Finland.

Stallings et al., "Software-Defined Networks and OpenFlow", The Internet Protocol Journal, vol. 16, No. 1, Mar. 2013.

Luo et al., "Software Defined Networking across Distributed Datacenters over Cloud", IEEE Int. Conference on Cloud Computing Tech, 2013.

Chappell, David, "A Short Introduction to Cloud Platforms", Aug. 2008.

* cited by examiner

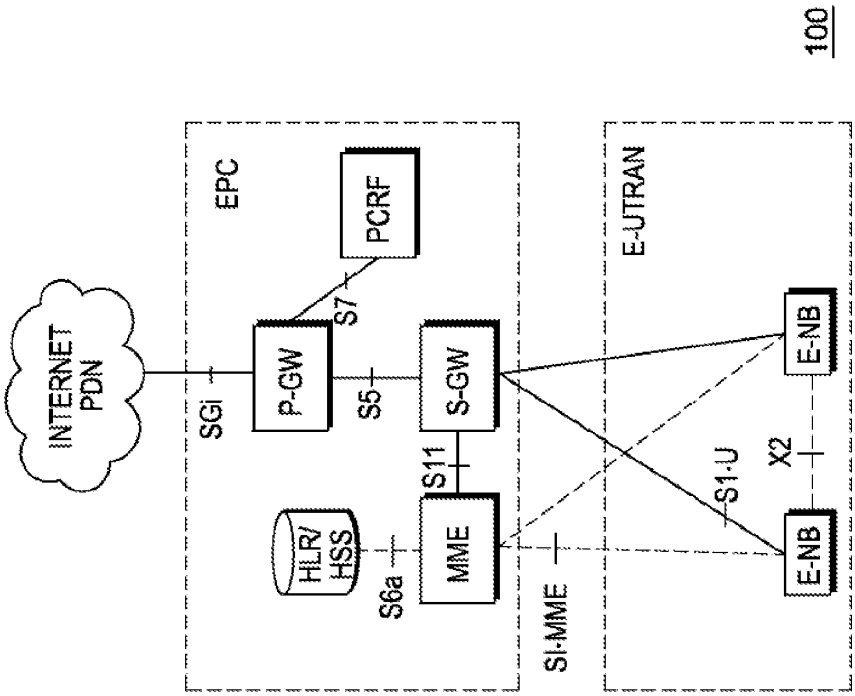

100
EPC
E-UTRAN
INTERNET PDN
SGi
P-GW
S7
PCRF
S5
S-GW
S11
MME
HLR/ HSS
S6a
S1-MME
S1-U
X2
E-NB
E-NB
LTE-Uu

FIG. 1

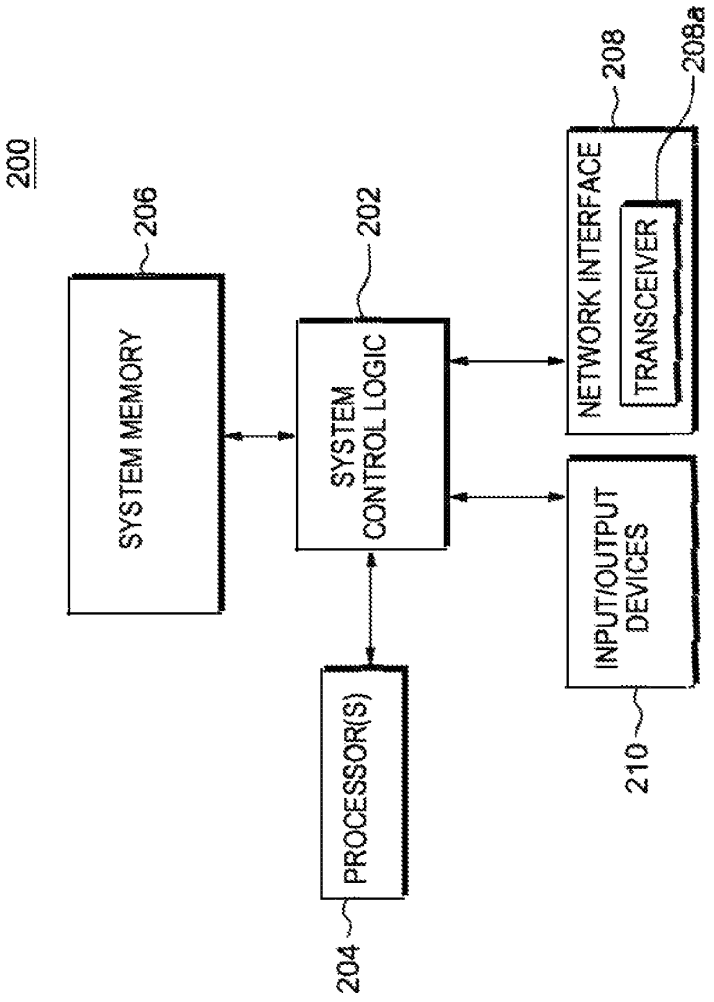

200
206
SYSTEM MEMORY
202
SYSTEM CONTROL LOGIC
208
208a
NETWORK INTERFACE
TRANSCEIVER
INPUT/OUTPUT DEVICES
210
PROCESSOR(S)
204

FIG. 2

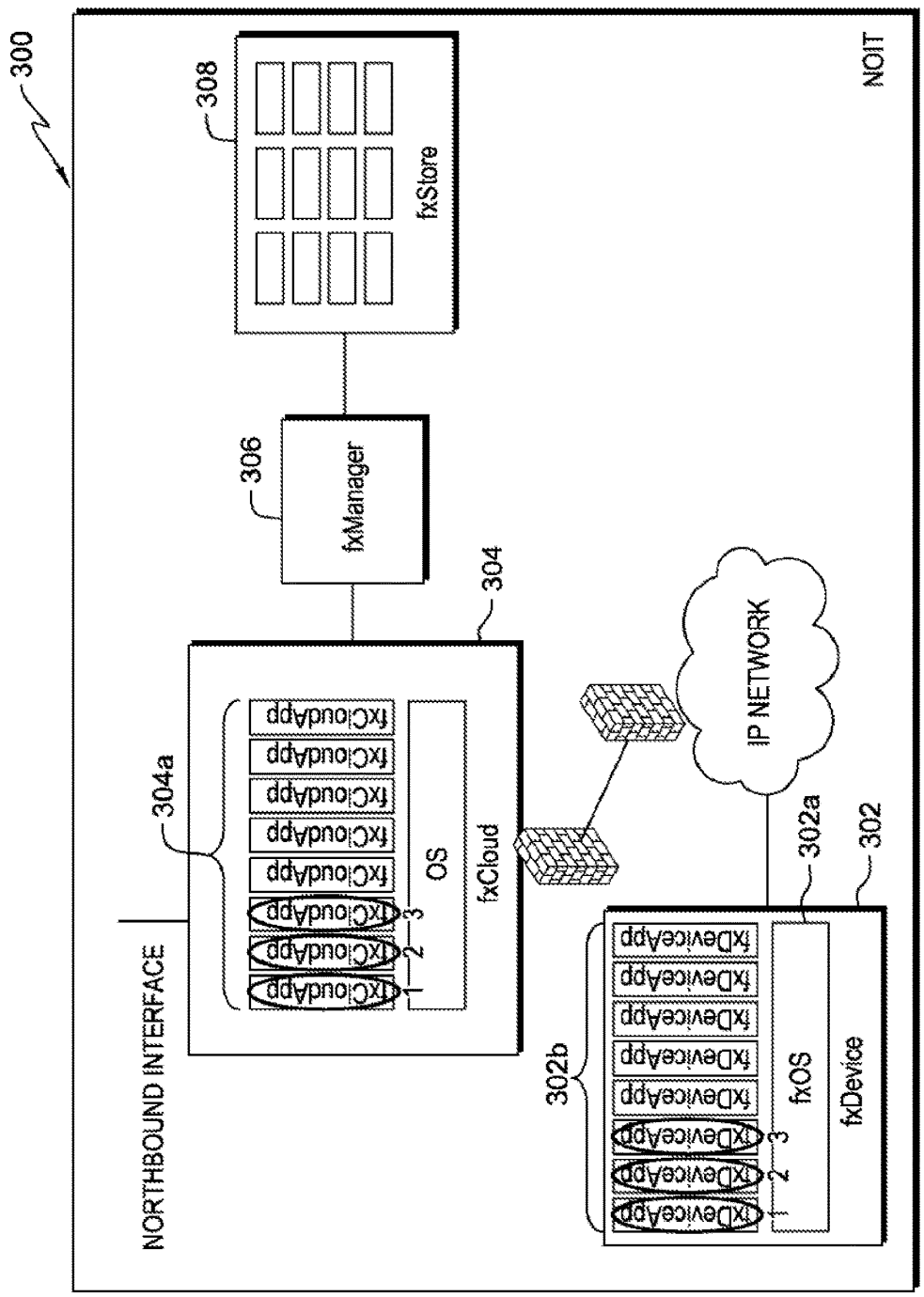

300
NOIT
308
fxStore
306
fxManager
304
304a
fxCloudApp
OS
fxCloud
IP NETWORK
NORTHBOUND INTERFACE
302a
302
302b
fxDeviceApp
fxOS
fxDevice
1
2
3

FIG. 3

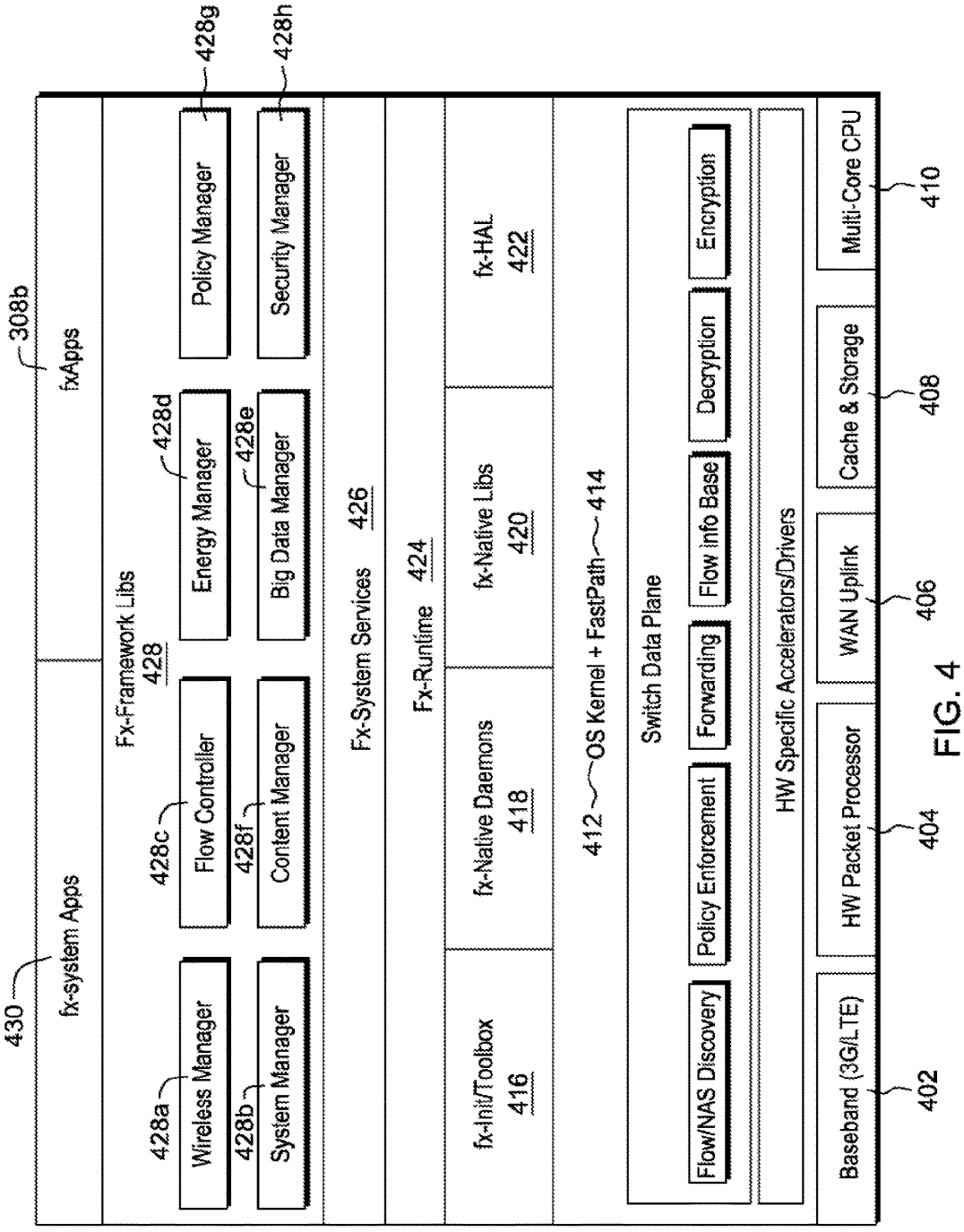

430
fx-system Apps
308b
fxApps
Fx-Framework Libs
428
428a
Wireless Manager
428b
System Manager
428c
Flow Controller
428f
Content Manager
428d
Energy Manager
428e
Big Data Manager
428g
Policy Manager
428h
Security Manager
Fx-System Services 426
Fx-Runtime 424
fx-Init/Toolbox
416
fx-Native Daemons
418
fx-Native Libs
420
fx-HAL
422
412
OS Kernel + FastPath
414
Switch Data Plane
Flow/NAS Discovery
Policy Enforcement
Forwarding
Flow info Base
Decryption
Encryption
HW Specific Accelerators/Drivers
Baseband (3G/LTE)
402
HW Packet Processor
404
WAN Uplink
406
Cache & Storage
408
Multi-Core CPU
410

FIG. 4

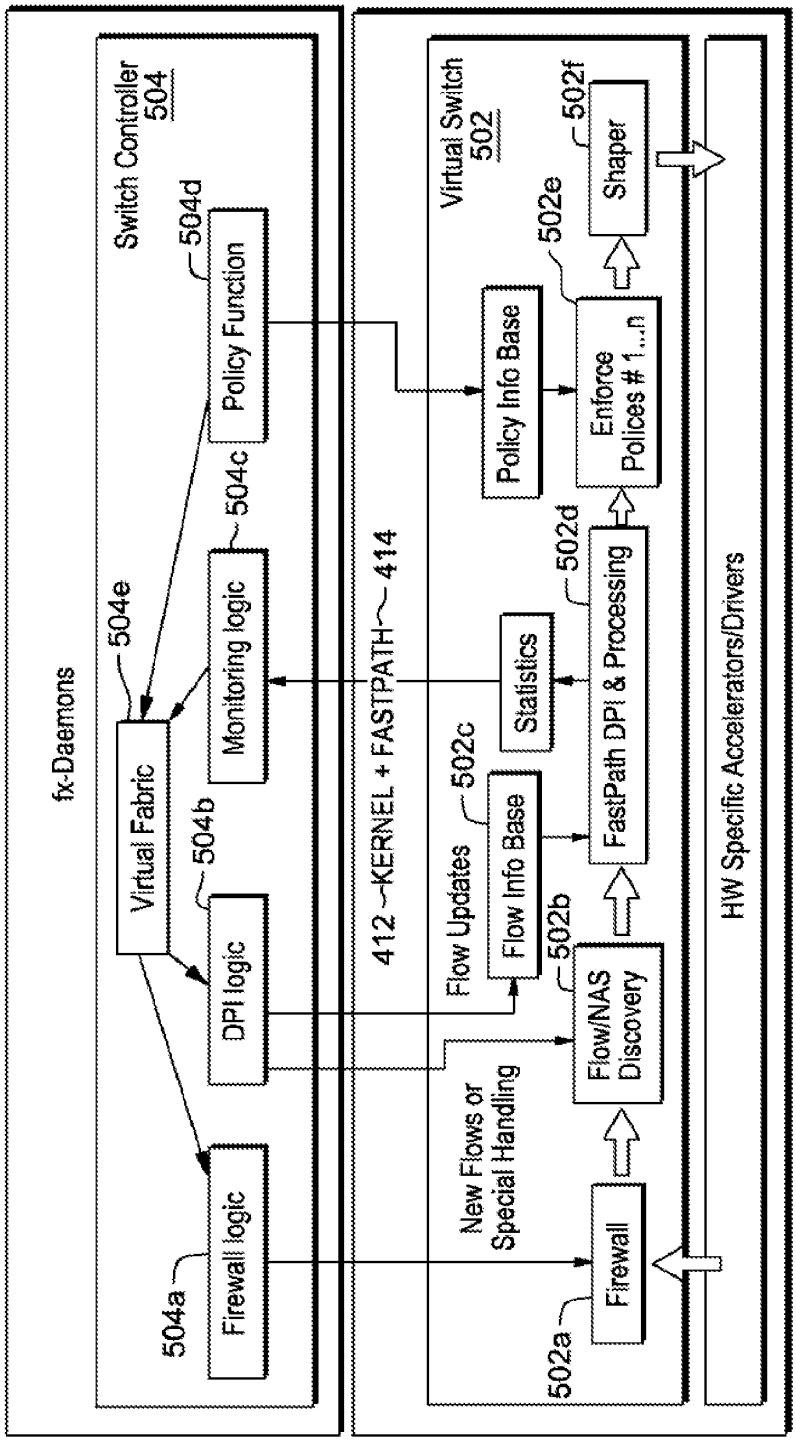

fx-Daemons
Switch Controller 504
504a
Firewall logic
504b
DPI logic
504c
Monitoring logic
504d
Policy Function
504e
Virtual Fabric
412 ~KERNEL + FASTPATH~ 414
Virtual Switch 502
502a
Firewall
502b
Flow/NAS Discovery
502c
Flow Info Base
Flow Updates
Statistics
502d
FastPath DPI & Processing
502e
Enforce Polices # 1...n
Policy Info Base
502f
Shaper
HW Specific Accelerators/Drivers
New Flows or Special Handling
Control
Data

FIG. 5

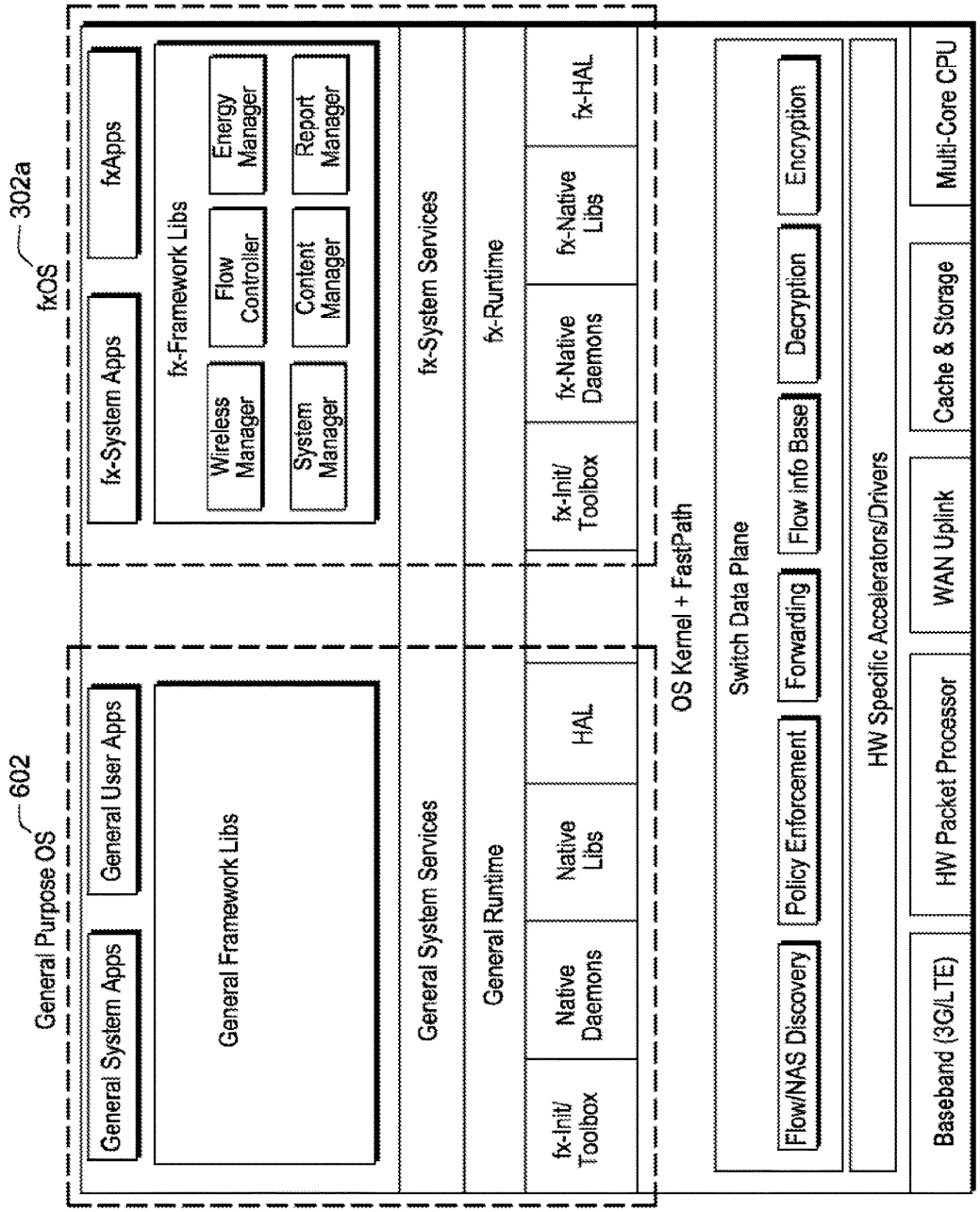

602
General Purpose OS
General System Apps
General User Apps
General Framework Libs
General System Services
fx-Init/ Toolbox
Native Daemons
General Runtime
Native Libs
HAL
302a
fxOS
fx-System Apps
fxApps
fx-Framework Libs
Wireless Manager
Flow Controller
Energy Manager
System Manager
Content Manager
Report Manager
fx-System Services
fx-Runtime
fx-Init/ Toolbox
fx-Native Daemons
fx-Native Libs
fx-HAL
OS Kernel + FastPath
Switch Data Plane
Flow/NAS Discovery
Policy Enforcement
Forwarding
Flow info Base
Decryption
Encryption
HW Specific Accelerators/Drivers
Baseband (3G/LTE)
HW Packet Processor
WAN Uplink
Cache & Storage
Multi-Core CPU

FIG. 6

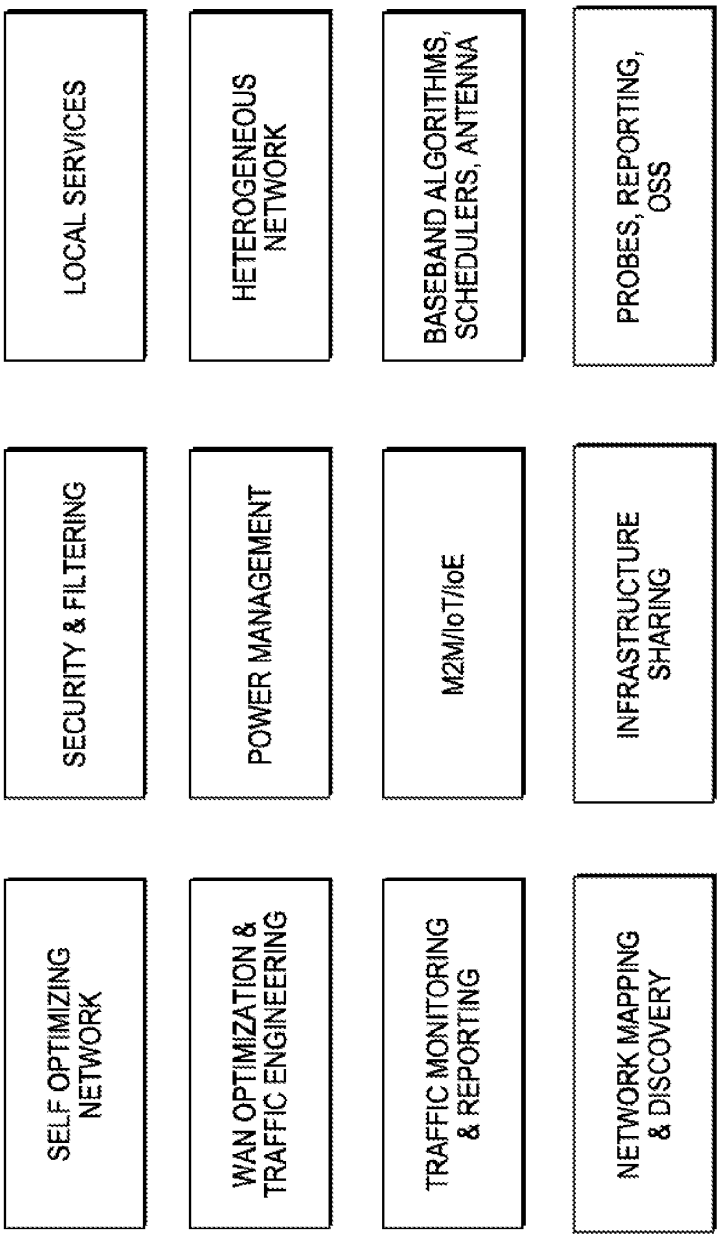

LOCAL SERVICES
HETEROGENEOUS NETWORK
BASEBAND ALGORITHMS, SCHEDULERS, ANTENNA
PROBES, REPORTING, OSS
SECURITY & FILTERING
POWER MANAGEMENT
M2M/IoT/IoE
INFRASTRUCTURE SHARING
SELF OPTIMIZING NETWORK
WAN OPTIMIZATION & TRAFFIC ENGINEERING
TRAFFIC MONITORING & REPORTING
NETWORK MAPPING & DISCOVERY

FIG. 7

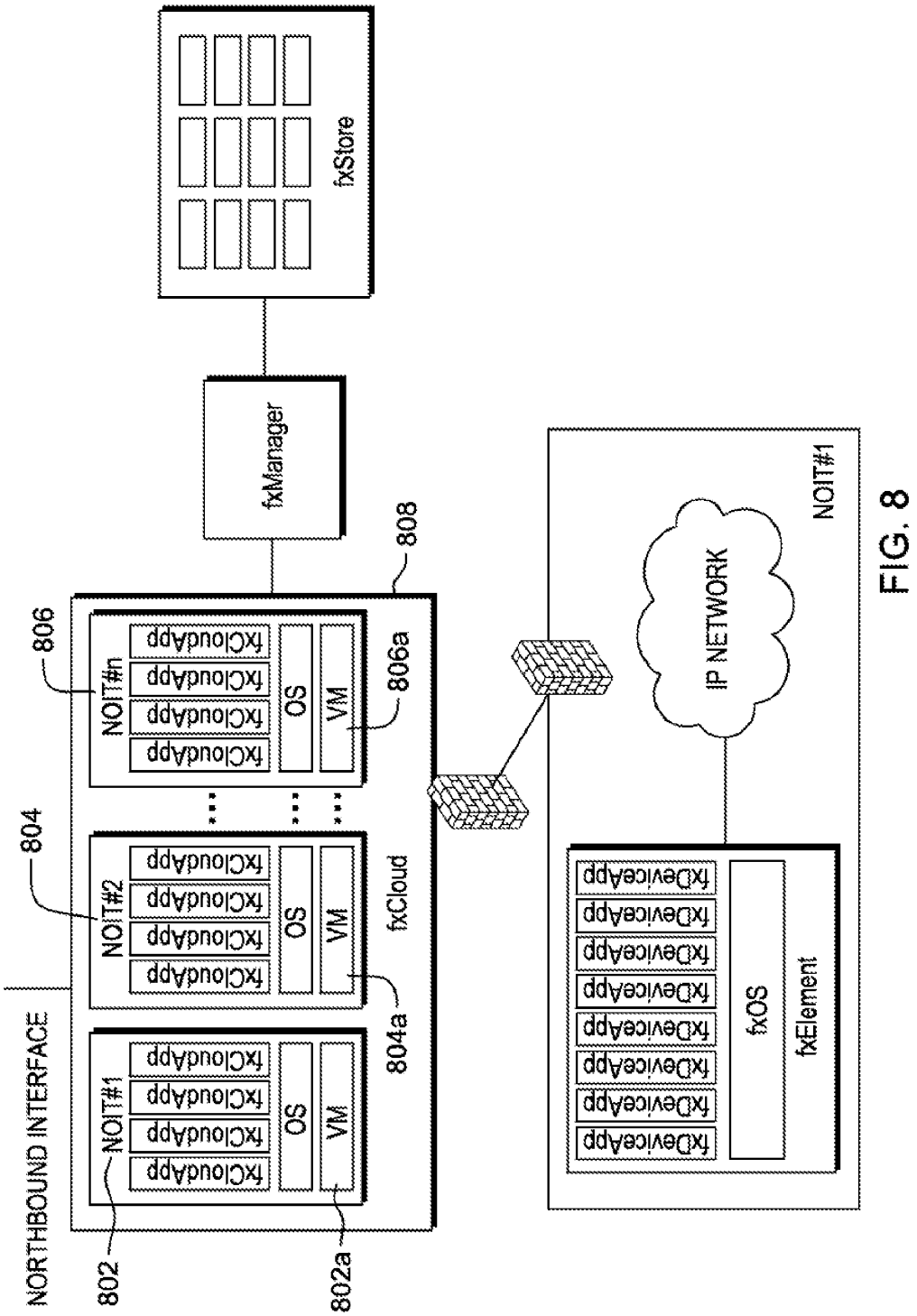

fxStore
fxManager
808
806
NOIT#n
fxCloudApp
OS
VM
806a
804
NOIT#2
fxCloud
804a
NOIT#1
NORTHBOUND INTERFACE
802
802a
IP NETWORK
NOIT#1
fxDeviceApp
fxOS
fxElement

FIG. 8

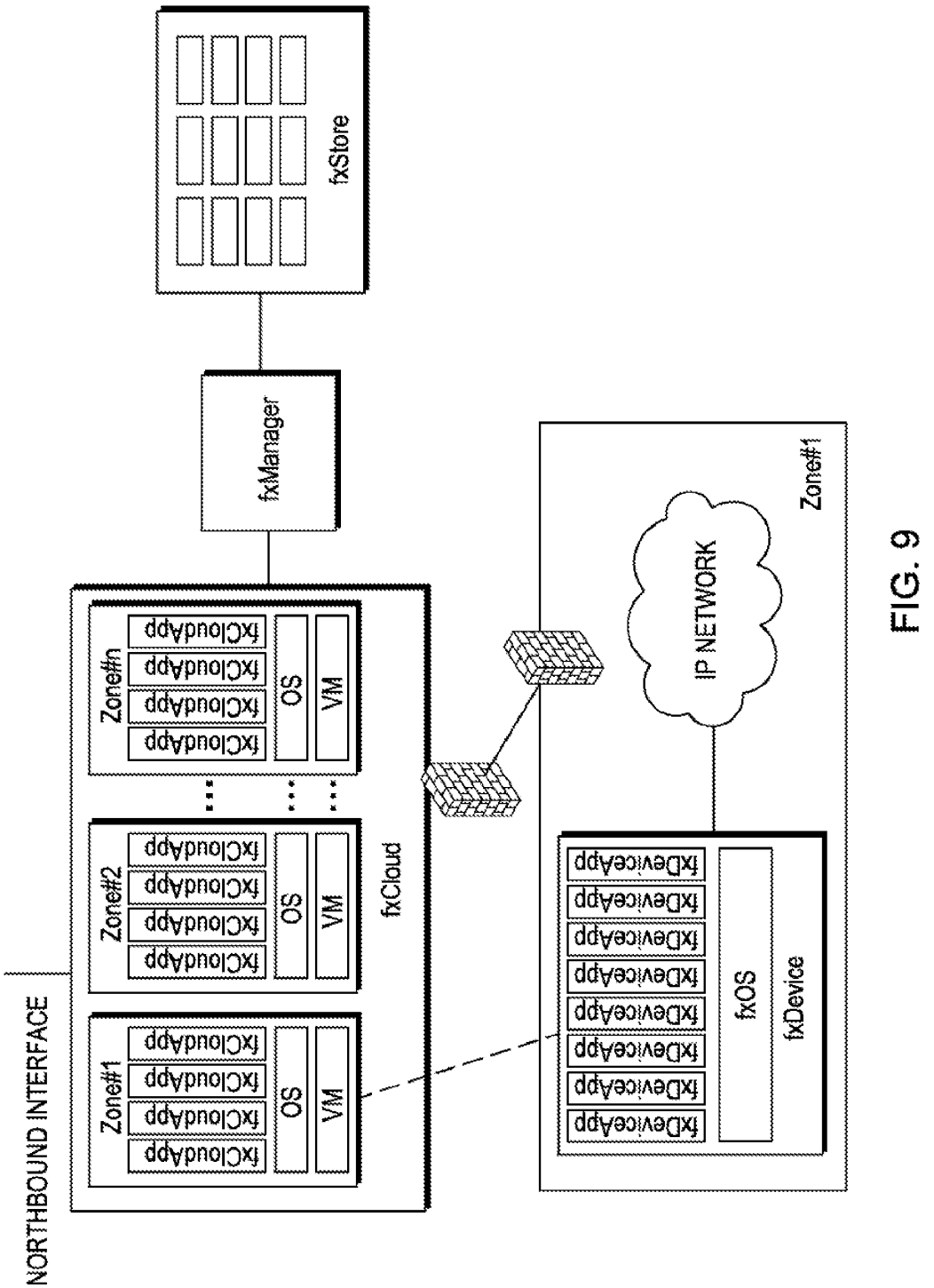

fxStore
fxManager
fxCloud
Zone#n
Zone#2
Zone#1
fxCloudApp
OS
VM
NORTHBOUND INTERFACE
IP NETWORK
Zone#1
fxDeviceApp
fxOS
fxDevice

FIG. 9

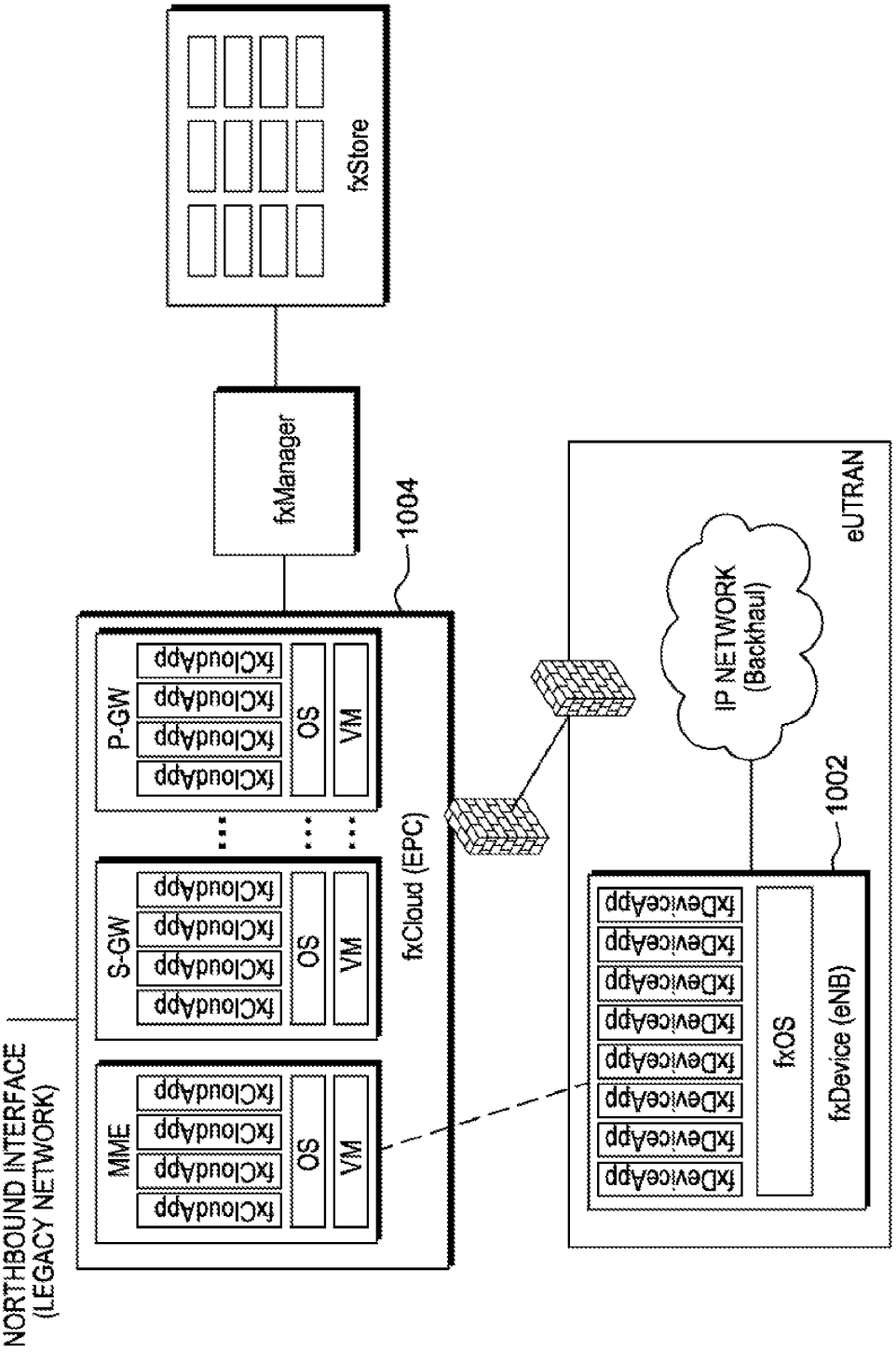

fxStore
fxManager
1004
fxCloud (EPC)
P-GW
S-GW
MME
fxCloudApp
OS
VM
NORTHBOUND INTERFACE
(LEGACY NETWORK)
IP NETWORK
(Backhaul)
eUTRAN
1002
fxDeviceApp
fxOS
fxDevice (eNB)

FIG. 10

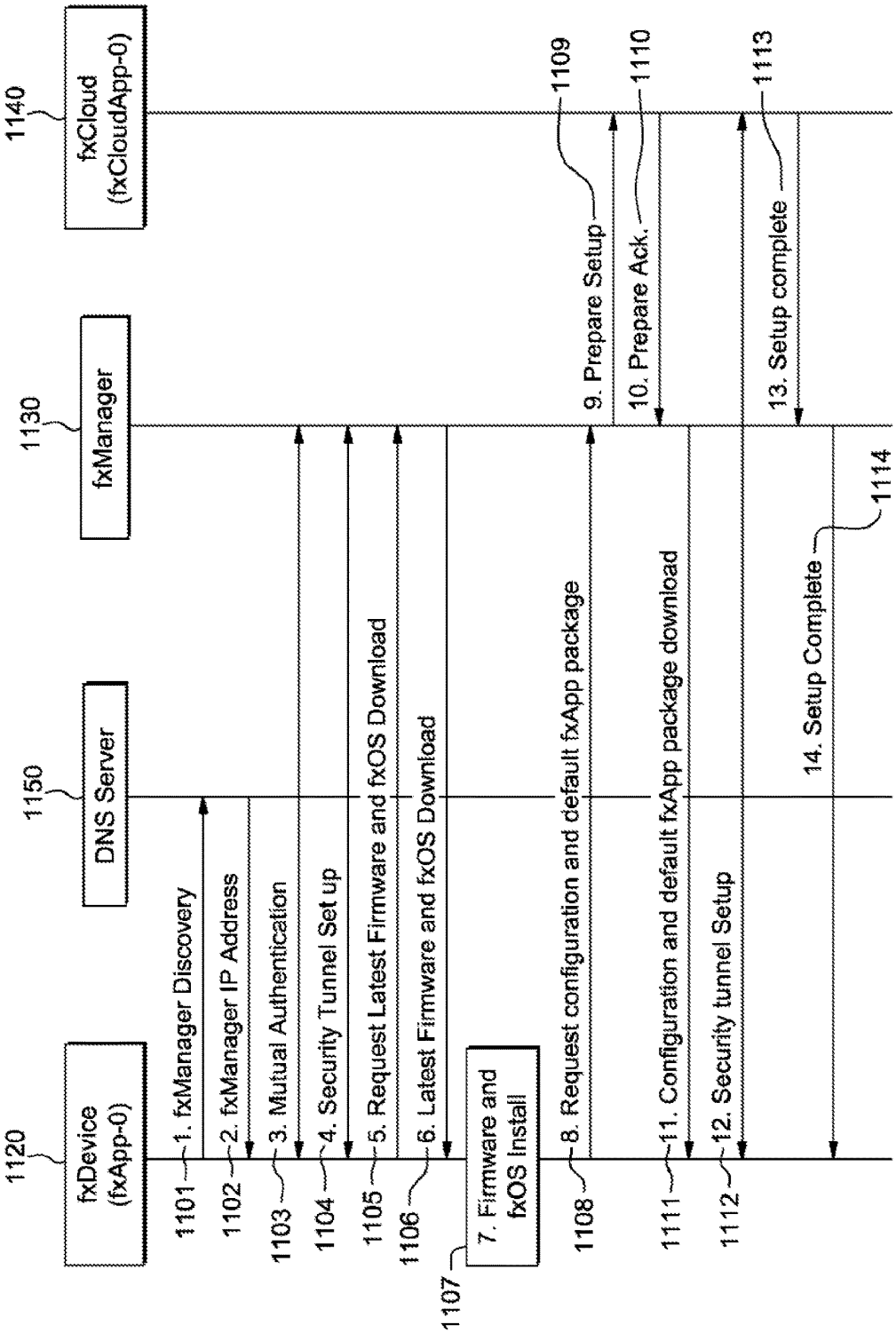

1120
fxDevice (fxApp-0)
1150
DNS Server
1130
fxManager
1140
fxCloud (fxCloudApp-0)
1. fxManager Discovery
2. fxManager IP Address
3. Mutual Authentication
4. Security Tunnel Set up
5. Request Latest Firmware and fxOS Download
6. Latest Firmware and fxOS Download
7. Firmware and fxOS Install
8. Request configuration and default fxApp package
9. Prepare Setup
10. Prepare Ack.
11. Configuration and default fxApp package download
12. Security tunnel Setup
13. Setup complete
14. Setup Complete
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114

FIG. 11

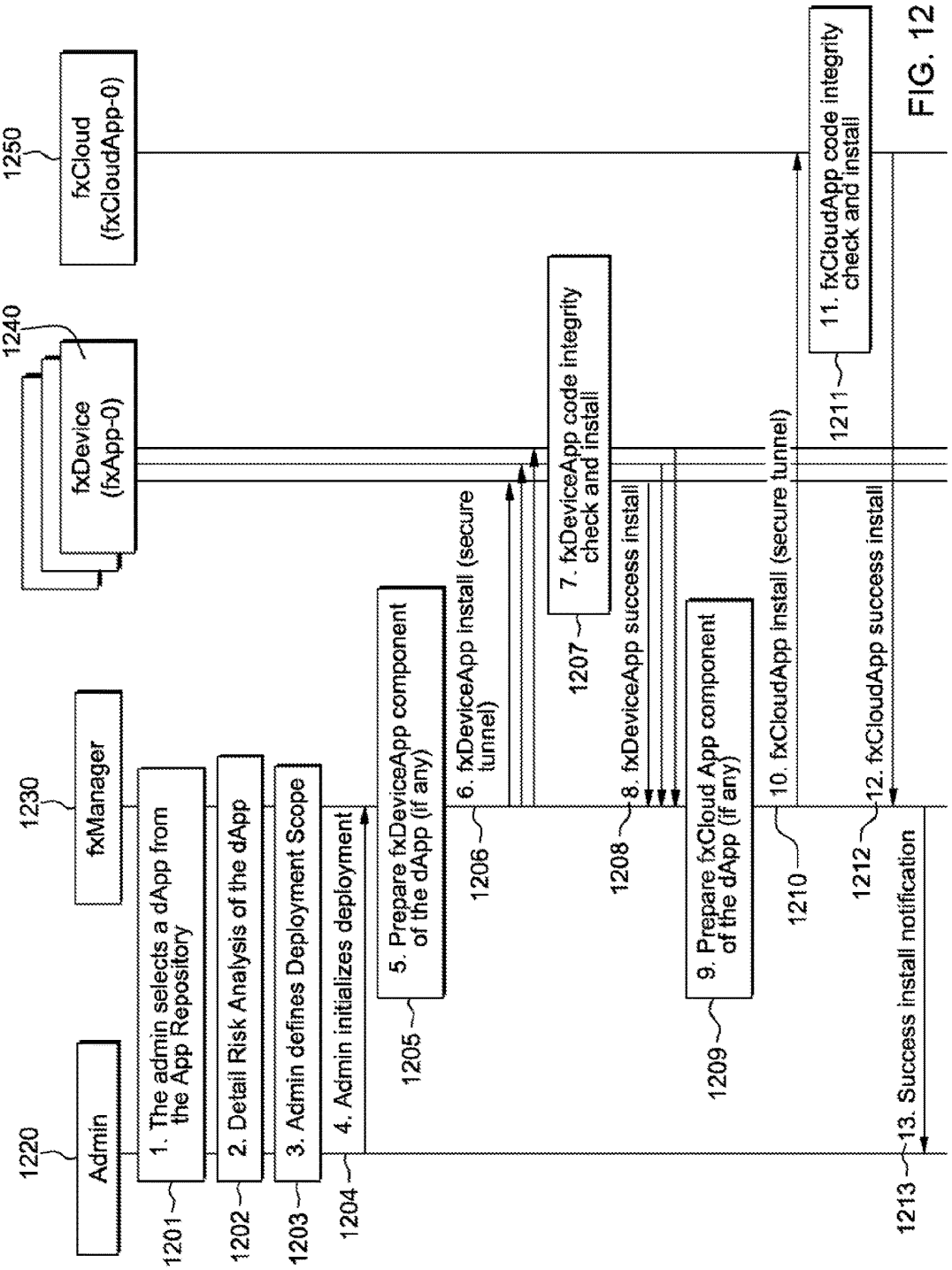

1220
Admin
1230
fxManager
1240
fxDevice (fxApp-0)
1250
fxCloud (fxCloudApp-0)
1201
1. The admin selects a dApp from the App Repository
1202
2. Detail Risk Analysis of the dApp
1203
3. Admin defines Deployment Scope
1204
4. Admin initializes deployment
1205
5. Prepare fxDeviceApp component of the dApp (if any)
1206
6. fxDeviceApp install (secure tunnel)
1207
7. fxDeviceApp code integrity check and install
1208
8. fxDeviceApp success install
1209
9. Prepare fxCloud App component of the dApp (if any)
1210
10. fxCloudApp install (secure tunnel)
1211
11. fxCloudApp code integrity check and install
1212
12. fxCloudApp success install
1213
13. Success install notification

FIG. 12

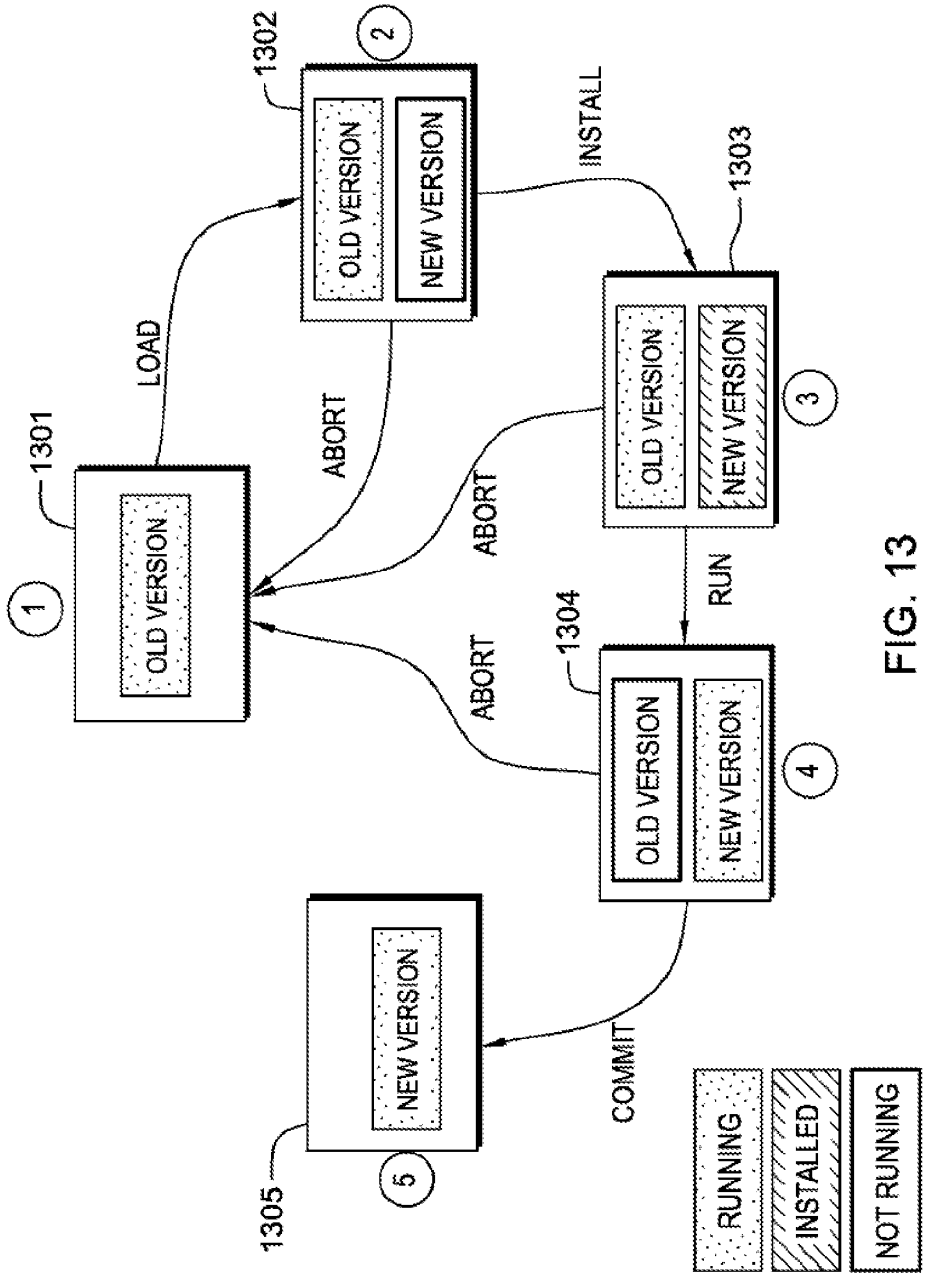

1301
OLD VERSION
1
LOAD
1302
OLD VERSION
NEW VERSION
2
INSTALL
1303
OLD VERSION
NEW VERSION
3
RUN
1304
OLD VERSION
NEW VERSION
4
COMMIT
1305
NEW VERSION
5
ABORT
ABORT
ABORT
RUNNING
INSTALLED
NOT RUNNING

FIG. 13

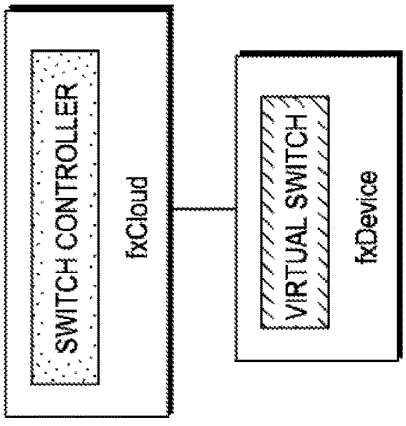


FIG. 14A

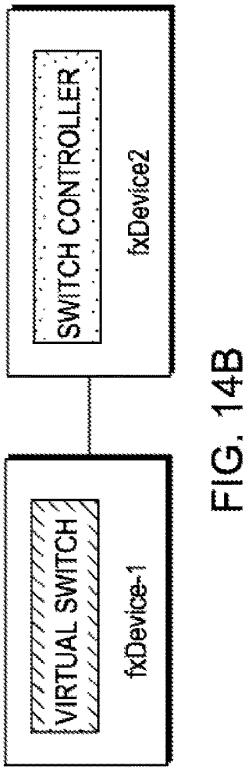


FIG. 14B

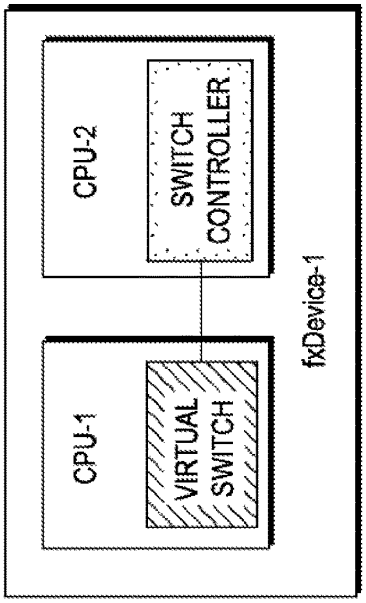


FIG. 14C

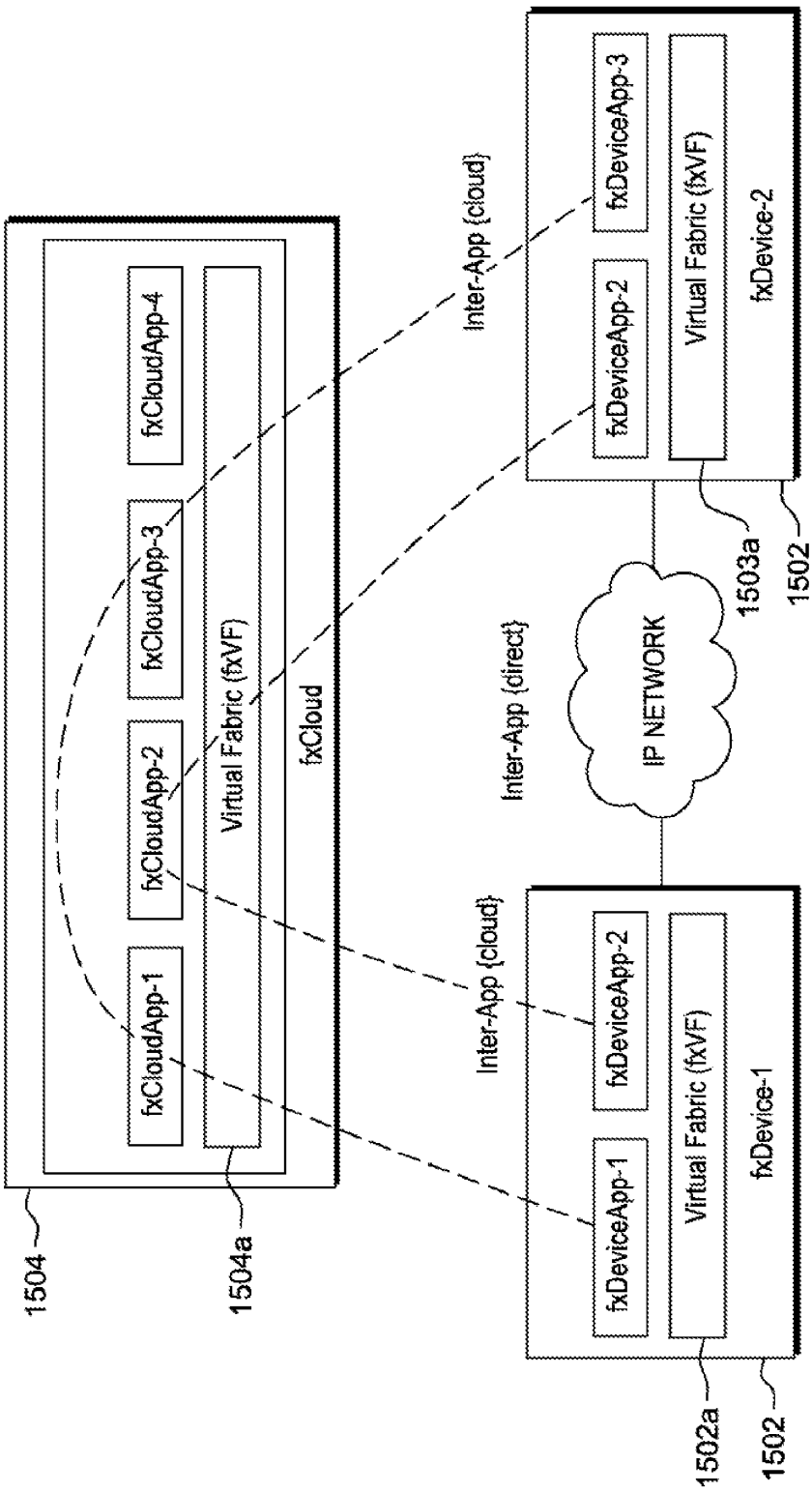

fxCloudApp-1
fxCloudApp-2
fxCloudApp-3
fxCloudApp-4
Virtual Fabric (fxVF)
fxCloud
1504
1504a
Inter-App {cloud}
Inter-App {direct}
IP NETWORK
fxDeviceApp-1
fxDeviceApp-2
Virtual Fabric (fxVF)
fxDevice-1
1502a
1502
fxDeviceApp-2
fxDeviceApp-3
Virtual Fabric (fxVF)
fxDevice-2
1503a
1502

FIG. 15

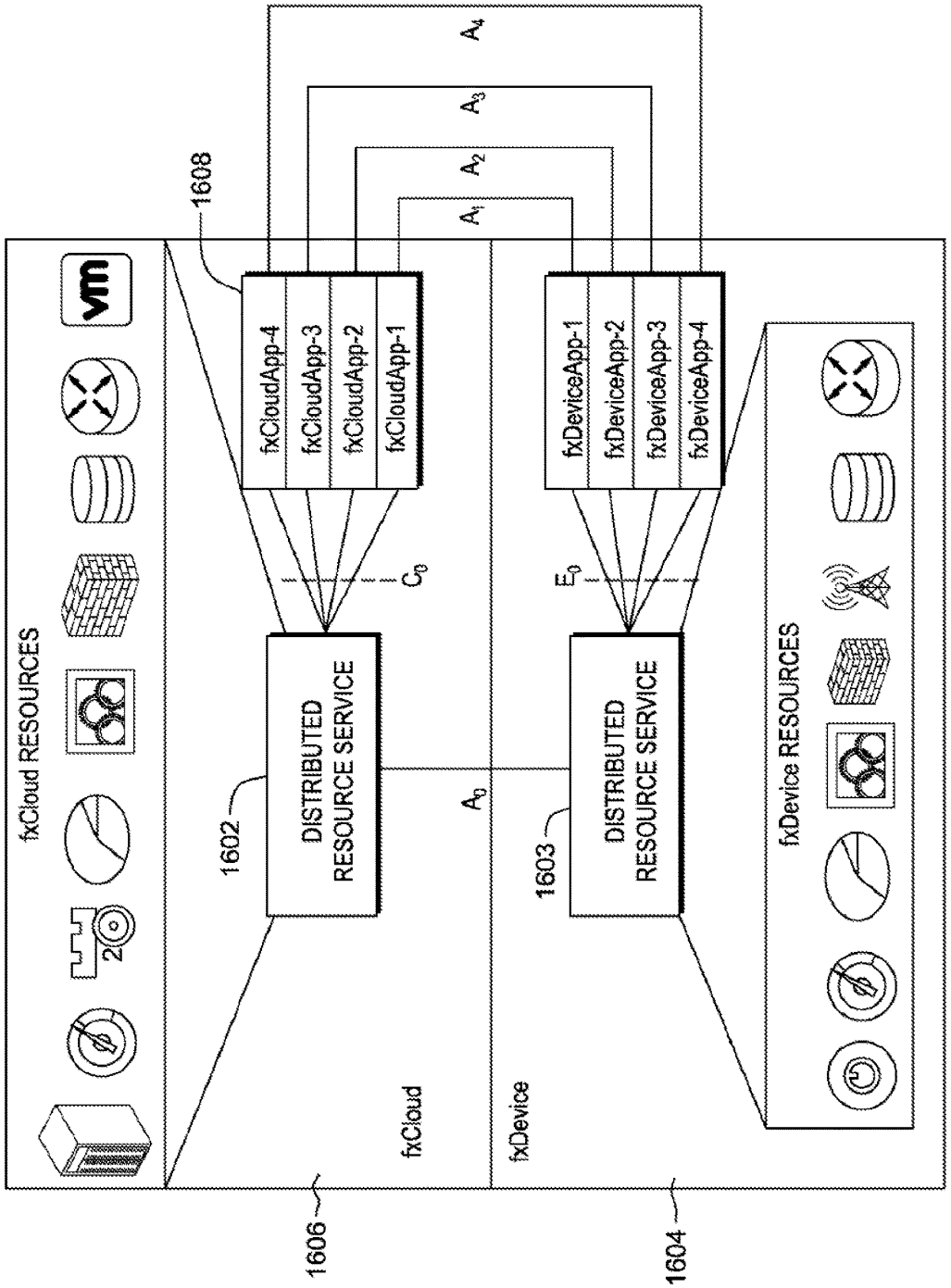

fxCloud RESOURCES
vm
1608
A4
A3
A2
A1
fxCloudApp-4
fxCloudApp-3
fxCloudApp-2
fxCloudApp-1
fxDeviceApp-1
fxDeviceApp-2
fxDeviceApp-3
fxDeviceApp-4
Cp
Ep
1602
DISTRIBUTED RESOURCE SERVICE
A0
1603
DISTRIBUTED RESOURCE SERVICE
fxDevice RESOURCES
fxCloud
fxDevice
1606
1604

FIG. 16

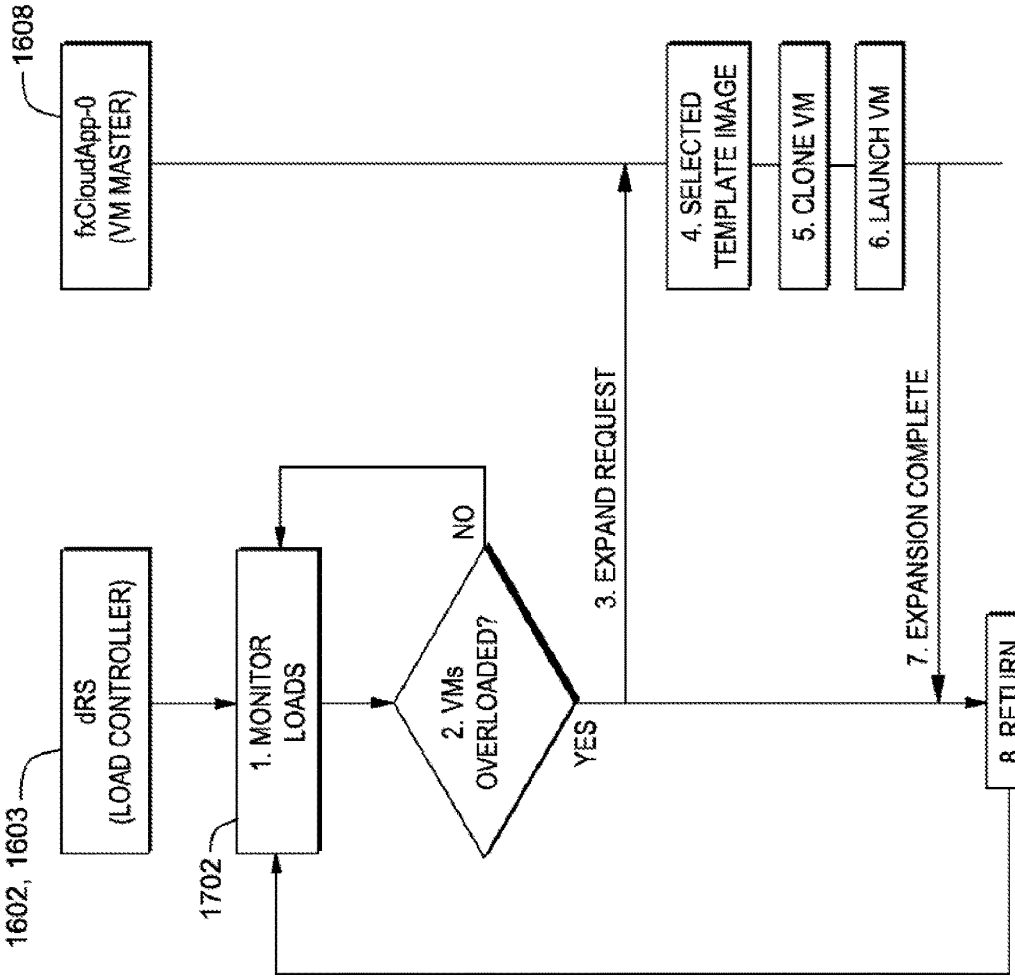

1608
fxCloudApp-0
(VM MASTER)
4. SELECTED
TEMPLATE IMAGE
5. CLONE VM
6. LAUNCH VM
3. EXPAND REQUEST
7. EXPANSION COMPLETE
dRS
(LOAD CONTROLLER)
1. MONITOR
LOADS
2. VMs
OVERLOADED?
NO
YES
8. RETURN
1602, 1603
1702

FIG. 17A

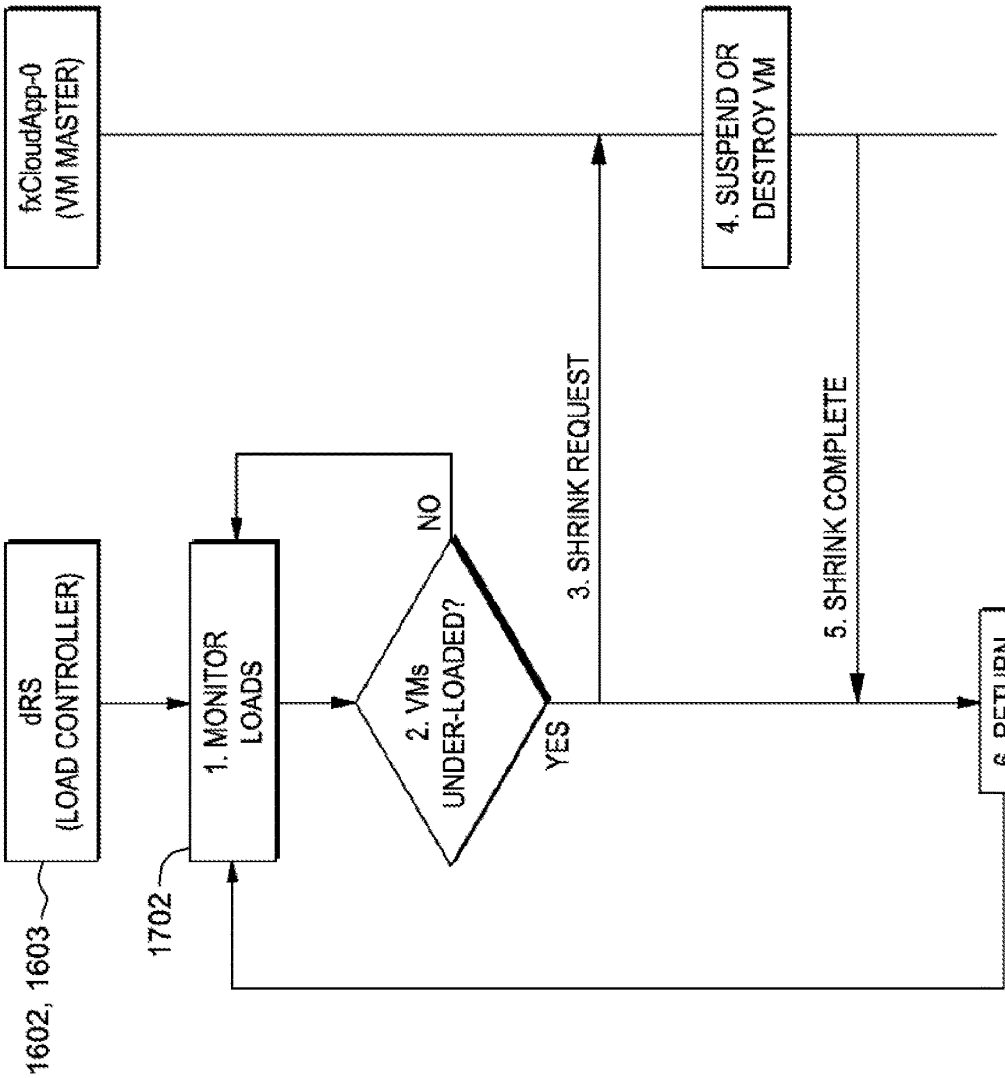

fxCloudApp-0
(VM MASTER)
4. SUSPEND OR
DESTROY VM
3. SHRINK REQUEST
5. SHRINK COMPLETE
dRS
(LOAD CONTROLLER)
1. MONITOR
LOADS
2. VMs
UNDER-LOADED?
NO
YES
6. RETURN
1602, 1603
1702

FIG. 17B

DISTRIBUTED SOFTWARE DEFINED NETWORKING

CROSS-REFERENCE TO RELATED PATENT APPLICATIONS

The present invention claims priority from U.S. Provisional Patent Application Ser. No. 61/834,807, filed Jun. 13, 2013 the disclosure of which is herein specifically incorporated by reference in its entirety.

FIELD OF THE INVENTION

This invention relates to a network architecture that facilitates secure and flexible programmability between a user device and across a network with full lifecycle management of services and infrastructure applications.

BACKGROUND

Existing network solutions are built on proprietary hardware and software. Network Operators & Information Technology (NOIT) can only configure the proprietary network infrastructure provided by vendors and are unable to add new and customized features or capabilities. As a result, the NOIT can only add new desired features by making such requests to their infrastructure vendors or pursuing standardization processes. But these existing approaches are time consuming and resource intensive.

SUMMARY OF THE INVENTION

An aspect of the disclosure herein is a plurality of network elements in a packet system, said network elements comprising: a flexible network device adapted to process data flows in the packet system having at least one fxDeviceApp; a flexible cloud platform having at least one fxCloudApp, wherein the at least one fxDeviceApp and at least one fxCloudApp are in secure communication to form a distributed application; and an application management portal which manages life cycle of the distributed applications.

Another aspect of the disclosure is a plurality of network elements in a packet system, said network elements comprising: a flexible cloud platform having a plurality of virtual machines, wherein each of the virtual machines has at least one fxCloudApp; and a flexible network device adapted to process data flows in the packet system having at least one fxDeviceApp, wherein the at least one fxDeviceApp and fxCloudApp are in secure communication to form a distributed application.

Another aspect of the disclosure is a plurality of network elements in a packet system, said network elements comprising: a flexible cloud platform having a plurality of zones, wherein each of the plurality of zones has at least one fxCloudApp; and a flexible network device adapted to process data flows in the packet system having at least one fxDeviceApp, wherein the at least one fxDeviceApp and fxCloudApp are in secure communication to form a distributed application.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** demonstrates the network architecture of a 4G system.

FIG. **2** illustrates a network element (or system) upon which the Distributed Software Defined Network operates.

FIG. **3** illustrates a Distributed Software Defined Network in accordance with the present disclosure.

FIG. **4** is an exemplary view of the software layers in the Flexible

Operation System (fxOS) Architecture.

FIG. **5** illustrates the FastPath architecture.

FIG. **6** presents a potential software layer of the fxOS co-existing alongside a general purpose OS.

FIG. **7** is an exemplary view of the fxStore Portal.

FIG. **8** demonstrates a shared fxCloud deployment.

FIG. **9** demonstrates a Zoned Deployment.

FIG. **10** demonstrates another deployment option as applied to cellular networks (especially Long Term Evolution (LTE)).

FIG. **11** shows an example of a signaling flow chart for the fxDevice Commissioning Procedure.

FIG. **12** shows an example procedure for Application Provisioning.

FIG. **13** presents the state machine for the application upgrade process.

FIGS. **14**A-**14**C are examples of the Distributed Switch Controller during Upgrade.

FIG. **15** shows how fxVF in the fxCloud and fxDevice.

FIG. **16** demonstrates the logical interfaces between the platform resources and applications.

FIGS. **17**A and **17**B demonstrate example procedures for cloud breathing procedures.

DETAILED DESCRIPTION

The Distributed Software Defined Network (dSDN) disclosed herein is an architecture that enables secure and flexible programmability across a network with full lifecycle management of services and infrastructure applications (fxDeviceApp). The dSDN also harmonizes application deployment across the network independent of the hardware vendor. As a result, the dSDN simplifies the network deployment lifecycle from concept to design to implementation to decommissioning.

DEFINITIONS, ACRONYMS & ABBREVIATIONS

The following terms, acronyms, abbreviations and descriptions are explained below and are used throughout the detailed description of the dSDN:

TERM	DESCRIPTION
2G	2nd Generation Cellular Technology
3G	3Rd Generation Cellular Technology
4G	4th Generation Cellular Technology
AP	WiFi Access Point
API	Application Programming Interface
AR	Application Repository
ASIC	Application Specification Integrated Circuit
AuC	Authentication Center
BS	Base Station
BSC	Base Station Controller
BTS	Base Station
CDN	Content Distribution Network
CN	Core Network
CNIE	Core Network Internet Edge
CPU	Central Processing Unit
dAP	Distributed Application Package
dApp	Distributed App
dCP	Distributed Content Service
dNS	Distributed Notification Service
DNS	Domain Name System

-continued

TERM	DESCRIPTION
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
dRS	Distributed Resource Service
DSCP	Differentiated Service Code Point (QoS)
E-UTRAN	Evolved UMTS Terrestrial Radio Access Network
EIR	Equipment Identity Register
eNB	Enhanced Node B (4G)
FIB	Flow Information Base
FQDN	Fully Qualified Domain Name
fxApp	FleXible Application, which has a fxDeviceApp and fxCloudApp component. fxDeviceApp and fxCloudApp of one application represent fxApp.
fxDeviceApp	FleXible Application which runs on fxDevice
fxBS	FleXible Base Station that is a BS that is powered by fxOS (an example of an fxDevice)
fxCloud	FleXible Cloud that interacts with fxDevice and on other network elements on the northbound interface
fxCloudApp	FleXible Cloud Application that is the application that runs on fxCloud and can be associated to one or more fxDeviceApp in fxDevice
fxManager	FleXible Manager that manager the fxOS lifecycle (provisioning, usage, de-provisioning)
fxOS	FleXible Operation System which run on the fxDevice as the OS and firmware
fxSDK:	FleXible Software Development Kit
fxSimulator	FleXible Simulator
fxStore	FleXible Store that presents fxApp to the network
administratorsGERAN	GPRS Edge Radio Access Network
GGSN	Gateway GPRS Support Node (used in core network of 2G/3G systems)
GMSC	Gateway Mobile Switching Center
GPRS	General Packet Radio Service
GPS	Global Positioning System
GTP	GPRS Tunneling Protocol
GW	Gateway
HAL	Hardware Abstraction Layer
HD	High Definition
HLR	Home Location Register
HSS	Home Subscriber Server
IAC	Inter Application Communication
IMS	IP Multimedia Sub-system
IMSI	International Mobile Subscriber Identifier
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
JVM	Java Virtual Machine
LA	Location Area
LAN	Local Area Network
LSP	Label Switching Protocol
M2M	Machine to Machine
MAC	Medium Access Control
MCP	Multi Chip Packaging
MIMO	Multi Input Multi Output
MME	Mobility Management Entity (used in core network of LTE)
MNO	Mobile Operator Network
MPLS	Multi Protocol Label Switching
MSC	Mobile Switching Center
MSISDN	Mobile Station
NAS	Non-Access Stratum
NB	Node B (3G)
NFC	Near Field Communication
NFV	Network Function Virtualization
NOIT	Network Operations & Information Technology; this term network broadly includes carriers, service providers, enterprise networks, and designated/contacted 3rd party administrators.
OS	Operating System
OTT	Over the Top
P-GW	Packet Gateway
PaaS	Platform-as-a-Service
PBX	Private Branch exchange (telephony)
PDG	Packet Data Gateway
PDN	Packet Data Network
PHY	Physical Layer
PLMN	Public Land Mobile Network
PnP	Plug and Play

-continued

TERM	DESCRIPTION
PP	Packet Processor
PR	Platform Resources
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RA	Routing Area
RAN	Radio Access Network of cellular networks
RAPI	Resources API
RFID	Radio Frequency Identifier
RNC	Radio Network Controller
S-GW	Serving Gateway
S1-AP	S1 Interface Application Protocol (an LTE protocol between MME and eNodeB)
SaaS	Software-as-a-Software
SDK	Software Development Kit
SDN	Software Defined Network
SGSN	Serving GPRS Support Node (used in core network of 2G/3G systems)
SIM	Subscriber Identity Module
SiP	System in a Package
SMS	Short Message Service
SoC	System on a Chip
SON	Self-Optimizing Network
SSDP	Simple Service Discovery Protocol
TA	Target Area
TNE	Test Network Environment
TTM	Time to Market
UMTS	Universal Mobile Telecommunication System
UPnP	Universal Plug and Play
URA	UMTS Routing Area
URI	Uniform Resource Identifier
UTRAN	UMTS Terrestrial Radio Access Network
VLR	Visitor Location Register
VM	Virtual Machine
VPN	Virtual Private Network
WAN	Wide Area Network
WLAN	Wireless Local Area Network
WLAN	Wireless Local Area Network
XMPP	Extensible Messaging and Presence Protocol

Database. One or more large structured sets of persistent data maintained upon a computer system organized and structured according to a software system defining rules for organization as well as responding to queries to read, write, or modify data as well as provide statistical information regarding the contained data. As used in this disclosure in describing the dSDN, a database may be either a single unified system or a distributed system wherein certain database elements are located upon different systems or servers which may be in different physical locations, acting in harmony to appear as one unified database. Where databases are described, it will be understood by one of ordinary skill in the art that (i) alternative database structures to those described may be readily employed, and (ii) other memory structures besides databases may be readily employed. Any illustrations or descriptions of any sample databases presented herein are illustrative arrangements for stored representations of information. The database formats may include relational databases, object-based models and/or distributed databases which could be used to store and manipulate the data types described herein. Likewise, object methods or behaviors of a database can be used to implement various processes, such as the described herein. In addition, the databases may, in a known manner, be stored locally or remotely from a device which accesses data in such a database.

The term "determining" and grammatical variants thereof (e.g., to determine a price, determining a value, determine an object which meets a certain criterion) is used in an extremely broad sense. The term "determining" encompasses a wide variety of actions and therefore "determining"

can include calculating, computing, processing, deriving, investigating, looking up (e.g., looking up in a table, a database or another data structure), ascertaining and the like. Also, "determining" can include receiving (e.g., receiving information), accessing (e.g., accessing data in a memory) and the like. Also, "determining" can include resolving, selecting, choosing, establishing, and the like. The term "determining" does not imply certainty or absolute precision, and therefore "determining" can include estimating, extrapolating, predicting, guessing and the like. The term "determining" does not imply that mathematical processing must be performed, and does not imply that numerical methods must be used, and does not imply that an algorithm or process is used.

The functionality and/or the features of a single device that is described may be alternatively embodied by one or more other devices which are described but are not explicitly described as having such functionality/features. Thus, other embodiments need not include the described device itself, but rather can include the one or more other devices which would, in those other embodiments, have such functionality/features.

Devices that are described as in "communication" with each other or "coupled" to each other need not be in continuous communication with each other or in direct physical contact, unless expressly specified otherwise. On the contrary, such devices need only transmit to each other as necessary or desirable, and may actually refrain from exchanging data most of the time. For example, a machine in communication with or coupled with another machine via the Internet may not transmit data to the other machine for long period of time (e.g. weeks at a time). In addition, devices that are in communication with or coupled with each other may communicate directly or indirectly through one or more intermediaries.

Although process (or method) steps may be described or claimed in a particular sequential order, such processes may be configured to work in different orders. Further, some steps may be performed simultaneously despite being described or implied as occurring non-simultaneously (e.g., because one step is described after the other step) unless specifically indicated. Moreover, the illustration of a process by its depiction in a drawing does not imply that the illustrated process is exclusive of other variations and modifications thereto, does not imply that the illustrated process or any of its steps are necessary to the embodiment(s), and does not imply that the illustrated process is preferred. Where a process is described in an embodiment the process may operate without any user intervention.

It will be readily apparent to one of ordinary skill in the art that the various processes of the dSDN described herein may be implemented by, e.g., appropriately programmed general purpose computer(s), special purpose computer(s) and computing device(s). Typically a processor (e.g., one or more microprocessors, one or more microcontrollers, one or more digital signal processors) will receive instructions (e.g., from a memory or like device), and execute those instructions, thereby performing one or more processes defined by those instructions. Instructions may be embodied in, e.g., one or more computer programs, one or more scripts.

A "processor" means one or more microprocessors, central processing units (CPUs), computing devices, controllers, microcontrollers, digital signal processors, or like devices or any combination thereof, regardless of the architecture (e.g., chip-level multiprocessing/multi-core, Reduced Instruction Set Computer (RISC), Complex Instruction Set Computer (CISC), Microprocessor without Interlocked Pipeline Stages, pipelining configuration, or simultaneous multithreading).

Further, programs that implement methods described herein may be stored and transmitted using a variety of media (e.g., computer readable media) in a number of manners. In some embodiments, hard-wired circuitry or custom hardware may be used in place of, or in combination with, some or all of the software instructions that can implement the processes of various embodiments. Thus, various combinations of hardware and software may be used instead of software only to implement the embodiments.

The term "non-transitory computer readable medium" in this disclosure refers to any medium, a plurality of the same, or a combination of different media, that participate in providing data (e.g., instructions, data structures) which may be read by a computer, a processor or a like device. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media may include, for example, optical or magnetic disks and other persistent memory. Volatile media may include dynamic random access memory (DRAM), which typically constitutes the main memory. Transmission media may include coaxial cables, copper wire and fiber optics, including the wires that comprise a system bus coupled to the processor. Transmission media may include or convey acoustic waves, light waves and electromagnetic emissions, such as those generated during radio frequency (RF) and infrared (IR) data communications. Common forms of non-transitory computer-readable media in which the dSDN may be implemented include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, any other magnetic medium, a CD-ROM, DVD, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a Random Access Memory (RAM), a programmable read only memory (PROM), an erasable programmable read only memory (EPROM), a flash electrically erasable programmable read only memory (FLASH-EEPROM), any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media in this disclosure may be involved in carrying data (e.g. sequences of instructions) to a processor. For example, data may be (i) delivered from RAM to a processor; (ii) carried over a wireless transmission medium; (iii) formatted and/or transmitted according to numerous formats, standards or protocols, such as Ethernet (or IEEE 802.3), Bluetooth, and Transmission Control

Protocol/Internet Protocol ("TCP/IP"), Time Division Multiple Access (TDMA), Code Division Multiple Access (CDMA), and 2G/3G; and/or (iv) encrypted to ensure privacy or prevent fraud in any of a variety of ways well known in the art.

Thus in some embodiments in this disclosure a description of a process may be a description of a non-transitory computer-readable medium storing a program for performing the process. The computer readable medium may store (in any appropriate format) those program elements which are appropriate to perform the method.

In an embodiment, a server computer, network element or centralized authority may not be necessary or desirable. For example, an embodiment may be practiced on one or more devices without a central authority. In such an embodiment, any functions described herein as performed by the server

computer or data described as stored on the server computer may instead be performed by or stored on one or more such devices.

In interpreting the present application (which includes the claims), one of ordinary skill in the art shall refer to the prosecution history of the present application, but not to the prosecution history of any other patent or patent application, regardless of whether there are other patent applications that are considered related to the present application, and regardless of whether there are other patent applications that share a claim of priority with the present application.

Numerous embodiments are described in the present application, and are presented for illustrative purposes only. The described embodiments are not, and are not intended to be, limiting in any sense. One of ordinary skill in the art will recognize that the disclosed embodiment(s) may be practiced with various modifications and alterations, such as structural, logical, software, and electrical modifications. Although particular features of the disclosed invention(s) may be described with reference to one or more particular embodiments and/or drawings, it should be understood that such features are not limited to usage in the one or more particular embodiments or drawings with reference to which they are described, unless expressly specified otherwise.

The present disclosure is not a literal description of all embodiments of the invention(s). Also, the present disclosure is not a listing of features of the invention(s) which must be present in all embodiments. A description of an embodiment with several components or features does not imply that all or even any of such components/features are required. On the contrary, a variety of optional components are described to illustrate the wide variety of possible embodiments of the present invention(s).

The following description has been presented for purposes of illustration and description and is not intended to be exhaustive or to limit the embodiments to the precise form disclosed. Many modifications and variations are possible in light of the teachings disclosed herein.

The embodiments were chosen and described to explain principles of operation and their practical applications. However, the scope of the invention is to be defined by the claims.

The major technical reason for inflexibility in network infrastructure has been related mostly to the proprietary hardware and rigid software architecture in the existing network elements (routers, switches, gateways, cellular base-stations, WiFi access points, etc.). With recent advancements in semiconductor device manufacturing, there is no longer a cost barrier to add computing power to many electronics in the market including networking infrastructure. The addition of such compute capabilities would open up new opportunities for programmable platforms in the network and ultimately creating more flexible network architecture and business models.

The Distributed Software Defined Network (dSDN) disclosed herein is an end-to-end architecture that enables secure and flexible programmability across a network with full lifecycle management of services and applications (fxApp). The dSDN also harmonizes fxApp deployment across the network independent of the hardware vendor. As a result, the dSDN simplifies the network deployment lifecycle from concept to design to implementation to decommissioning. In this disclosure, the dSDN is applied to the wireless networks as an exemplary embodiment. However, the dSDN is not limited to wireless networks embodiments and it could be applied to many other network types including enterprise, wireline service providers, data centers, and Internet of Things (IoT).

FIG. **1** demonstrates the network architecture of a 4G system **100**. (In alternative embodiments, the system and method disclosed may also be implemented in a 2G/3G network as well). In a cellular network, the base station or BS (denoted as BTS in 2G, NodeB or NB in 3G, and eNodeB or eNB in 4G systems) is responsible for termination of a wireless link (air-interface) to the mobile device (also known as the User Equipment (UE)). The BS then transports the user data to the core network and communicates control-signaling messages with the core network. In a typical cellular network, there are thousands of BS's covering nations. Each BS covers an area known as a cell. As the demand for mobile Internet expands, the mobile network operator (MNO) needs to deploy smaller cells so it can reuse the frequency more often.

A BS is made up of the following main functional blocks: 1) radio frequency (RF) front end; 2) clock; 3) baseband; 4) power manager; and 5) central processing unit (CPU). Some BS's also have a dedicated packet processor (PP) to accelerate packet processing in hardware. BS vendors may also use custom or merchant System-on-a-Chip (SoC) or Multi-Chip-Packaging (MCP) solutions to combine various functions into a single chip (e.g., PP, CPU, and baseband).

Even though there has been attempts to open up various interfaces in the BS (such as the Open BS Architecture Initiative that defines interfaces between these functional modules in the BS), there has been very little to no efforts to unify the programmability of the BS itself. As a result, the MNOs suffer from the following difficulties. First, the core network Internet edge (CNIE), Packet Data Gateway (PDG) and the surrounding functions are becoming extremely complicated, non-scalable, and expensive. Second, each MNO needs certain customization and feature sets. Currently, they depend on their vendors for these customizations, which could cost the MNO tremendously both financially and in regards to time-to-market (TTM); hence, hindering innovation. In addition, usually these features are put into standards or in the vendor's feature set, thus eliminating the MNO differentiation against the other MNOs. Third, if the MNO is multi-vendor in their CNIE, coordinating all the vendors to implement the same features is usually a difficult and a time-consuming effort. As a result, more add-on network appliances are introduced into the CNIE which adds to the network management complexity. Fourth, as a BS becomes Internet Protocol (IP) based, there is more visibility into the user traffic types and new innovative opportunities are missed for creative traffic shaping features, backhaul bandwidth optimization, prioritization at radio edge, power management algorithms, etc. Fifth, many features implemented in CNIE present suboptimal performance. For example, the filtering enforcement at CNIE is fundamentally inefficient since the packets have to travel all the way through the expensive air-interface, the backhaul, various other core network elements and transport networks to get to the CNIE. Sixth, the Radio Access Network (RAN) deployments are a very expensive endeavor for MNOs. The current rigid BS designs limit innovation and force the MNOs to undergo major upgrades every few years.

The problems stated above apply to many different types of networks even though here they are presented in cellular networks as a focus. In the present disclosure, the dSDN exhibits a new paradigm in the software programmability of networks that would address the problems above and enable many more advanced features.

The dSDN may be located at a network element (or system) **200** which is shown in detail in FIG. **2** or in alternative embodiments the functions of the dSDN may be divided among a plurality of network elements (or systems) which are similar in structure to network element **200**. Each network element **200** may comprise one or more system control logic **202** coupled with at least one or a plurality of processor(s) **204**, system memory **206**, a network interface **208** (including a transceiver **208***a*), and input/output (I/O) devices **210**. The processor(s) **204** may include one or more single-core or multi-core processors. The processor(s) **204** may include any combination of general-purpose processors and dedicated processors (e.g., graphics processors, application processors, baseband processors, etc.). System control logic **202** for one embodiment may include any suitable interface controllers to provide for any suitable interface to at least one of the processor(s) **204** and/or to any suitable device or component in a packet network in communication with system control logic **202**. System control logic **202** for one embodiment may include one or more memory controller(s) to provide an interface to system memory **206**. System memory **206** may be used to load and store data and/or instructions, for example, for network element **200**. System memory **206** for one embodiment may include any suitable volatile memory, such as suitable dynamic random access memory (DRAM), for example. System memory **206** may also include non-volatile memory including one or more tangible, non-transitory computer-readable media used to store data and/or instructions, for example, such as the embodiments described herein with regard to the dSDN. The non-volatile memory may include flash memory, for example, and/or may include any suitable non-volatile storage device(s), such as one or more hard disk drive(s) (HDD(s)), one or more compact disk (CD) drive(s), and/or one or more digital versatile disk (DVD) drive(s). The system memory **206** may include a storage resource physically part of a device on which the network element **200** is installed or it may be accessible by, but not necessarily a part of, the device. For example, the system memory **206** may be accessed over a network via the network interface **208** and/or over Input/Output (I/O) devices **210**. Network interface **208** may include a transceiver **208***a* to provide a radio interface for network element **200** to communicate over one or more network(s) and/or with any other suitable device. Network interface **208** may include any suitable hardware and/or firmware. The network interface **208** may include a plurality of antennas to provide a multiple input, multiple output radio interface. Network interface **208** for one embodiment may include, for example, a wired network adapter, a wireless network adapter, a telephone modem, and/or a wireless modem. For one embodiment, at least one of the processor(s) **204** may be packaged together with logic for one or more controller(s) of system control logic **202**. For one embodiment, at least one of the processor(s) **204** may be integrated on the same die with logic for one or more controller(s) of system control logic **202**. In various embodiments, the I/O devices **210** may include user interfaces designed to enable user interaction with the network element or system **200**, peripheral component interfaces designed to enable peripheral component interaction with the network element or system **200**, and/or sensors designed to determine environmental conditions and/or location information related to the network element or system **200**. In various embodiments, the peripheral component interfaces may include, but are not limited to, a non-volatile memory port, a universal serial bus (USB) port, an audio jack, and a power supply interface.

dSDN Architecture

In the dSDN architecture, the network features are virtualized and may be distributed across separate network elements that cooperate together to create an advanced, programmable, and scalable network. In an alternative embodiment, the network features may be located at a single network element. A high-level overview of a dSDN system **300** is depicted in FIG. **3**. The dSDN system **300** consists of a flexible network device (fxDevice) **302**, a flexible cloud platform (fxCloud) **304**, an application management portal (fxManager) **306**, and an infrastructure application market place (fxStore) **308**. The fxDevice **302** is a network device that is powered by a sandboxing operating system named flexible operating system (fxOS) **302***a*. The sandboxing operating system ensures each application runs as a dedicated process in sure isolation from the other applications. The fxOS **302***a* may be built based on an existing sandboxed operating system (OS) (e.g., Android®) by extending several aspects of such an OS with routing/networking layers and supports for operating as the wireless infrastructure. The fxDevice **302** may host several independent and securely isolated applications (on top of fxOS **302***a*) named here fxDeviceApp **302***b*.

In the dSDN system, fxDeviceApp **302***b* may have a sister app in the backend cloud infrastructure (i.e., flexible cloud platform **304**) referenced here as fxCloudApp **304***a*. The fxCloudApp **304***a* in the cloud is paired with its fxDeviceApp **302***b* in the fxDevice **302**. The fxCloudApp **304***a* and the fxDeviceApp **302***b* collectively form a distributed application (dApp or fxApp). In this description, when fxDeviceApp is referenced it shall mean any application that may have software components in the fxDevice, fxCloud, or both. In general, an fxApp package may include the following software components:

fxDeviceApp binary;
fxCloudApp binary;
Manifest files; and
Signatures.

It is important to note that the fxDeviceApp **302***b* and fxCloudApp **304***a* could use any protocol to communicate with each other and there is no need for standardizing this communication allowing for ultimate freedom for the developers. This allows for the system to operate in a loosely coupled autonomous fashion allowing for asynchronous communication in a distributed fashion.

The fxApps lifecycle is managed via fxManager **306**. The fxManager **306** presents a user-friendly portal to the network administrator (admin) to discover, test prior to deployment, provision, and deprovision dApps. Using fxManager **306**, the admin may discover new fxApps from the fxStore **308**, which presents all the tested and certified vApps as well as showing the supported fxOS version, support hardware platforms, and other information such as reviews and number of commercial deployments by all NOIT customers.

fxOS

fxDevice **302** is defined as any networking equipment embedded with a special flexible operating system (fxOS) **302***a*. The purpose of fxOS **302***a* is to enable a carrier-class programmable platform. The fxOS **302***a* is configured to do, but is not limited to, the following. First, since fxOS **302***a* is designed for network equipment, it allows for efficient execution and running all the time. A plurality of power save modes may be implemented to ensure energy saving schemes without hindering the functionality of the fxDevice

302. Second, it enables simultaneous execution of several flexible applications (fxDeviceApp **302***b*) on the fxDevice **302**. Third, it allows for application of a legacy compiled binary or byte-code of another language (for example, Java or Android®) in addition to fxDeviceApps **302***b*. Fourth, it allows FastPath processing for data plane packets. The packet forwarding in this case separated from the control plane operations. The FastPath engine allows integration with specialized hardware accelerators or packet processors (PP). The packet forwarding may be implemented with software/kernel enhancements all in software. Another FastPath implementation may use both software/kernel enhancement and hardware packet processors (PP). Fifth, the fxOS **302***a* is capable of Deep Packet Inspection (DPI) and identifying all the flows and their related protocols, services and present this information in a format useful to application developers. The fxOS **302***a* can act on the data flows as the directed by the application developer. Sixth, applications are securely isolated from each other and to communicate with them is completely configurable by policies defined by the admin. Seventh, resource utilization of each application is policed such that an application may not be able to exceed its usage. For example, an application is pre-allocated the memory size, cache/storage space, and CPU cycles. Eighth, the applications may be signed with unique certificate security keys. The security keys and certificates may be allocated and/or signed by a certificate authority. The fxDevice **302** may be protected by validating signed applications to run or to install in the device. Ninth, it allows hot upgrade of the software and applications on the platform with little to no interruption to the operational aspect of the platform and its applications. Tenth, the fxOS **302***a* may boot by using the secure code from the network servers (cloud-booting). This would ensure tamper-proof software foundation for the fxDevice **302**. Eleventh, there may be a secure and programmable firewall in the fxOS **302***a*. This embedded firewall protects against attacks to the fxDevice **302** and network attached to the network. The fxOS **302***a* firewall may be configured by the developer and it does allow replacing of attack detection algorithms (e.g., man-in-the-middle detection algorithm). The fxOS **302***a* may allow secure caching of user data, web pages, and media files (video and audio) in the fxDevice **302**. The secure connections to the cloud (fxCloud **304**) can enable sharing and control of the caching between the fxCloud **304**, fxDevice **302**, or the rest of the network. Twelfth, the fxOS **302***a* can support multi-threading and multi-core CPUs. This feature may dynamically distribute the threads to various CPU core without developers assistance.

The fxOS **302***a* allows secure connection to the fxCloud **304**. This secure connection allows communication between fxDeviceApps **302***b* in the fxDevice **302** and fxCloudApp **304***a* in the fxCloud **304**. The fxOS **302***a* controls the access of applications to platform APIs. An application is categorized into an Access Level (AL) based on which the fxOS **302***a* decides that the application does not exceed its AL and access APIs that it is not authorized. The fxCloud **304** and the fxOS **302***a* create a virtual fabric (fxVF) for messaging between applications. The actual message can travel directly between two fxDevices or it may traverse the fxCloud **304**. The messaging complexity may be abstract for the developer.

FIG. **4** demonstrates an examplary view of the software layers in the fxOS **302***a*. Each layer of this software stack is modified and redefined to optimize for embedded networking equipment of wireless and wireline nature. Starting from the bottom of the stack, the kernel also connects to the hardware resources via the device drivers. Examples of such resources are wireless baseband SoC **402**, hardware packet processor **404**, Wide Area Network (WAN) uplink interface **406**, cache and storage **408**, and multi-core CPU **410**. The OS kernel **412** is enhanced with the FastPath function **414** to accelerate packet processing (by help of hardware features such as Intel's Data Plane Development Kit (DPDK) or pure software enhancements) and other related data path functions. The enhanced kernel with FastPath **412**, **414** would enable visibility into the data flows, sniffing the Non-Access Stratum (NAS) signaling (which is the signaling between the UE and the Core Network and the RAN is supposed to be transparent to), policy enforcements, regular packet forwarding functions, collection of flow statistics into Flow Information Base (FIB), and encryption/decryption functions.

Above the kernel, there are several tools **416**, native daemons **418**, native libraries **420** and Hardware Abstraction Layers (HAL) **422**. These tools abstract out the hardware dependencies for the upper layers and programmers. The Runtime **424** is an embedded virtual machine capable of securely isolating and executing the applications. System Services **426** are a set of services always running and available to the developers (e.g., timing and messaging services).

fx-Framework Library **428** consists of several Frameworks. The new frameworks are the library extensions introduced explicitly for the dSDN framework. Each of these frameworks adds a set of methods of (functions) and data structures for the following examples:

Wireless Manager **428***a*: manipulation, monitoring, and configuration of the wireless interfaces;

System Manager **428***b*: configuration, provisioning, and changing major settings of the fxDevice **302**;

Flow Controller **428***c*: allows for manipulation of data path, tunneling, switching, and optionally coordinate this activity with fxCloud **304**;

Energy Management **428***d*: enables fine-grained control over various elements in the fxDevice **302** that could be moved into various power stares, for example: ON, OFF, SLEEP, SUSPEND;

Big Data Manager **428***e*: manipulation, processing, and organization of data collected from various elements. This framework could enable close coordination between the fxDevice **302** and the fxCloud **304**;

Content Manager **428***f*: for data sharing (database sharing) between the applications within the fxDevice **302** or the fxCloud **304** and between applications in fxDevice **302** and fxCloud **304**. The Content Manager **428***f* may use URI to address the contents to be shared. The Content Manager **428***f* allows access to the application data according to the policy settings and permissions;

Policy Manager **428***g*: responsible for execution and implementation of polices set up by the admin. The application developers could use this framework to perform queries about the permissions, limitations, and rules;

Security Manager **428***h*: allows for access to the security protocol libraries and rewriting some of security algorithms such as the man-in-the-middle detection algorithm;

General Framework: general compute and logic building that may be inherited from existing general OS frameworks.

Using the frameworks, there are potentially at least two types of application types that are possible. First, the System Native Applications **430** that are provided as the initial application load into the platform. These applications could

be used by other applications by links or API. Second, the fxDeviceApps **302***b* that are written by third party developers or the NOIT.

FIG. **5** further details the FastPath **414** architecture. In this approach, the FastPath **414** is divided into the Virtual Switch (data-plane) **502** and the Switch Controller (control-plane) **504**. Such separation makes possible flexible deployment wherein the function of the Switch Controller **504** could move to the fxCloud **304** by an admin decision or by the dSDN system intelligence for example in the case of failure, redundancy, stateful reboot, or Hot Upgrade. The Virtual Switch **502** may include several functions and may interact with the Switch Controller **504** in the following exemplary ways:

Firewall **502***a*: implements the policies given to it by the firewall logic **504***a* in the fxOS Daemons.

Flow/NAS Discover **502***b*: continuously monitors data packets and discovers new flows and NAS signaling. It would then report it to the DPI Logic **504***b* where the flow intelligence resides and matching to meta-data happens. DPI Logic **504***b* in turn updates the Flow Information Base Information (FIB) **502***c* of the Virtual Switch **502**.

FastPath & DPI Processing **502***d*: forwards, routes packets, and performs DPI functions as instructed by the FIB **502***c*. It also collects statistics of packets and reports them to the Monitoring Logic **504***c*.

Policy Enforcement **502***e*: this function enforces various policies on the traffic such as QoS marking according to the policies received from the Policy Logic **504***d*.

Shaper **502***f*: this function could shape (queue) the traffic according to the available bandwidth on the uplink or WAN interface.

Virtual Fabric (fxVF) **504***e*: enables transparent switching of application and system messages of a dApp between the fxCloud **304** and the fxDevice **302** and between different dApps. The fxVF **504***e* is an abstraction layer that hides most of the complexities of messaging from the developers. The fxVF **504***e* uses the policies and firewall rules to enforce security and policies.

The fxOS **302***a* is mostly used for networking applications. There are some potential use cases where there is a need for a general purpose OS to run the legacy software applications with no need of recompilation. An example of a double OS situation may be where a digital signage (using a general purpose OS such as Android®) in a shopping mall also acts as a smart small-cell (using the fxOS **302***a*). In such a solution, any existing Android® application (for example) could be loaded in the fxDevice **302** and in parallel networking applications (fxDeviceApps **302***b*) could run on the same fxDevice **302**.

FIG. **6** presents a potential software layer of the fxOS **302***a* co-existing alongside a General Purpose OS **602**. Another implementation may be a standalone fxOS without the company of another general OS. It is important to note that the shown co-existence allows reuse of the kernel so there would be no need to report the drivers of hardware resources twice.

fxCloud

The fxCloud **304** is an integral part of the dSDN framework. The fxCloud **304** connects to various parts of the fxDevice **302** to form a single virtual view of the system. Examples of the main fxCloud **304** responsibilities include the following. First, it interacts and manages all applications, firmware, and fxOS **302***a* in the fxDevice **302**. Second, it secures isolation environments for distributed apps. Third, it directs the Virtual Machine lifecycle management including load-linked VM creations and destruction (Cloud Breathing). Fourth, the fxCloud **304** executes instructions defined by the fxManager **306**. Fifth, it implements uniform policy (including security) distribution and execution across dSDN **300** (including the fxDevice **302**, fxDeviceApps **302***b*, fxCloudApps **304***a*, and dApps). Sixth, Plug-and-Play commissioning of ihefxDevice **302** introduction and network expansion. Seventh, Distributed Resource Service (dRS) by which distributed applications can seamlessly communicate and message to each other using published APIs or proprietary methods within the pre-defined policies. Eighth, security violation discovery and alert system. Ninth, translation and/or exposure of Resources APIs (RAPI) to the systems outside the dSDN **300** within pre-defined policies. Tenth, fxCloud **304** in coordination with the fxDevice **302** creates a Distributed Notification Service (dNS) across the dSDN **300** by which the applications could be notified of an event to wakeup and/or respond to the event. Eleventh, fxCloud **304** in coordination with the fxDevice **302** creates a Distributed Content Management Service (dCMP) which provides a virtualized and distributed database system for applications to share application-specific or system-specific information with the pre-defined policies. Twelfth, the Switch Controller **504** function in the fxCloud **304** could take control of the controller function of the fxOS **302***a* in the fxDevice **302**. This would allow controller switching from local to remote and back. Examples of such a configuration is a temporary reason where the fxOS Switch Controller **504** may not be available during reboot or software upgrade.

In the discussion below, these services and features are expanded upon.

fxDeviceApp-0 & fxCloudApp-0

The fxDevice **302** and fxCloud **304** have embedded System Applications categorized as fxDeviceApp-0 and fxCloudApp-0, respectively. The fxDeviceApp-0 is responsible for all default and core functional capabilities of the fxDevice **302** as listed below:

Firmware of the fxDevice;

Default communication protocols (e.g. LTE Sl-AP, baseband software, security protocols, GTP, DNS, tunneling protocols);

Message communication protocol between the fxDevice and fxCloud;

Initial fxDevice settings;

Boot Loader;

fxDevice X.509 digital certificate;

OS and Kernel upgrades;

Upgrade and commissioning agents for the fxDeviceApp-0 upgrades and related procedures;

Establishing and initiating the message communication protocol between the fxDevice and fxCloud;

General fxDeviceApp install and upgrades;

Access to fxDevice hardware and software information.

The fxCloudApp-0 is responsible for all default and core functional capabilities of the fxCloud **304** such as:

Upgrade and commissioning agents for the fxCloudApp-0 upgrades and related procedures;

Hypervisor upgrades;

Virtual Machine lifecycle management (e.g. creation, suspension, destruction);

fxCloud X.509 digital certificate management;

OS and kernel upgrades;

General fxCloudApp install and upgrades;

Default communication protocols (e.g. security protocols, DNS, tunneling protocols).

Each software component can be managed and updated individually. To simplify the management of these software components, each software components can be sub-numbered such as fxDeviceApp-0.1, fxDeviceApp-0.2, fxCloudApp-0.1, fxCloudApp-0.2, etc.

fxManager

The network administrator may use the fxManager **306** to control much of the dSDN system **300**. The fxManager portal could use HTML5, for example, as a frontend technology. The major functions of the fxManager **306** may include the following.

fxDevice Management:

Commissioning (plug-and-play), decommissioning, firmware/OS upgrading/downgrading;

Suspend/resume functions;

Unification of zones and partitions for example in case of network merger;

Detailed management of fxDevices (remote secure login) where the administrator may zoom into a particular or a group of fxDevices to monitor and set management alarms;

Security policy setting.

Application lifecycle management: fxManager **306** may work, coordinate, and/or perform these tasks in conjunctions with fxCloud **304** for:

Provisioning, de-provisioning, upgrade/downgrade applications.

Some applications may only have a fxDevice component (Device-Centric Apps), may only have fxCloud component (Cloud-Centric Apps), or may have both fxDevice and fxCloud components (Hybrid Apps);

Creating a Test Network Environment (TNE) where the administrator can test certain configuration and applications without impacting the production networks;

Uploading applications developed by the NOIT or their partners into the Application Repository (AR). The AR is placeholder where the application packages are downloaded and stored for testing and verification prior to installation into the live dSDN system. An application could have several components (e.g. fxDeviceApp binaries, fxCloudApp signatures, manifest files, signatures) bundled together to form the Distributed Application Package (dAP);

Browsing through the certified applications in the fxStore and downloading them to the AR;

Application license accounting services by which a secure and accurate accounting of the installed applications are performed regularly. This inventory management function allows the administrator and the 3rd party ISV to account for live and active instants of the software in the network;

Application review tool where the network admin could use to review feedbacks and comments of other NOIT (maybe anonymous reviews) and could input his/her reviews for others (maybe anonymous reviews).

fxCloud Management:

Zoning and partitioning the network for administrative purposes. Zones and partitions could optionally put into different Virtual Machine (VM);

VM load management and Cloud Breathing policies;

The configuration conflict resolutions. Some configurations applied to the resources may conflict with each other (e.g., one firewall rule may state drop Netflix packets explicitly and another rule may state allowing Netflix explicitly). The fxCloud resource manager could monitor these configurations and discover the conflicts and signal those to the network administrator.

Security Management:

Security policy setting;

Inter-App communication setting;

Security of communication between fxDevice and fxCloud;

API Access Level (AL) definitions and (re-)assignment to the applications (permissions);

Risk Analysis of the applications;

Certificate and signature verification of the applications;

Content sharing between applications and associated policies.

An fxManager Portal per the above description may have an "Account" button which can be used to create admin users and their privileges. The fxManager Portal could also include a geographical map illustrating locations of fxDevices and their status. The fxManager **306** functions could be integrated into other network management tools. The fxManager **306** could potentially build the inventory of the network devices (fxDevices) showing all the available devices and statistics. As described above, the network administrator could optionally procure the applications a third party hosted application store (fxStore). The fxStore **308** could potentially categorize the applications several groups (as shown in FIG. **7** which is an example view of the fxStore portal). The fxStore **308** could also show the reviews of the advertised applications. In order to maintain quality, the third party fxStore **308** may perform rigorous testing of the applications and accompany those results with the advertised application.

fxSDK

The flexible Software Development Kit (fxSDK) may include a development environment and other tools to facilitate development, testing, debugging and verifications of applications for dSDN environments. The fxSDK is unique in several aspects exemplified as follows. First, since the dSDN applications may have a device component (fxDevice **302**) and cloud component (fxCloud **304**), the fxSDK allows the developer to develop the components together to simplify the development and testing. Second, the dRMS offers various system level APIs with remote accessibility. In other words, the APIs in the fxDevice **302** could be accessed via the applications in the fxCloud (fxCloudApps) and vice versa. The fxSDK simplifies the usage system created APIs and application created APIs. Third, it is essential for the dSDN to ensure quality and security of applications in the network. The fxSDK may be accompanied with an extensive Risk Analysis Utility (RAU) which verifies all the APIs that are being used by the application and based on its potential danger to the network, it would show the risk analysis and may make specific suggestions to reduce the risk. Once completed, the risk analysis result is included in the dAP for posting to the fxStore and consumed by the fxManager user (the admin). Fourth, the fxSDK would also ensure that the applications do not exceed their planned memory and resource usage. Fifth, the Integrated Development Environment (IDE) may provide various tools to actively demonstrate errors, defined APIs, security dangers, and excessive resource usage. Sixth, the IDE could also provide a flexible

Simulator (fxSimulator) for the developer test the application in a network setup in presence of other network elements.

Deployment Examples

The dSDN **300** offers flexible deployment options. In this section, a few exemplary embodiments are presented.

FIG. **8** demonstrates a shared cloud deployment where several NOITs (**802**, **804**, **806**) share an fxCloud **808**. In order to ensure complete isolations, each NOIT may be assigned a separate Virtual Machine (VM) **802***a*, **804***a*, **806***a*. A good example of this deployment option may be when multiple service providers outsource their fxCloud deployment to a third party vendor that offers a SaaS/PaaS solution. It is important to note that in this option, each NOIT should have its own secure instance of the whole system. This option could potentially create a full outsourcing model for network operations including possibilities of seamless merging of networks of two or more NOITs.

Another deployment option may be a Zoned Deployment as shown in FIG. **9**. In this option, a large NOIT may partition their network into a plurality of zones (Zone #1, Zone #2, . . . Zone #n) for administrative, software compatibility grouping, or other reason. For example, it is possible that in a large network deployment, there would be different versions of fxOS with varying capabilities in the network (the older hardware may not support newer fxOS version). In this case, the service provider may decide to group the fxDevices per fxOS versions for management simplifications.

FIG. **10** is another embodiment deployed in a Programmable & Virtualized Cellular Network. In this example, the dSDN is applied to the cellular networks (e.g., LTE). As shown, the fxDevice **1002** here is the LTE eNB (4G base-station) and the fxCloud **1004** hosts the LTE core network (EPC) functions such as MME, S-GW, and P-GW in separate VMs. Within each EPC function, new features could be added using the dSDN framework. For example, a new application to optimize signaling for Machine-2-Machine (M2M) devices could be loaded into the fxDevice **1002** and the MME function in the fxCloud **1004**.

PRINCIPAL PROCEDURES & SERVICES

The dSDN framework could offer extensive services and capabilities that would simplify the programmability of the network. In this section below, a set of main procedures and services are presented as further exemplary embodiments.

fxDevice Commissioning Provisioning

FIG. **11** illustrates a signaling flow chart for a fxDevice Plug-and-Play Commissioning Procedure. This procedure enables fxDevice commissioning and initial setup to provide a seamless and Plug-and-Play (PnP) deployment. The fxDevice relies on the fxDeviceApp-0 and the fxCloudApp-0 to connect to the backend management system and perform the commissioning procedure. In step **1101**, the fxDevice **1120** tries to discover the fxManager **1130** using the fxManager Fully Qualified Domain Name (FQDN). The fxDevice **1120** is configured to always look for the same FQDN. If the NOIT has a private cloud deployment, the DNS server in the NOIT will be configured to point to its own fxManager IP address. If the cloud is shared, the NOIT will configure their DNS server to point to the shared cloud IP address. In step **1102**, the fxDevice **1120** receives the IP address of the fxManager **1130**. In step **1103**, the mutual authentication may be performed where the fxDevice **1120** authenticates the fxManager **1130** and vice versa. In step **1104**, a secure tunnel may be created between the fxDevice **1120** and the fxManager **1130**. In step **1105**, the fxDevice **1120** may optionally request for the latest firmware/OS version. In this phase, the fxDevice **1120** informs the fxManager **1130** of its hardware information (e.g. vendor, model number, CPU model/speed, memory/cache size) and software information (e.g., current firmware version, current OS version, fxDeviceApp-0 version package stating versions of fxDeviceApp-0 components). In step **1106**, the fxManager **1120** may send the latest firmware/OS version (or point the fxDevice to the right URL to download the latest load). In step **1107**, if a new firmware/OS load is downloaded and verified, the fxDevice **1120** installs it. In step **1108**, the fxDevice **1120** requests the default settings and the default application package (the initial applications that the NOIT admin would like to install in fxDevice **1120** prior to other applications). In step **1109**, the fxManager **1130** prepares the fxCloud **1140** for the new fxDevice **1120** integration into the network. In step **1110**, once completed, the fxCloud **1140** informs the fxManager **1130** of its readiness. In step **1111**, the fxManager **1120** sends all the default settings and the default application package to the fxDevice **1120**. In step **1112**, the fxDevice **1120** then sets up a secure connection to the fxCloud **1140** (more precisely the fxCloudApp-0). In step **1113**, a commissioning completion message is set from the fxCloud **1140** to the fxManager **1130**. In step **1114**, the fxManager **1130** signal to the fxDevice **1120** the completion of the commissioning and starts th regular operation of the fxDevice **1120**. At this phase, fxManager **1130** can optionally modify the settings in the security connections between fxDeviceApp-0 and fx-CloudApp-0 and between fxDeviceApp-0 and fxManager **1130**.

Application Provisioning Procedure

The Application Provisioning Procedure enables provisioning and de-provisioning of applications across the dSDN system. For simplicity, the term provisioning in this disclosure is used to present all similar procedures of provisioning and de-provisioning. The application provisioning procedures are usually triggered by the network administrator and are orchestrated by the fxManager, which works in collaboration with the fxDeviceApp-0 and fxCloudApp-0.

FIG. **12** shows an example procedure for Application (dApp) Provisioning. In step **1201**, the network admin identifies the applications that would need to be installed in the dSDN system. These applications are already downloaded (from the fxStore or other sources) into the Application Repository (AR). Prior to installation in the live network, the authorized network admin **1220** has probably performed rigorous testing of the application in the test network. In step **1202**, the fxManager **1230** performs a detailed analysis on the potential risks that this application may create on various parts of the network and other applications. The analysis tool uses the secure manifest file and other information to discover, for example, all the APIs used by the application, the required access-level authorization level for each of those APIs, requirements on the fxVF (inter-app, intra-app communication needs), and usage of platform resources including conflict discovery and resolutions. In step **1203**, the admin defines the deployment scope that the application would be applied. Examples of deployment scopes could be a given zip code, a city, an administrative domain (zone), a shopping mall, an office

building, or manual handpicking of sites or fxDevices **1240**. The fxManager **1230** verifies compatibility of the OS version and resouces requirements of the application and the fxDevices **1240**. If the fxManager **1230** discovers incompatibility, it informs the admin and request adjustment to the deployment scope or proposes upgrading incompatible fxDevice if possible. In step **1204**, the admin **1220** initiates the installation process. In step **1205**, the fxManager **1230** unpacks the dApp and if there is fxDeviceApp component, it would prepare it for submission to the fxDeviceApp-0 of fxDevices in the deployment scope. In step **1206**, installation commands are sent to the fxDeviceApp-Os in a secure connection already set up as part of the fxDevice Commissioning. In step **1207**, the fxDevice **1240** verifies the fxDeviceApp package and performs necessary integrity checks of the received software package. Once verified, the fxDeviceApp is installed and re-verified. In step **1208**, the completion of successful install is sent to the fxManager **1230** by the fxDevice **1240**. In step **1209**, the fxManager **1230** prepares the fxCloudApp component of the dApp (if any). In step **1210**, installation commands are sent to the fxCloudApp-Os in a existing secure connection. In step **1211**, the fxCloud **1250** verifies the fxCloudApp package and performs necessary integrity checks of the received software package. Once verified, the fxCloudApp is installed and re-verified. In step **1212**, the completion of successful install is sent to the fxManager **1230** by the fxCloud **1250**. In step **1213**, the fxManager **1230** informs the admin **1220** of the successful installation process.

Hot Upgrade Procedure

The Hot Upgrade refers to a procedure by which the software upgrades on the system have no or minimal implication on the functionalities offered by the software component subject to the upgrade process. Herein are presented two types of software upgrades. For simplicity, the use of the term upgraded in this disclosure represents both upgrade and downgrade since they both use identical processes. First, upgrade of applications on the fxDevice (i.e., fxDeviceApp) and fxCloud (i.e., fxCloudApp). Second, upgrade of core functions in the fxDevice (aka fxDeviceApp-0) or fxCloud (aka fxCloudApp-0).

The upgrade procedures for fxDevice and fxCloud are similar. Here, the focus is on the fxDevice upgrade process since it is technically more challenging due to stricter resource limitations in the fxDevice.

General Application Upgrade

The applications could be upgraded automatically or manually by the network admin. In either case, the installation of the upgrade follows a similar procedure as the Application Provisioning Procedure. FIG. **13** presents the state of the machine for the application upgrade process. In state **1301**, the old version of the application is still running. In Load state **1302**: the new version of the application is downloaded in the local storage while the old version is still running. In Install state **1303**: the new version is installed (in the memory) while the old version is still running. In Run state **1304**: the new version is activated taking charge of all the related data and application state and the old version is deactivated while still remaining in the local storage. In Commit state **1305**: the old version is fully removed after ensuring the new version is running properly as expected.

fxDeviceApp-0 Upgrade

The fxDeviceApp-0 is a collection of software components that perform the core functionalities of the fxDevice. As a result, this hot upgrade is quite challenging. The most important thing is to ensure that major functionalities of the fxDevice (such as packet forwarding) remain intact during the upgrade process. The fxDeviceApp-0 upgrades may be categorized as:

No OS Kernel upgrade; or

OS Kernel upgrade.

No Kernel Upgrade

As described earlier, the packet forwarding function could be broken into the Virtual Switch and the Switch Controller. In order to maintain, the packet forwarding function, the Switch Controller function could be performed by another device or a second processor in the same device while the fxDevice is being upgraded. FIGS. **14**A-**14**C demonstrate a few examples of distributed Switch Controller function during the upgrade. In some cases, it may be possible for the fxCloud to take over the Switch Controller function (FIG. **14**A), or a neighboring fxDevice (FIG. **14**B), or second CPU in the same fxDevice (FIG. **14**C). The actual software upgrade procedure would be similar to Application Provisioning Procedure and the state machines are the same as the one shown in FIG. **13**.

Kernel Upgrade

In this case, the packet forwarding process is mostly unavailable. Therefore, one potential solution might be for the fxDevice to redirect the entire bit stream to a neighboring fxDevice. In other words, the fxDevice would have just a bare minimum routing function working and the rest of the function would be performed by a neighboring fxDevice.

All the other upgrade procedures and state machines are similar to the ones performed for the No Kernel Upgrade process.

Virtual Fabric Service

The Virtual Fabric (fxVF) provides an abstraction layer for application to communicate with each other whether they are in the fxDevice or fxCloud. Various frameworks and services may use the fxVF service. FIG. **15** illustrates fxVF **1502***a*, **1503***a*, **1504***a* in the fxDevices **1502** and **1503** and fxCloud **1504**. The fxVF **1502***a*, **1503***a*, **1504***a* may use Extensible Messaging and Presence Protocol (XMPP) for this messaging as the default protocol. It is important that the developers may use their own communication protocols between the fxDeviceApps **1502***b*, **1503***b* and the associated fxCloudApp **1504***b*. fxVF **1502***a*, **1503***a*, **1504***a* provide a secure routing mechanism.

Intra-App Messaging

This is an example of where one application in the fxDevice communicates with the same instance of the app in another instance of the same application in another fxDevice. The actual messages could go directly between the fxDevice or via the fxCloud. As an example, in the case of mobile networks, this messaging could be used to transfer user specific context from one eNB to another as a user hands off to a new eNB. In LTE, Private Messages on X2 interface could be used for messaging between the eNBs that act as the fxDevices.

Inter-App Messaging

In some cases, different applications may need to communicate with each other via their published APIs. In this

case, the security policies set up by the network administrator determines which applications could communicate with each other for what purpose. The fxVF follows the security policies determined by the network administrator for inter-application communications.

Distributed Resources Service (dRS)

The resources could be platform resources or APIs offered by the applications. The fxCloud and fxDevice resources could include (as examples):

Firewall;

Data and statistics;

Storage (usually abundant at fxCloud);

Compute (usually abundant at fxCloud);

Load evaluator;

General settings;

Routing engine (usually relevant to the fxDevice);

Virtual Machine master controller (usually relevant to the fxCloud);

Power controller (usually relevant to the fxDevice); and

Wireless engine (usually relevant to the fxDevice).

An app (fxDeviceApp or fxCloudApp) could expose APIs to be used by the other apps. The inter-app communication is enabled by the fxVF where the policy and security provisions are enforced. The APIs exposed may be RESTful (representational state transfer) and could travel across physical network elements. Since the fxManager defines fxVF policies, the administrator could ultimately specify which APIs between which apps could communicate with each other. FIG. **16** demonstrates the logical interfaces between the platform resources and applications. As shown in FIG. **16**, the distributed Resources Service (dRS) **1602**, **1603** comprises software agents residing in the fxDevice **1604** and the fxCloud **1606**. These dRS agents manage the access to the platform resolutions (including potential configuration conflict resolution) as well as facilitating inter-app APIs. The dRS **1602**, **1603** allows or disallows access to platform resource or inter-app communication according to the policies defined by the network administrator using the fxManager. In summary, the dRS **1602**, **1603** provides services to application developers for:

Exposing APIs to other applications;

Configuring and managing platform resource;

Policy enforcement and authorization of applications access to platform resource and other app's APIs; and

Policy conflict resolution.

Distributed Content Service (dCS)

The distributed Content Service (dCS) allows the developers to seamlessly store and share the contents generated by one application with other applications and its associated application in the cloud. The dCS simplifies access to the data and brings in storage virtualization to the applications. In other words, the developers no longer would need to know where the data is actually stored (in the cloud or on the device) and would be able to access them easily. The dCS implementation may use Virtual Fabric (fxVF) and distributed Resources Service (dRS).

Distributed Notification Service (dNS)

The distributed Notification Service (dNS) is another potential tool for developers that could wake or ping an application when a particular event has occurred. For example, a load monitoring application could be notified when the CPU load on a particular fxDevice (or a target area) exceeds a certain threshold. In turn, such an exemplary application could make smart decisions on reducing the load on the CPU by forcing handoffs of users to neighboring cells (i.e. fxDevices). The dCS implementation may use Virtual Fabric (fxVF), distributed Resources Service (dRS) and distributed Content Service (dCS).

Cloud Breathing Procedures

In general, resources (compute, storage) at the cloud are more abundant. However, software licensing costs and other limitation may require smart management of resources. The Cloud Breathing, here, is defined as a mechanism where the cloud resources are automatically expanded as load increases on the system or reduced as the load decreases. This creates an automatic elasticity in the cloud dimensions. FIGS. **17**A and **17**B demonstrate procedures for the cloud breathing. FIG. **17**A is expansion and FIG. **17**B is reduction. The Load Controller in the dRS **1602**, **1603** could be used to monitor the load (step **1702**) on the existing VMs (e.g. CPU or memory utilization loads). The VM Master of the fxCloudApp-0 **1608** could effectively act on the decisions made by the dRS's Load Controller. Again, the network administrator via the fxManager defines the policies and thresholds for such decisions.

Security Aspects

The security aspects of this solution are of outmost importance in order to ensure quality in the product networks.

Identity

To create a security model, the dSDN network elements should have unique identities. The following identities may be defined:

fxDevice Identity (DID): this identity is unique globally and may be allocated at manufacturing and included in the X.509 digital certificate of the fxDevice;

fxCloud Identity (CID): this identity is unique to the NOIT domain and may be allocated by the network administrator and included in the X.509 digital certificate of the fxCloud;

fxDeviceApp Identity (DAID): this identity may be globally unique and may be allocated by the fxSDK at the point of code creation by a globally accessible server;

fxCloudApp Identity (CAID): this identity may be globally unique and may be allocated by the fxSDK at the point of code creation by a globally accessible server;

dApp Identity (AID): This identity may be globally unique and may be constructed by concatenation of DAID and CAID together;

Application Developer Identity (ADID): this is a globally unique identity allocated by a globally accessible server and may be included in the developer's X.509 digital certificate and is used to sign the final developed applications.

For addressability purposes, the identities may be presented in a URI format using the FQDN of the NOIT.

Access Level

The applications are given different access levels. The AL is used by the dRS to determine which APIs or class of APIs an application can access (e.g., a fxDeviceApp running on fxDevice should not be able normally to reboot the fxDe-

vice). The required AL is generated by the fxSDK and included in the manifest files. It may also be published in the fxStore for that give application. A preset of ALs could be defined to categorize the applications security risks.

Software Security Verification

It is important for the target network element to verify the authenticity and integrity loaded software prior to install. The uniquely defined security keys of the application developers sign the applications. The following rules may be followed to ensure software security verification:

Only the verified OS/Firmware software can install on verified hardware;

Only the verified OS/Firmware software can run on verified hardware (continuous verification of essential parts of the software);

Only the verified fxDeviceApp software can install on verified hardware;

Only the verified fxDeviceApp software can run on verified hardware (continuous verification of essential parts of the software).

Each fxDeviceApp is accompanied by a manifest that is generated by the Integrated Device Electronics (IDE) at the time of compilation of the fxDeviceApp. The manifest file specifies which libraries and frameworks the fxDeviceApp uses.

The fxManager uses this information to determine the security risk of the fxDeviceApp as well as the required Access Level (AL) to run this fxDeviceApp. In order to ensure that manifest file is actually genuine manifestation of the fxDeviceApp and its integrity remains intact, a hashing algorithm could be used to generate a signature. An example is shown below:

Signature=HMAC-SHA256(K, Message);

where

Message=fxDeviceApp+Manifest

K=shared secret key between fxStore and fxManager.

The signature is generated by the IDE/fxSDK at the compile/build phase and is packaged with application binary and the manifest file. Once the distributed application package (dAP) is downloaded into the fxManager (Application Repository), the components of the signature, application binary, and manifest file are unpackaged. The fxManager then uses the same hashing algorithm to calculate the signature as above using the pre-shared key (K). If the calculated signature and the unpackaged signature match, it would prove the authenticity of the manifest file and the integrity of both manifest file and the fxDeviceApp.

Code Generation Process (Compilation):

fxDeviceApp compiled→Manifest file generation→Signature calculation→Create fxDeviceApp Package

Install Process (Consumption):

Un-package fxDeviceApp→Signature calculation→Signature comparison

This model works if the key K can be shared with two trusted parties (fxManager and fxStore). The manifest file that is generated by the fxSDK shall be secured by integrity checks to ensure it remains intact throughout the transaction (i.e. from compilation to consumption).

Communication Security

Security mechanisms built into the fxDeviceApp-0 and fxCloudApp-0 forces the fxDevice to only use the configured fxCloud servers for Software Security Verification, fxDeviceApp downloads, and fxDeviceApp-fxCloud communication. Optionally, the fxDeviceApp-fxCloud communication can be customized per application requirements. One implementation of this interface could use SPDY Protocol or per-application VPN.

For the purpose of inter-App communication, an Inter-App Communication (IAC) policy template is applied to an fxDeviceApp that defied the communication policies between the fxDeviceApps and fxCloudApps. The network administrator may use one of the pre-configured IAC policy templates (exemplified below):

Isolated: this type of applications cannot communicate with any other application in the same fxDevice and does not have any associated fxCloudApp;

Isolated-Extended: this type of fxDeviceApps cannot communicate with any other fxDeviceApps in the same fxDevice but could communicate with its associated fxCloudApp;

Private: a private fxDeviceApp in one fxDevice could communicate with same private fxDeviceApps in other fxDevice but does not have any associated fxCloudApp;

Private-Extended: a Private fxDeviceApps in one fxDevice could communicate with same private fxDeviceApp in other fxDevice and has its associated fxCloudApp;

Community: a group of fxDeviceApps that belong to the same community can communicate with each other;

Promiscuous: a Promiscuous app could communicate with any other Promiscuous application and Community application;

Custom: the network administrator could create custom IAC Policy templates and could include (as examples):

Template name;

Inherited pre-configured template if any;

The protocols allowed to use for communication;

The maximum data rate allowed for inter-app communication between a pair of fxDeviceApps or between an fxDeviceApps and an fxCloudApp;

Policies related to the other fxDeviceApps in the same fxDevice, in other fxDevice, associated fxCloudApp and unassociated fxCloudApps;

Policies related to direct fxDeviceApp to fxDeviceApp communication or relayed via fxCloud communication.

Secure Boot

The secure boot refers to a capability where the main boot code is in the fxCloud with the following exemplary procedure:

The Boot Loader (BL) in the fxDeviceApp-0 of the fxDevice looks for a particular DNS (DNSSEC) name of the fxCloud. The network administrator should define the DNS entry in their network. If there is a private fxCloud, the resolved domain name points to the fxCloud. If using shared cloud, the resolved domain name points to the cloud point of presence;

A secure tunnel (TLS) is created between the BL in fxDevice and the fxCloud using pre-burned X.509 certificate;

BL identify itself to the fxCloud and include System Information;

fxCloud determines the appropriate boot file and points the BL to the exact address of the boot file in the fxCloud;

BL downloads the correct boot file;

BL runs Software Security Verification;

BL reboots the fxDevice and the system will be ready for operation.

Frameworks

The following describes an example of the API Framework for the dSDN. Some APIs are purely local to the fxDevice or the fxCloud and some may be extended from the fxDevice all the way to the fxCloud using dRS.

General Framework

This framework refers to general libraries and APIs inherited from the legacy OS. An example of the legacy OS may be Embedded Android®. This framework allows for general processing and may be used for algorithmic applications.

Wireless Framework

The Wireless Framework adds a set up functions related to managing and controlling the wireless module. Table 1 highlights some examples of methods/functions that may be available in the Wireless Framework.

TABLE 1

Wireless Framework

Method	Description
discoverNeighbors	Discovers wireless neighbors and returns their identities
retrieveNeighborInfo	Retrieves detail information of neighboring BS/AP information. This method may transparently call cloud servers to gather more information
measureInterference	Measures wireless interference observed on a frequency or a set of frequency channels
setWirelessConfig	Sets various parameters in the wireless module such as power, frequency channels, Multiple Input/Multiple Output (MIMO) antenna configuration, etc.
definePHYReceiverAlgorithm	Defines the physical layer receiver algorithm
connectedDevice	Returns details about the connected mobile devices (length of connection, data tx/rx) to a given wireless interface (channel, band, sector)
forceOffload	Offloads connected mobile devices to other access technologies (e.g. from cellular to WiFi)
homeAreaNetworking	Control of home automation wireless technologies such as ZigBee or Z-Wave
setNeighborCells	Sets neighboring cells information
forceHandoff	Handoffs connected mobile devices to the same access technology but different channel or frequency
topMovers	Identifies the mobile devices with highest degree of mobility
positionDevice	Accurate positioning of mobile devices which may include geo-fencing data
borrowSpectrum	Allows one BS to borrow spectrum from another
configureMACScheduler	Configures the MAC scheduler (for example prioritize a particular user)
loadBalance	Enables load balancing across radio channels
defineMACScheduler	Defines the MAC scheduler algorithm

Security Framework

The Security Framework allows the developers to use a specialized security functions/methods. Some of these new methods may use the legacy OS frameworks. Table 2 highlights some examples of methods/functions that may be available in the Security Framework.

TABLE 2

Security Framework

Method	Description
setDOS Detect	Detects DOS attacks on user plane as requested
configureFirewall	Provides firewall functions such as traffic filtering, permissions (this may use dRS)
defineDOSDetectAlgorithm	Allows replacing and redefining the DOS algorithm
encryptTraffic	Encrypts the content of a message or a class of traffic as requested
authenticateUser	Authenticate users as requested
integrityCheckContent	Verifies authenticity of a message using its checksum information
configures IPS	Configures IPS functions on the user plane

FastPath Framework

The FastPath Framework allows the developers to identify and manipulate the data path with a DPI capability. Table 3 highlights some examples of methods/functions that may be available in the FastPath framework.

TABLE 3

FastPath Framework

Method	Description
matchTraffic	Identifies and matches the traffic type requested by the developer
detectCongestion	Detects if there is congestion on the data path
identifyTopURLs	Identifies the most visited URLs by the users
detectLocalConversation	Detect conversations and packet routing between the end user on the same fxDevice
detectDevice	Detects traffic of a given mobile user device
redirectTraffic	Redirect data traffic to a given network element. The traffic may be identified with a user or a class of traffic
connected Devices	Returns all the connected mobile devices to a given fxDevice
assionAddress	Allocates a particular IP address to the mobile device
setPriority	Set traffic priority to a class of traffic
markTraffic	Marks the user traffic with the desired QoS tags
setupTunnel	Sets up data tunnel from fxDevice to a destination
Adscript	Adds a JavaScript to an active transiting web page
detectIdenticalFlows	Identifies the identical flows coming from the WAN/uplink interface on the fxDevice
aggregateFlows	Enables aggregation of flows to save WAN/uplink utilization
routeTunnel	Routes a packet or a class of traffic onto an established tunnel
identifyTopDevices	Identifies users that generate most traffic load

Messaging Framework

The Messaging Framework allows the developers to send messaging between the applications residing in the fxDevice

and fxCloud. Table 4 highlights some examples of methods/functions that may be available in the Messaging Framework.

TABLE 4

Messaging Framework

Method	Description
sendIntraApp	Sends an Intra-App message from one fxDevice tn another
sendInterApp	Sends an InterApp message from one application to another
sendfxCloud	Sends a message to the fxCloud
sendSMS	Sends an SMS to the user of choice

Caching Framework

The Caching Framework allows the developers to cache particular contents or files that are accessed frequently in the caching engine. The caching can be done locally in the fxDevice, clustered cache (shared amongst a few fxDevice), or cloud caching. Table 5 highlights some examples of methods/functions that may be available in the Caching Framework.

TABLE 5

Caching Framework

Method	Description
storeForward	Enables Store and forward model for a target class of traffic where the target traffic is matched and stored upon a condition and forwarded the condition is relieved
cacheWebSite	Caches the most visited web sites
cacheVideo	Caches the most viewed videos
cacheCloud	Caches the requested content or traffic type in the cloud storage in fxCloud

Management Framework

The Management Framework enables the developers to manage the BS platforms and perform administrative procedures. Table 6 highlights some examples of methods/functions that may be available in the Management Framework.

TABLE 6

Management Framework

Method	Description
upgradeFirmware	Started and executes the firmware upgrade procedure on the fxDevice
upgradeOS	Started and executes the operating system upgrade procedure on the fxDevice
remoteBootup	Performs remote boot up process where the fxDevice works closely with the fxCloud to perform the bootup
remotePower	Allows fxCloudApps to remotely turn on and off a fxDevice or elements in the fxDevice (e.g. the wireless)
gatherStatistics	Instructs fxDevice to gather specific statistics. Various commands may be consolidated for gathering statistics. Number of active users Stats of time spent by users in the cell Stats of quality perceived by the users Neighboring cell info Overall traffic passed through the system Total number of connected devices per frequency, per carrier, per site, per location Handover failure statistics Cloud aggregated data processing and full report Total traffic passed Applications used (meta-data correlated) Bandwidth consumed User's mobility pattern
setMonitoring	Allows the developer to get reports and statistics of the fxDevice traffics, errors, and neighboring environment. The reporting criteria could be set up as: 1) Period, 2) Event based (reports only if a threshold is passed; a threshold could be based on absolute numbers or deltas), 3) One-time (i.e. the information is pulled once). The exact parameters to get report on may use the same data structures used by the method "getStatistics".
getSystemInfo	Retrieves static and dynamic system info such as: Hardware info: vendor, model number, capability profile. GPS, Cellular, WAN interfaces, RFID/NFC Heat temperature Vendor Specific Information Software info: OS version, firmware information CPU load, per-application load (CPU, memory, accelerators) Loaded apps and status Memory utilization Storage utilization Interface status & utilization (RF, backhaul, management...)
getfxDeviceLocation	Retrieves location of the fxDevice Location would be presented in GPS coordinates and civil address Location enhancement: Cloud, map data, and neighbor discovery could help more accurate positioning
getPowerConsumption Rate	Retrieves current power consumption rate of the fxDevice
configureWAN	Confioures the WAN settinos
retrieveNeighborInfo	Retrieves neighbor information from the fxCloud
getRFIDInfo	Returns the RFID/NFC information of the fxDevice
setRFIDInfo	Sets the RFID/NFC information of the fxDevice

Extensible API Framework

This framework allows the developer to create custom APIs and make it accessible for other applications (in the fxDevice or in the fxCloud). In turn, the fxCloud could present these APIs using e.g. REST technologies (via the fxCloud Northbound Interface) to the developer outside the dSND system. There might be limits put on the APIs exposed through the cloud to avoid potential misuse or security threats. The dRS enforces the security policies defined by the admin (via fxManager). The applications take a role of Client Application (CA) or Server Application Role (SA). The CA and SA could be distributed in the fxDevice or fxCloud. The CA makes requests and SA serves the requests.

The Extensible APIs could be categorized (as examples) into:

Read-Only Data: this type of APIs allows a CA to read data from SAs without the ability to change or request any particular action from the SAs

Read-Write Data: this type of APIs allows CA application to read data from SAs with the ability to change some of their data but no option for requesting any particular action from SAs;

Procedural: this type of APIs allows CAs to execute a particular procedure in SAs. The rights to change any data still depends on whether the API allows Read-Only Data or Read-Write Data types.

Cloud Management Framework

The Cloud Management Framework presents a collection of methods enables management of the cloud services and resource. Table 7 highlights some examples of methods/functions that may be available in the Cloud Management Framework.

TABLE 7

Cloud Management Framework

Method	Description
detectCloudCongestion	Detects congestions on a particular VM
deleteVM	Deletes an active VM permanently
suspendVM	Suspends an active VM but doesn't remove it
activateVM	Activates an already created VM
createVM	Creates a new VM but doesn't activate it

EXAMPLES OF DAPPS

The dSDN create an end-to-end programming platform and the possibilities of vApps are only limited by the developers' imagination. An example list of potential vApps are presented in Table 8 below.

TABLE 8

highlights examples of use cases possible by the Distributed Software Defined Network (dSDN)

Use	Description	FrameworkiMethods} used
1	Distributed SON: For example, the WiFi AP could search for the least crowded channel to minimize the interference	Wireless {discoverNeighbor, retrieveNeighborInfo, measure nterference, setWirelessConfig, load Balance}, General Framework, Messaging {sendIntraApp}
2	BS Hot Firmware Upgrade without impact to the basic functions of the BS. For this function to support full hot upgrade without operational impact, the Communication SoC should be able to	Management {upgradeFirmware}
3	BS Hot infOS Upgrade without impact to the basic functions of the BS. For this function to support full hot upgrade without operational impact, the full platform should have dual processor.	Management {upgradeOS}
4	Secure network-based booting. This allows to keep the code securely in the cloud to avoid tampering by the ODM/CM or Man-in-the-Middle attacks	Management {remoteBoot}
5	Turning on or off APs if nobody is in the office. Sensors in the building send information to fxCloudApp which in turn orders power control of the APs	Management {remotePower}
6	Store and forward model for low priority M2M applications where the target traffic is matched and stored in the case of network congestion and forwarded once the network congestion is relieved.	DataPath {matchTraffic, detectCongestion}, Cache {storeForward}
7	Replacing MAC Scheduler	Wireless {defineMacScheduler}
8	Replacing Receiver Algorithms	Wireless {definePHYReceiverAlgorithm}
9	Caching most visited website or video clip in the enterprise	DataPath {identifyTopURL}, Cache {cacheWebSite}
10	The most viewed videos and webcasts are cached in the wireless edge	DataPath {identifyTopVideos}, Cache {cacheVideo}
11	Local Routing: data and voice sessions between two parties on the same cell are routed within the same cell bypassing the core network	DataPath{detectLocalConversation, RouteLocal}
12	Local Routing: data and voice sessions between two parties on the nearby cells are routed locally bypassing the core network	DataPath{detectConversationDetect, routeLocal}
13	Temporary redirection of a data flow for monitoring and security reasons. In this use case, a particular user's traffic is tracked by redirection to a monitoring station. If user is mobile, such profile of tracking and associated redirection is transferred from fxDevice to fxDevice either directlv or via the fxCloud	DataPath {detectDevice, matchTraffic, redirectTraffic}
14	Power Calendar: On-demand coverage (triggered by passing traffic thresholds in certain cells or by calendaring): Operators need to design their network for peak usage rates at high costs, leaving their network underutilized most of the time	DataPath {detectCongestion}, Management {remotePower}
15	DOS attack recognition and action could be blocking the user, the app, and/or notifying the user via an SMS message	Security {setDOSDetect, configureFirewall}, Messaging {sendfxCloud}
16	Applying certain ACLs to the BS. For example, the students in the class can only access Facebook between the breaks and access to Facebook is blocked during the class	Security {configureFirewall}
17	Forced offloading of some cellular users (users that have attached to the same cell for a while) to WiFi by changing "something" in the cellular connection to force WiFi connection (Heterogeneous)	Wireless {identifyConnectedDevice, forceOffload}
18	Service continuity and anchoring at eNB for WiFi using the same LTE IPaddress (Heterogeneous)	DataPath {identifyConnected Devices, assignAddress}
19	Graphic/general processing for special use cases such as City BS	General Frameworks
20	Application filtering/throttling (secondary conditions: time-based, location-based, subscriber-type, none)	DataPath {matchTraffic}, Security {configureFirewall}

TABLE 8-continued

highlights examples of use cases possible by the Distributed Software Defined Network (dSDN)

Use	Description	FrameworkiMethods} used
21	Local Services: The enterprise user can discover Bonjour or UPnP devices and services (e.g. printer) even when connected to LTE/3G small-cell in the enterprise. In this case, the small cell or fxDevice pro-actively listens to Bonjour and UPnP discovery and advertisement messages (SSDP) and cache the available services until a device reguest for such information. In the case of Bonjour, fxDevice could listen to Multicast DNS (mDNS) messages sent over cellular connection (LTE/3G) and multicast them on the LAN or WLAN interfaces.	DataPath {matchTraffic, routeLocal}
22	Local Services: The premise-owner (e.g. airport, hotel) can provide ads under the browser for sponsored WiFi or cellular (3G/LTE) access over multi-mode small cell. The premise owner could use this service to advertise special offers or third party ads. This technigue uses HTML <script> for example to add special content to the bottom of the pade.	DataPath {matchTraffic, addScript}
23	Distributed Content Distribution Network (CDN). In this use case, the fxDevices function as CDN that bring the contact closer to the end users.	DataPath {matchTraffic}, Cache {cacheVideo}
24	In case of national security, the service provider may limit the access to the network so only the law enforcement can use the cellular network. In this case, the admin defines a Target Area (TA) using an fxCloudApp in the fxCloud where the emergency lockdown needs to be applied. The IMSI/MSISDN number of law enforcement mobile devices are sent to the fxDevice in the TA which in turn enforce the policy. The policy may allow the public to just sent SMS while the law enforcement could have full access and priority to the system.	DataPath {matchTraffic, setPriority}, Security {configureFirewall}
25	Emails and web browsing monitored by the cloud services assisted by the agent in the dSDN network. In this case, the fxDevice detects certain traffic (email or web in this case) and redirects them to a cloud service for security cleansing.	DataPath {matchTraffic, redirectTraffic}
26	Analytics pre-analysis: The data collected from M2M devices could be processed (noise could be filtered out) and packaged (compressed) for consumption by cloud services	DataPath {matchTraffic}, General Framework, Messaging {sendfxCloud}
27	Certain traffic could be marked with QoS DSCP codes for further processing in the network	DataPath {matchTraffic, markTraffic}
28	Proprietary SP services and applications that run purely between the phone and the BS . For example, the SP may implement a certain application for the law enforcement to get direct feed of surveillance cameras in a target area (TA).	DataPath {matchTraffic}, Security {configureFirewall}, General Framework
29	IP-PBX implementation in the small-cell or WiFi AP for enterprise use. This would allow a simple out of a box solution. The VoIP phones connect to the fxDevice via the LAN connections. The WAN connection could connect to an fxCloudApp for PSTN calls	DataPath {matchTraffic}, General Framework
30	Network management system can add its own agent/probe to collect and consolidate data and transmit it to its cloud element using any protocol	Management {gatherStatistics, getSystemInfo, control}, General Framework
31	Application based performance monitoring on the wireless link (number of packets sent in idle mode, retransmit rate...). Such information could be shared with the app developer to optimize its application	Management {gatherStatistics}, General Framework
32	Seamless network sharing and mergers. This is the case where two Network Operators would like to share or merge their network. Using dSDN's infManager, the network administrators could merge the network administration control. This process may include setting up the desirable default fxApp packages of the network operator	fxManager Portal Service
33	Following a user to understand where it faces coverage issues or call drops. Once discovered, the system could send a message to the user acknowledging an issue and that the SP is trying to fix this. The information is uploaded to the cloud for further analysis	Management {gatherStatistics}, General Framework, Messaging {sendSMS}
34	Reporting: This allows the developer to get reports and statistics of the fxDevice traffics, errors, and neighboring environment. The reporting criteria could be set up as: 1) Period, 2) Event based (reports only if a threshold is passed; a threshold could be based on absolute numbers or deltas). 3) One-time (i.e. the information is	Management {setMonitoring}, General Framework
35	Proprietary security protocol between the application in the mobile device and the BS in enterprise deployment. This would allow SP to create enterprise-class LTE for indoor using small cells	Security {encryptContent, authenticateUser, Integrity, configureFirewall}, DataPath {RouteLocal}, General Framework
36	The femto cell at home that can't scan neighboring cells, gets a list of neighboring cells from the fxCloud or neighbors to broadcast to improve outbound handoffs	Wireless {discoverNeighbors, retrieveNeighborInfo, setNeighborCells}, Management {retrieveNeighborInfo}
37	IMS-capable femto enabling various SP services to initiate or transferred between home network and mobile devices	Messaging {sendfxCloud}, General Framework

TABLE 8-continued

highlights examples of use cases possible by the Distributed Software Defined Network (dSDN)

Use	Description	FrameworkiMethods} used
38	Integrated smart home with multi-mode femto/WiFi router. For example, the application in the smart home router (fxDevice) allows for coordination of home physical security service (such as ADT services). With the fxCloudApp, the user could control the security status of his/her home and remotelv control the settina of the	Wireless {homeAreaNetworking}, Messaging {fxCloud}, General Framework
39	Home CCTV DVR implementation in an integrated home router. The camera feeds are recorded on the same router caching engine	Cache {cacheVideo}, Messaging {sendfxCloud}, General Framework
40	Enterprise CCTV DVR implementation in an integrated router with cloud backup. The camera feeds are recorded in the cloud.	Cache {CacheVideo, CacheCloud}, Messaging {fxCloud}, General Framework
41	User traffic separation into various MPLS/LSP (integrating VPN and BS). In this case, users are classified into groups and each group is tunneled to the core network. For example, law enforcement use the encrypted tunnel while the other users use the default tunnel	DataPath {Match, SetupTunnel, RouteTunnel}
42	Dynamic spectrum allocation where a BS borrows unused spectrum from the neighboring BS in case of temporary congestions	DataPath {detectCongestion}, Wireless {discoverNeighbors, setWirelessConfig}, Management {retrieveNeighborInfo, borrowCarrier}, Messaging {fxCloud}
43	Proxy ANDSF (Access Network Discovery and Control Function) in the BS. When the UE tries to access the ANDSF, the Proxy ANDSF fxDeviceApp in the BS intercept the ANDSF messages and replies on behalf of the ANDSF. This would speed up the WiFi network discovery for example.	DataPath {matchTraffic}, General Framework, Messaging {sendfxCloud}
44	HD Audio calling: audio codecs in BS allows the phones with the right capabilities to communicate with HD Audio quality calling. If one of the mobile devices can't handle the HD quality audio, the BS performs the transcoding of the audio.	DataPath {matchTraffic}, General Framework
45	Enabling mobility on White Space Spectrum: The latest white space database will be downloaded. The mobile devices with special software applications make requests to the BS to retrieve the local available white space frequencies. This would allow the mobile device to handoff seamless between BS using the white space spectrum. The BS would advertise such a capability to the UEs. The secure communication allows peer-to-peer communication between the UE and the BS for certain applications such as White Space Channel lookup or site acquisition tools for field engineering and RF planning.	Messaging {sendfxCloud}, General Framework
46	Site acquisition tool where the technician uses a software tool on his mobile phone to interact with BSs to determine the best place to deploy the new BS. The phone app interests with fxDeviceApps in BSs and fxCloudApp. For example, fxDeviceApp performs air interface qualitv and aives a orioritv to traffic aenerated bv for this	Messaging {sendfxCloud}, General Framework. DataPath {matchTraffic, setPriority}
47	DNS caching at BS to accelerate DNS lookup for subsequent users. This would improve the user experience by reducing response delays	DataPath {matchTraffic}, General Framework
48	Continuous SLA monitory for certain protocols, applications or general link quality for jitter, delays, bandwidth	DataPath {matchTraffic}, Management {gatherStatistics}, General Framework
49	Enterprise users can use the small cell deployed in their enterprise free of charge. The DPI engine in fxDevice classifies the traffic and identifies the enterprise private data and they would be routed locally and charging records are generated by the BS (the SP decides how to bill the enterprise user and it mav decide to make such usage free	DataPath {matchTraffic, routeLocal}
50	On-premise small-cell aggregation where on fxDevice takes the responsibility of aggregating multiple small-cells on the premise and presenting them as a single BS/Node-B/eNodeB to the core network. In this case the aggregating fxDevice (AfxE) may take responsibility of local radio resource manager (RRM) or self-optimizing network (SON) server. AfxE can also implement and enforce policies concerning local service access on the premise (enterprise) as well as enable seamless handover between cellular technologies and on-premise (enterprise) local area network (LAN) for example using WiFi technology.	General Frameworks, Wireless {discoverNeighbors, retrieveNeighborInfo, setNeighborCells, setWirelessConfig}, Messaging {sendIntraApp}
51	Self adjusting fxCloud compute resources: The fxCloud creates and tears down virtual machines (VMs) based on the traffic load measured by itself or the packet processing and data plane engine. It is also possible that the VM adjustments are made based on the rush hour or pre-configured hours of the day or manually by the administrator.	CloudManagement {removeVM, createVM, detectCloudCongestion, actiavteVM, suspendVM}
52	Core Network Function Virtualization (cNFV) using distributed Software Defined Networking (dSDN): In the case, once an fxDevice is commissioned, its fxDeviceApp0 (i.e. firmware) uses predefined DNS names to discover the rest of the core network (e.g. MME, S-GW Rtc 1 The corp network elements could be created and	General Frameworks, Messaging {sendInterApp, sendintraApp), CloudManagement {deleteVM, suspendVM, createVM, activateVM, detectCloudCongestion}
53	Smart display and digital signage as wireless base stations: this use case allows combining the capabilities of a base station and the smart display in shopping malls, airports, and enterprises. There is a already electric power and connectivity for the smart display that the base station function could use.	General Frameworks

TABLE 8-continued

highlights examples of use cases possible by the Distributed Software Defined Network (dSDN)

Use	Description	FrameworkiMethods} used
54	Tracking a mobile user at base station level even if the UE is in the idle mode. Currently in the idle mode, the MME, SGSN, or VLR/MSC can only know the location of the UE in with a accuracy of LA/RA/URA which is a very large area. The administrator uses the fxCloudApp and requests tracking of the user/UE (the admin may provide target area where the user/UE may be located; e.g. zip code or town name). The fxCloudApp contacts its fxDeviceApp in the fxDevices in the target area (TA) requesting information of a particular IMSI/MSISD/TMSI. Once an fxDevice responds indicating knowledge of the user location, the fxCloudApp directs that serving fxDevice to inform the fxDeviceApp in the next fxDevice of the instructions to track the user and contact fxCloudApp when they user hands off to due to its mobility. fxCloud could create user friendly interfaces and presentations to show for example the direction the user is moving, predict next location of the user, and estimate arrival of the user to a certain location	DataPath {matchTraffic}, Messaging {sendIntraApp, sendfxCloud}
55	In case of emergency, fxDevices in a target area block all public traffic and only allow law enforcement sessions using specially encryption and authentication between the UE and the fxDevice or fxCloud (depending on the requirements)	DataPath {matchTraffic, setPriority}, Security {configureFirewall, encryptTraffic, authenticateUser}, General Framework
56	SON triggered by the data traffic consumption. In this case, the mobile cell configuration is adjusted based on the current traffic of the cell. It could for example force handoff some users (high users or users moving fast) to neighboring cells.	DataPath {identifyTopDevices}, Wireless {forceHandoff}
57	Time based QoS and policy. In this use case, the infManager pushes certain policies in target area (TA). Such polices could be turning on extra capacity (frequency channels) or QoS policies and traffic prioritization on the WAN like	DataPath {matchTraffic, setPriority}, Wireless {discoverNeighbors, setWirelessConfig}
58	Fast moving users detection: in this use case, the application in the BS measures the speed of the user (based on the time was handed off to this cell until it was handed over to the next cell). This information could be used by the fxDevice or fxCloud to force handoff the mobile user to a larger cell to reduce handoff rates	General Frameworks, Wireless {identifyTopMovers, forceHandoff}
59	Emergency broadcast of information and live camera feeds. In case of emergency, the enterprise network administrator temporarily block all traffic and only allow emergency related traffic across the target area in the enterprise network. For example, the network admin broadcasts the camera feeds and alerts over the WiFi access points using multicast IP techniques. The client devices and digital sinages can listen to that particular multicast and show the camera feeds and emergency alerts	DataPath {matchTraffic, setPriority}, Security {configureFirewall}, General Framework
60	Dynamic advertising in shopping malls. The shopper would use an app from the shopping mall. When the shopper brings up or refreshes the app, the dSDN allows the shopping mall to pinpoint the exact indoor/outdoor location of the user (e.g. the shop the shopper is ini and send special coupon and ads for that store to the shopper.	General Frameworks, Wireless {positionDevice}, Messaging {sendIntraApp, sendfxCloud}, Security {configureFirewall}, DataPath {matchTraffic, redirectTraffic}
61	Smart tour guide. The tourist would use an app from the tourism board or museum. When the user brings up or refreshes the app, the dSDN allows to pinpoint the exact location of the tourist (e.g. the painting he/she is looking at) and send tour guide information (e.g. audio, video, hypertext). dSDN allows to localize the positioning and	General Frameworks, Wireless {positionDevice}, Messaging {sendIntraApp, sendfxCloud}, Security {configureFirewall}, DataPath {matchTraffic, redirectTraffic}
62	General hotspot landing page	DataPath {matchTraffic, redirectTraffic}, Security {authenticateUser}
63	Proprietary wireless mesh implementation	Wireless {discoverNeighbor, retrieveNeighborInfo, measureInterference, setWirelessConfig}, General Framework, Messaging {sendIntraApp}
64	Power and energy consumption map of the network. A real-time fxCloudApp pulls the power consumption details from each fxDevice periodicaly or on demand. The pulled data then are combined with the location information and correlated into a map to present a viewable consumption man	Management {getfxDeviceLocation, getPowerConsumptionRate}
65	Network deployment optimization. This application allows network operator to get advice for best optimal reconfiguration and/or redeployment of the network. For example, the network admin could get suggestion that based on the power consumptions and coverage perceived and spectrum utilization, it might be better to replace a particular group of smaller cells with a single macro-cell. This app could also pull information from larger OEMs based on the subject deployment scenario (the OEMs either provide this information statically to the tool or provide network APIs for use by this app e.g. using REST protocol)	Management {getfxDeviceLocation, getPowerConsumptionRate, gatherStatistics}
66	High availability. This app allows two or more fxDevice to form a redundancy cluster where they participate in statueful redundancy and load-balancing	Messaging {sendIntraApp}, General Framework

TABLE 8-continued

highlights examples of use cases possible by the Distributed Software Defined Network (dSDN)

Use	Description	FrameworkiMethods} used
67	Bandwidth Bursting. This feature enables the dynamic adjustment of backhaul or WAN link bandwidth as the demands change in the enterprise or public wireless deployments. In this case, the fxDeviceApp in the fxDevice monitors the WAN link utilization, once a preset threshold percentage has reached, it messages its sister fxCloudApp which in turn would send a request to the fxDeviceApp to increase or decrease the WAN bandwidth. If additional backend reconfiguration is needed, the fxCloudApp performs those additional changes prior to informing the fxDeviceApp of the bandwidth	Management {getStatistics, configureWAN}
68	Wireless link quality measurement by the small-cell or WiFi AP: The wireless SoC in the small-cell or WiFi AP continuously monitors the link quality for power adjustment and handover reasons. Such information could be exposed to the app developers to create cell-level analytics and reports without the need to have test-drives or field testing of service quality	Management {gatherStatistics}, General Framework
69	Data Path Consolidation: An application on the fxDevice discovers identical data feeds from a cloud server to two or more users. The fxDeviceApp in the fxDevice can discover this potential duplicate flows at the session establishment (via a proxy function) and it would consolidate those sessions into one on the uplink towards the cloud server. A good example of this use case is the web-based desktop sharing where different many users in the same office site get the same feed from the cloud server.	FastPath {detectIdenticalFlows, aggregateFlows}

The following references are herein specifically incorporated by reference in their entirety into this disclosure:

[1] 3GPP TS 23.002, "Network Architecture";

[2] 3GPP TS 23.402, "Architecture enhancements for non-3GPP accesses";

3] 3GPP TS 23.401, "General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access";

[4] Open Base Station Architecture Initiative (OBSAI); and

[5] SPDY Protocol (http://datatracker.ieff.org/doc/draft-mbelshe-httpbis-spdy/).

The invention claimed is:

1. A system comprising:

a programmable network device adapted to host a plurality of network device applications;

a programmable cloud device adapted to host a plurality of cloud applications, wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications; and

wherein the plurality of network device applications and plurality of cloud applications device form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware.

2. The system of claim **1**, wherein the programmable network device and the programmable cloud device each include an access network interface unit configured to send and receive communications and a processor with a memory associated with the access network interface unit.

3. The system of claim **1** further comprising:

an application management portal capable of managing life cycles of the distributed applications; and

an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested and certified distributed applications to the application management portal.

4. The system of claim **3**, wherein the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications.

5. The system of claim **3**, further comprising:

a distributed resource service (DRS) which controls access to a plurality of resources upon receiving instructions from the application management portal.

6. The system of claim **5**, wherein the plurality of resources include at least one of a group consisting of: a firewall; data and statistics; storage; a load evaluator; general settings; a routing engine; a virtual machine master controller; a power controller; and a wireless engine.

7. The system of claim **5**, wherein the DRS further includes a load controller adapted to monitor loads on at least one of the plurality of network device applications and at least one of the plurality of cloud applications and effect change in accordance with thresholds received from the application management portal.

8. The system of claim **3** further comprising:

a distributed notification service wherein notice of a predetermined event is sent from at least one of the plurality of network device applications to at least one of the plurality of cloud applications.

9. The system of claim **1**, wherein at least one of the plurality of network device applications is responsible for a plurality of functional capabilities of the programmable network device and is capable of at least one of the group consisting of: updating firmware of the programmable network device; updating default communication protocols of the programmable network device; initializing programmable network device settings; boot loading the programmable network device; providing security digital certificates; providing operating system (OS) and kernel upgrades; and upgrading and commissioning agents for upgrading at least one of the plurality of network device applications and related procedures.

10. The system of claim 1, wherein at least one of the plurality of cloud applications is responsible for a plurality of functional capabilities of the programmable cloud device and is capable of at least one of the group consisting of: upgrading and commissioning agents for updating at least one of the plurality of cloud applications; hypervisor upgrades; virtual machine and isolated execution environment lifecycle management; security digital certificate management; operating system (OS) and kernel upgrades; and default communication protocols.

11. The system of claim 1, wherein the programmable network device is an LTE eNB and the programmable cloud device hosts LTE core network functions in separate virtual machines from the group consisting of: MME, S-GW, and P-GW.

12. The system of claim 1, further comprising:

a virtual fabric which provides a secure communication layer for said at least one of the plurality of network device applications and said at least one of the plurality of cloud applications.

13. The system of claim 1, further comprising:

a second programmable network device which includes at least one of the plurality of network device applications which is the same as at least one of the plurality of network device applications in the programmable network device and where each of the network device applications can communicate directly.

14. The system of claim 1, further comprising:

a distributed resource service which is capable of providing at least one service from a group consisting of: exposing application programming interfaces (APIs) to other applications; configuring and managing platform resources; policy enforcement and authorization of the plurality of network device applications and the plurality of cloud applications to access [to] platform resources; and policy conflict resolution.

15. The system of claim 1 further comprising:

a distributed content service wherein contents generated by the at least one of the plurality of network device applications are shared with at least one of the plurality of cloud applications.

16. The system of claim 1, wherein the programmable network device, at least one of the plurality of network device applications, the programmable cloud device, and the at least one of the plurality of cloud applications each have unique identities.

17. The system of claim 1, wherein at least one of the plurality of network device applications and at least one of the plurality of cloud applications are given an access level to determine which APIs can be accessed.

18. The system of claim 1, wherein the at least one programmable network device and the programmable cloud device are adapted to verify the authenticity and integrity of updates to the plurality of network device applications and the plurality of cloud applications.

19. A system comprising:

a programmable network device adapted to host a plurality of network device applications;

a programmable cloud device adapted to host a plurality of cloud applications, wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications;

wherein the plurality of network device applications and plurality of cloud applications form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware;

an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications; and

an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested distributed applications to the application management portal.

* * * * *

Appendix A

Claim Chart for Unpatentability of the ‘823 Patent (Stewart)

Claim Chart for U.S. Patent No. 11695823 ('823 Patent)

I. Grounds of Unpatentability

	Claims of '823 Patent	Statute(s)	Prior Art
1.	Claims 1, 2, 3, 4, 8, 15 and 19	35 U.S.C. §102	Stewart
2.	Claims 1, 2, 3, 4, 8, 15 and 19	35 U.S.C. §103	Stewart

II. Prior Art Relied Upon

Reference
Ex. 1004 U.S. Pub. No. 20120278454 to Stewart *et al.* (**Stewart**) was filed as U.S. Patent Application No. 13/450,625 on April 19, 2012, published on November 1, 2012, and issued on January 22, 2019 as U.S. Patent No. 10187494. Stewart is prior art to the '823 Patent at least pursuant to 35 U.S.C. §102(a)(1) and (2).

III. Claim Chart

Claims of '823 Patent	Prior Art Disclosure
Claim 1 [P]	
A system comprising:	To the extent that the preamble is limiting, Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation.[1] Stewart discloses *a system*.[2] Stewart (Ex. 1004) at ¶¶ [0003] and [0043]. ***See, e.g.*, Stewart** The present invention generally relates to software application development and deployment on remote devices, and more specifically to **systems**[3] and methods for configuring or managing gateway devices at various locations. Stewart at ¶[0003] **FIGS. 1A and 1B illustrate various exemplary systems that may implement embodiments described herein**. However, these systems are exemplary only and other systems and configurations are envisioned. Stewart at ¶[0043]

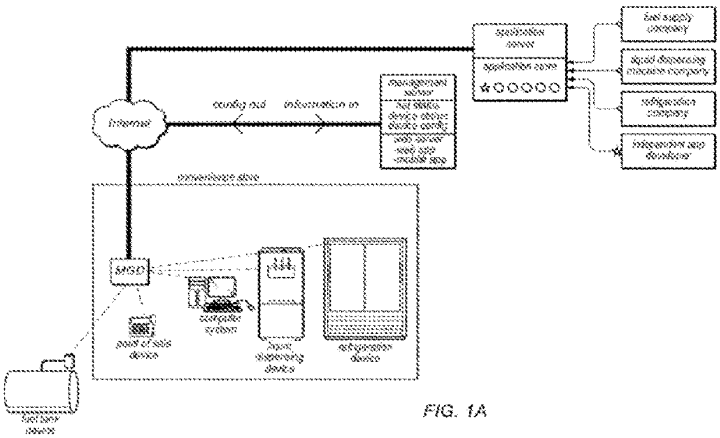
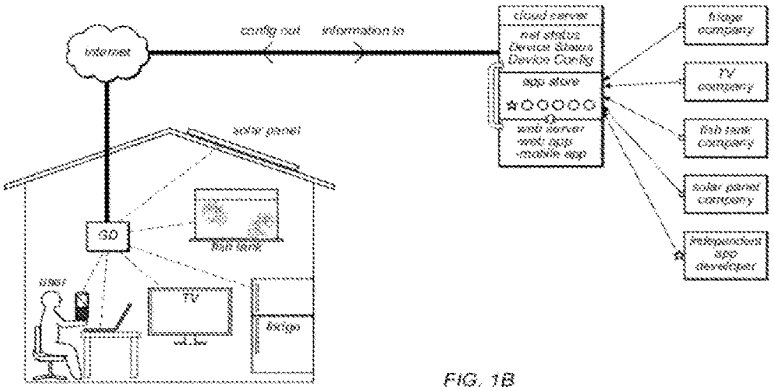
[1] The discussion in this Claim Chart is equally applicable to both the anticipation and obviousness grounds for rejection unless otherwise noted.
[2] Language in italics throughout this document denotes claim language of the claims of the '823 Patent.
[3] Unless otherwise noted, all bold used in quotes of the disclosures of the prior art is emphasis added to highlight relevant portions.

Claims of '823 Patent	**Prior Art Disclosure**
Claim 1 [a]	
a programmable network device adapted to host a plurality of network device applications;	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *a programmable* (e.g., configurable) *network device* (e.g., a gateway device) *adapted to host a plurality of network device applications* (e.g., applications or application programs on the gateway device). Specifically, Stewart discloses a system that includes a gateway device that is "configurable with one or more application programs." Stewart (Ex. 1004) at ¶ [0042]. Stewart further discloses that "the gateway device may install [and thus host] applications provided by the cloud server to manage the devices coupled to the gateway device." Stewart (Ex. 1004) at Abstract. ***See, e.g.*, Stewart** Configuring and managing gateway devices. A **gateway device** may be installed at a location, such as a merchant location or home. The gateway device may manage (e.g., monitor and/or control) one or more devices at the location. For example, the gateway device may manage various fuel tank devices, point of sale devices, refrigeration devices, liquid dispensing devices, etc., at a convenience store. The gateway device may provide reported information to and receive commands from a cloud server, which may be in communication with other devices which may be used to view reported information of or control the managed devices. **The cloud server may also be configured to install applications, e.g., for execution by the cloud server, the gateway device, and/or other devices. For example, the gateway device may install applications provided by the cloud server to manage the devices coupled to the gateway device**. Stewart at Abstract **Gateway Device**—refers to a device that is configured to couple to a wide area network, such as the Internet, and also couple (either wired or wirelessly) to one or more devices, and which **is configurable with one or more application programs** to monitor and/or control the one or more devices. The term "Merchant Gateway Device" refers to a Gateway Device that is configured to be placed at a merchant location, e.g., which is configurable with one or more application programs to monitor and/or control one or more devices that would typically be located at a merchant location. Stewart at ¶[0042]

Claims of '823 Patent	**Prior Art Disclosure**
Claim 1 [b]	
a programmable cloud device adapted to host a plurality of cloud applications	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *a programmable* (e.g., configurable) *cloud device* (e.g., cloud server) *adapted to host a plurality of cloud applications* (e.g., applications or application programs on the cloud server). Specifically, Stewart discloses a system that includes a "cloud server" that hosts "applications and may facilitate selection and deployment of these applications on the gateway device." Stewart (Ex. 1004) at ¶¶ [0010]; See also *Id*. at Abstract and ¶¶ [0012], [0067], and [0115]. Indeed, Stewart discloses that its system may include one or more servers, such as an application server and a management server, each of which may be (together or separately), the cloud server. Stewart (Ex. 1004) at ¶¶ ¶¶[0054]-[0055], [0068], [0078], and Figs. 1A and 1B. The cloud server is programmable at least in that (1) applications may be executed on the cloud server, and (2) the cloud server includes an application server and applications may be developed and uploaded to the cloud server. Stewart (Ex. 1004) at ¶¶ [0011]-[0012], [0067], and Figs. 1A and 1B. ***See, e.g.*, Stewart** Configuring and managing gateway devices. A gateway device may be installed at a location, such as a merchant location or home. The gateway device may manage (e.g., monitor and/or control) one or more devices at the location. For example, the gateway device may manage various fuel tank devices, point of sale devices, refrigeration devices, liquid dispensing devices, etc., at a convenience store. The gateway device may provide reported information to and receive commands from a **cloud server**, which may be in communication with other devices which may be used to view reported information of or control the managed devices. **The cloud server may also be configured to install applications, e.g., for execution by the cloud server, the gateway device, and/or other devices. For example, the gateway device may install applications provided by the cloud server to manage the devices coupled to the gateway device.** Stewart at Abstract The gateway device may also be coupled to a server (e.g., a cloud server) over a wide area network, such as the Internet. **It is noted that references to "server" herein may refer to a dedicated server or a cloud server**. The server may be used to configure and/or control the gateway device (or premise devices coupled to the gateway device). More specifically, in some embodiments, in order to manage these different premise devices, applications may be developed for execution on gateway devices (or devices coupled to a gateway device). For example, **a server may host various ones of these applications and may**

Claims of '823 Patent	Prior Art Disclosure
	facilitate selection and deployment of these applications on the gateway device, or devices coupled to the gateway device. In another embodiment, applications may be developed that execute on a server coupled to a gateway device, where the application executes on the server to manage or control a gateway device (and/or devices associated with the gateway device), or multiple gateway devices. The application may instead execute on the server to manage or control premise devices that are connected to one or more gateway devices. Stewart at ¶[0010] For example, a software development kit (SDK) or application programming interface (API) may be used to develop applications that can be installed on the gateway device. As noted above, these applications may also be executable by other devices as well. For example, **the applications may be executable by the gateway device, a server, and/or other devices (e.g., user devices)**…. Stewart at ¶[0011] **These applications may be developed and uploaded to a server, referred to as an application server. The application server may execute software to operate as an "application store".** The applications described herein are also referred to as "packages", and thus the server may be said to host a "package store" or "PACKAGE WAREHOUSE™". Thus various different application developers (or "package developers") may create various different applications that the upload to the application server for purchase by customers (e.g., merchants) who are operating gateway devices. Stewart at ¶[0012] The computer system may be in communication with a server, referred to as a **management server (e.g., a cloud server, as described below). The management server may be located on the Internet, or may be in direct communication with the gateway device (e.g., without accessing the Internet), and/or both, as desired. The gateway device may be in direct communication with the management server to** report information gathered by the devices or **perform control actions**, among other possibilities. **A user may be able to configure the gateway device** directly (e.g., via a web site hosted by the gateway device or via an application executing on the computer system) or may be able to configure the gateway device over the Internet, **via the management server**. Stewart at ¶[0054] **As shown in FIG. 1A (and as discussed above), the gateway device may be in communication with one or more servers (e.g., which may be a single server or a server farm, as desired) over the Internet connection**. The one or more servers (which may be a "cloud server" or "cloud servers") may provide

Claims of '823 Patent	Prior Art Disclosure
	various functionality for the convenience store, the gateway device, and/or other devices or users, as desired. Stewart at ¶[0055] In one embodiment, the management server may achieve various features described above by executing one or more "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. 1A. Embodiments of the invention provide an improved system for developers to create and sell such applications and an improved system for users to browse and purchase such applications. Stewart at ¶[0067] As described herein, an application server may store and/or provide a plurality of "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. 1A. **The application server may be the same server as the management server, or may be a different server. Each of the management server and the application server may be cloud servers. In the following description, the terms "server" and "cloud server" are used to refer generically either or both of the management server and the application server (and/or other associated servers).** Stewart at ¶[0068] FIG. 1B may operate similar to FIG. 1A except that the gateway device is placed in a home and controls premise devices of the home. Additionally, **in the embodiment of FIG. 1B, the separate management server and application server of FIG. 1A have been combined as a "cloud server"**. Note that these two servers may still be physically separate, or may be implemented by the same one or more servers with separate software functionality, as desired. Thus, the cloud server may be implemented as two separate sets of servers (one for management and one for applications) or may be implemented together, among other possibilities. Stewart at ¶[0078] In 702, a plurality of applications may be stored. The applications may be developed and/or received in the manner described above, with respect to FIG. 5, although other embodiments are envisioned. **The plurality of applications may be hosted by a server (e.g., in cloud storage accessible by the cloud server).** Stewart at ¶[0115]

Claims of '823 Patent	**Prior Art Disclosure**
	Stewart at Fig. 1A:  FIG. 1A Stewart at Fig 1B:  FIG. 1B
Claim 1 [c]	
wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *wherein the plurality of network device applications* (e.g., applications or application programs on the gateway device) *and the plurality of cloud applications* (e.g., applications or application programs on the cloud server) *are in secure communication with each other to form distributed applications* (e.g., executing applications "in a distributed fashion").

Claims of '823 Patent	**Prior Art Disclosure**
to form distributed applications; and	Specifically, Stewart discloses generally that the applications described therein may be executed in a distributed fashion, and thus constitute distributed applications, and also provides a specific example with respect to Fig. 6 of a distributed application. Stewart (Ex. 1004) at ¶¶ [0016], [0111], [0126] and Fig. 6. Stewart explicitly discloses distributed applications, including the application components executed by the gateway device and the application components executed by the cloud server. *See* Stewart (Ex. 1004) at ¶¶ [0016], [0111], [0126]. Stewart provides a specific example of such a distributed application with respect to its Figure 6, which is "a tank application that is configured to determine the fuel level in the fuel tank and present the fuel level information in a user interface" where "the tank application is executed in a distributed fashion, by both the gateway device (on the capture side) and the management server (on the presentation side)." *Id*. at ¶[0111] and Fig. 6. As discussed with respect to elements 1[a] and 1[b] above, the applications on the gateway device correspond to the *network device applications*, and applications on the management server (cloud server) correspond to *cloud applications*. Stewart also discloses that the gateway device is *in secure communication with* the server, thus resulting in the applications on the gateway device being in secure communication with the applications on the cloud server. Stewart (Ex. 1004) at ¶¶ [0051], [0085], [0227]-[0230]. For example, "the gateway device may include security software or circuitry, e.g., to provide a firewall for the Internet connection, perform communication encryption, protect credit card information provided over the network connection, etc." Stewart (Ex. 1004) at ¶[0051]. Additionally, "an application may request to communicate with a server or other entity over a network, such as the Internet. The gateway device may determine whether the communication is allowed using a security policy". Stewart (Ex. 1004) at ¶[0228]. ***See, e.g.*, Stewart** Once the gateway device has been configured with one or more applications, and hence has the ability to manage the coupled premise devices (e.g., via the installed applications), it may provide information reported by the premise devices to a server and/or control the premise devices in response to messages. For example, the reported information may be presented to a user, e.g., via the server. In one embodiment, the server may provide that information to a user that has logged in to the server (e.g., a website of the server). Alternatively, or additionally, the server may provide reported information to a device of the user (e.g., such as a cell phone or device controlled by the user), which may execute an application (or a plurality of applications) for providing an interface to the server or gateway device. In further embodiments, the user may be able to receive reported information and control various devices via a device coupled to the gateway device, e.g., via local area connection, such as when the Internet is down. In some embodiments, **the applications described above may be**

Claims of '823 Patent	Prior Art Disclosure
	executed in a distributed fashion, e.g., among the gateway device, the server, and/or other devices, such as user devices. Stewart at ¶[0016] FIG. 6 illustrates an exemplary use of APIs for a developed application. In the illustration of FIG. 6, the application is a tank application that is configured to determine the fuel level in the fuel tank and present the fuel level information in a user interface. In the embodiment of FIG. 6, **the tank application is executed in a distributed fashion, by both the gateway device (on the capture side) and the management server (on the presentation side)**. Stewart at ¶[0111] In one embodiment, the gateway device may provide a local area network (LAN) that at least a subset of the premise devices may connect to, e.g., in order to communicate with the gateway device and/or the Internet. The gateway device may provide a wireless hotspot for the convenience store, e.g., which may be accessible by employees of the convenience store and/or customers of the convenience store, as desired. In further embodiments, **the gateway device may include security software or circuitry, e.g., to provide a firewall for the Internet connection, perform communication encryption, protect credit card information provided over the network connection, etc**. Stewart at ¶[0051] In some embodiments, such as shown in the embodiment shown in FIG. 2, the gateway device may have a more direct connection to the server than through the Internet. **For example, the gateway device may be coupled to the server via a private network or alternate WAN, e.g., that may provide more security than over the Internet.** Alternatively, or additionally, the connection to the server may be via a virtual private network (VPN). Regardless of the type or method of connection, the gateway device may be configured to provide reported information to the server, which may store the information (and/or derived information) in the cloud based storage that is coupled to the server. Stewart at ¶[0085] **Note that where the application is distributed across a plurality of different devices, the server may provide the appropriate portions to each of the different devices for installation. For example, for an application that is executed by the server and the gateway device, the server may install its portion and provide the gateway device portion to the gateway device for installation.** Thus, the selected applications may be installed at their proper locations. Stewart at ¶[0126].

Claims of '823 Patent	Prior Art Disclosure
	In some embodiments, each application and/or device may have a security policy. For example, applications that handle sensitive information (e.g., credit card information) may be executed in a different portion of the gateway device (e.g., by secure logic of the gateway device) in order to securely handle the information. **In some embodiments, security policies may dictate how applications, devices, and/or third parties (e.g., communicating with the gateway device over the Internet) are able to communicate.** For example, an application for monitoring fuel tank devices may have a security policy that only allows communication with the fuel tank devices, and not, for example, devices that handle financial information. Similarly, the fuel tank devices (or applications associated therewith) may only provide the reported information to specific servers or third parties over the Internet. **The security policy may be implemented by the gateway device and/or management server.** For example, the gateway device may control how an application communicates with devices controlled by the gateway device as well as how the application provides information to third parties, e.g., over the Internet. Stewart at ¶[0227] In one embodiment, an application may request to communicate with one of the premise devices controlled by the gateway device, and the gateway device may determine whether or not that application is allowed, based on the security policy. In this embodiment, the gateway device may use rules to determine that a electronic sign application is only able to control or retrieve reported information from electronic signs, and not any other device. Accordingly, the request may be accepted if it meets the security policy and rejected if it does not. **Similarly, an application may request to communicate with a server or other entity over a network, such as the Internet. The gateway device may determine whether the communication is allowed using a security policy. For example, the application may only be allowed to communicate with specific IP addresses, domain names, etc., and may not be allowed to access other entities on the Internet (in order to protect the reported information)**. Stewart at ¶[0228] Similarly, third party communications may be controlled as well. For example, in one embodiment, all outside communication may be prohibited. In other embodiments, a server associated with a device or application may be allowed to provide incoming communications, but the communications may be restricted to certain applications or devices. For example, communications from the energy company may be restricted to applications or devices associated with that energy company. Similarly, any device communications may be controlled via security policies. Thus, the security policies may be used to control communications between third parties, applications, and/or devices. **The security policies may also be extended to providing information to or**

Claims of '823 Patent	Prior Art Disclosure
	receiving information from third parties on the cloud server (e.g., the management server), devices associated with users, etc. Stewart at ¶[0229] Alternatively, or additionally, some applications may communicate over a VPN based on security policies. Further types of policies may be assigned to each application, e.g., within a user interface provided by the management server. Stewart at ¶[0230]
Claim 1 [d]	
wherein the plurality of network device applications and plurality of cloud applications device [*sic*] form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications .	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *wherein the plurality of network device applications* (e.g., applications or application programs on the gateway device) *and plurality of cloud applications device* [*sic*[4]] (e.g., applications or application programs on the cloud server) *form unified capabilities* (e.g., programs on the gateway device and programs on the cloud server work together in a unified manner to perform the full functionality of the applications) *enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications* (e.g., the use of APIs to program the applications on both the gateway device and cloud server) *independent of network device hardware and cloud device hardware* (e.g., developed applications are compatible with the gateway device, the management server, and/or other devices that may execute portions of the application). Requester notes that this limitation recites *enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware*. The recitation of such "functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the prior art." MPEP §2114; *See also* Request at Section II(D). However, to the extent this limitation is given patentable weight, Stewart discloses enabling the use of APIs in its systems. Regarding that the applications on the network device and cloud device *form unified capabilities*, Stewart explicitly discloses distributed applications, including the application components executed by the gateway device (corresponding to the "plurality of network device applications" recited in claims 1 and 19 of the '823 Patent) and the application components executed by

[4] Requester notes that Claim 1[d] includes the word "device" after the first recitation of "cloud applications," which appears to be a typographical error.

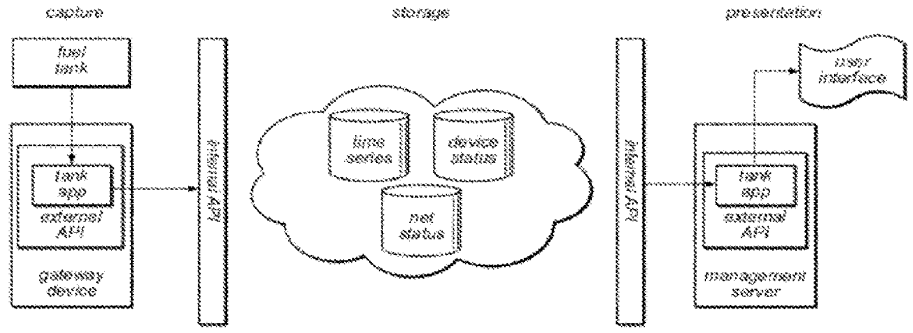
Claims of '823 Patent	Prior Art Disclosure
	the cloud server (corresponding to the "plurality of cloud applications" recited in claims 1 and 19 of the '823 Patent). *See* Stewart (Ex. 1004) at ¶¶ [0016], [0111], [0126]. Stewart also discloses that the applications can be distributed applications. *Id*. at ¶¶ [0016], [0111], [0126] and Fig. 6. Stewart provides a specific example of such a distributed application with respect to its Figure 6, which is "a tank application that is configured to determine the fuel level in the fuel tank and present the fuel level information in a user interface" where "the tank application is executed in a distributed fashion, by both the gateway device (on the capture side) and the management server (on the presentation side)." *Id*. at ¶[0111] and Fig. 6. While Stewart does not use the term "unified capabilities", identity of terminology is not required. MPEP §2131 (citing *In re Bond*, 910 F.2d 831 (Fed. Cir. 1990)). As a POSITA would understand, because the applications are distributed applications, the applications on the cloud server and the applications on the gateway device necessarily form unified capabilities, because that is the nature and purpose of distributed applications. Geier Decl. at ¶ 70. Moreover, it is clear from the teachings of Stewart that the distributed applications of Stewart work together in a unified manner to perform the full functionality of the applications, thus forming *unified capabilities*. Geier Decl. at ¶ 71. Regarding the element of *enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware*, Stewart also discloses that developers can use an application programming interface (API) to develop an application that is compatible with and can be installed on the gateway device, the management server, and/or other devices that may execute portions of the application. Stewart (Ex. 1004) at ¶¶ [0011], [0067], [0071]-[0072], [0074], [0106], [0118]-[0119]. Additionally, as shown in Figure 6 of Stewart, the distributed applications of Stewart can use APIs to facilitate the performance of the applications, and to program the gateway and server portions of the applications in the form of receiving and transferring information between them. Stewart (Ex. 1004) at ¶¶ [0111]-[0113] and Fig. 6; Geier Decl. at ¶ 73. While Stewart does not explicitly state that the applications are programmed by *upper layer* APIs *independent of network device hardware and cloud device hardware,* Stewart does teach that the developed applications "are compatible with the gateway device, the management server, and/or other devices that may execute portions of the application". Stewart (Ex. 1004) at ¶¶ [0106]. As would be understood and appreciated by a POSITA, it is the function and purpose of APIs to allow for programing in a manner that is independent of device hardware. Geier Decl. at ¶ 74. It is therefore an inherent characteristic of an API that it can be employed to program applications in a manner that is removed from or "independent of" (and in that sense, in an upper layer with respect to)

Claims of '823 Patent	Prior Art Disclosure
	"hardware." Geier Decl. (Ex. 1003) at ¶ 75. If a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation. MPEP §2114; *See also* Request at Section II(D). For these reasons, Stewart either explicitly or inherently discloses *enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware.* **(Ground 2, Obviousness)**:To the extent that Stewart does not anticipate this limitation, it is nonetheless rendered obvious in view of the knowledge of a POSITA for the reasons detailed in Section IV(C)(2) of the Request. *See also* Geier Decl. at ¶ 101. As discussed in the Request, the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware) was well within the knowledge of a POSITA prior to the June 13, 2023 priority date of the '823 Patent. Request at IV(C)(1); Geier Decl. at. at ¶ 106. Accordingly, a POSITA would have known about and understood the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware). *Id.* at ¶ 109. Additionally, a POSITA would have understood that the APIs of Stewart are "upper layer" APIs within the meaning of the '823 Patent, and that they would be enabled "to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" as recited in independent claims 1 and 19 of the '823 Patent. *Id.* ***See, e.g.*, Stewart** For example, a software development kit (SDK) or **application programming interface (API) may be used to develop applications that can be installed on the gateway device**. As noted above, these applications may also be executable by other devices as well. **For example, the applications may be executable by the gateway device, a server, and/or other devices (e.g., user devices)**…. Stewart at ¶[0011] In one embodiment, the management server may achieve various features described above by executing one or more "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. 1A. Embodiments of the invention provide an improved system for developers to create and sell such applications and an improved system for users to browse and purchase such applications.

Claims of '823 Patent	Prior Art Disclosure
	Stewart at ¶[0067] In one embodiment, the gateway device may not be aware of, or may not be able to manage, various ones of the premise devices of the convenience store. A user may accordingly install one or more applications in order to allow the gateway device to manage the convenience store devices. For example, the user may choose to install an application specific to the type of fuel tank device (or sensor) that is present at the convenience store. **Accordingly, the server may provide at least a portion of the application to the gateway device.** The gateway device may then install and execute the provided application on the gateway device, and the installed application may allow the gateway device to manage the fuel tank device (e.g., receive reported information from the fuel tank device and/or control the fuel tank device, as desired). In alternate embodiments, rather than a user specifying installation of the application, the server may automatically install the application without requiring user input selecting the application. For example, the gateway device may detect the presence of the fuel tank device and provide identification information to the server. The server may then automatically provide a corresponding application to allow the gateway device to communicate with the fuel tank device. Stewart at ¶[0071] **In some embodiments, a portion of the application may be installed on the management server.** For example, the application may include a first portion (provided to the gateway device) to adequately perform management of the fuel tank device, and the application may include a second portion for user interaction. **For example, the second portion of the application may be executed by the server** to provide the received reported information to various other devices for presentation, provide a user interface (e.g., a graphical user interface or "GUI") to control or modify settings associated with the fuel tank device, provide a user interface to set up fuel level thresholds (for when the fuel tank should be refilled), set up contact information associated with the fuel tank (e.g., whom to contact when the fuel level is below a threshold), purchase order information for requisitioning more fuel when the fuel tank is depleted, etc. The second portion may be a pluggable module into a management application (e.g., within a website) provided by the server. In one embodiment, the management server may include a database of available interfaces (e.g., associated with each application that is configured for execution on the management server) and, in response to an application being installed, associate the interface (or application portion) with the gateway device or user for which the application was installed. Thus, the application portion may already be present on the management server and may simply be activated upon installation (e.g., by associating a user or location with the application in

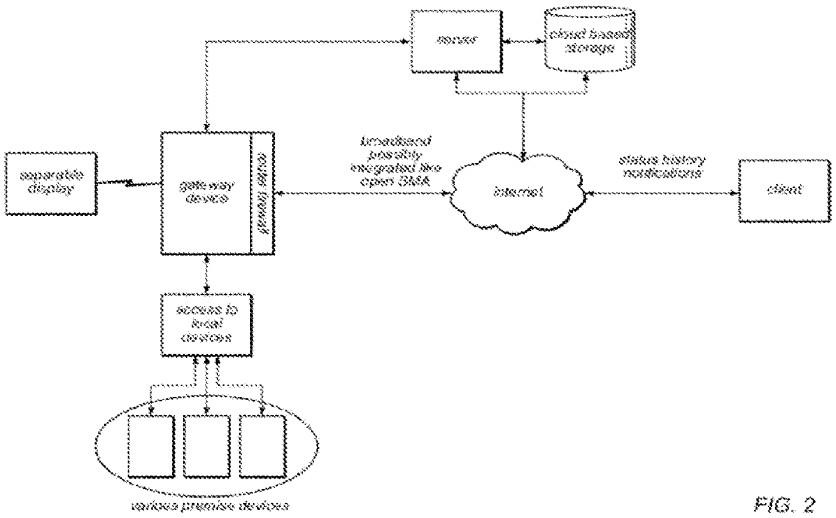
Claims of '823 Patent	Prior Art Disclosure
	a database). Thus, the second portion may "add on" to a management interface for the convenience store provided by the server. Stewart at ¶[0072] Similarly, applications specific to various POS devices, liquid dispensing machines, refrigeration devices, or any devices of the convenience store may be installed on the gateway device, the server, and/or other devices. These applications may be preferably developed by the manufacturers of the respective convenience store devices (e.g., where interaction with the devices uses a proprietary protocol); however, in alternate embodiments, the applications may be generated by any party (e.g., 3rd party developers). **In one embodiment, developers may use an API and/or SDK in order to develop the applications stored on the application server. The API or SDK may be an open API or SDK or may only be provided to authorized users or companies**. Stewart at ¶[0074] **The SDK and/or API may allow the developer to develop an application that is compatible with the gateway device, the management server, and/or other devices that may execute portions of the application.** In one embodiment, the SDK or API may be downloaded as executable program instructions (e.g., when downloaded as a development environment) and/or as documentation. For example, the SDK may include a compiler and/or simulator for compiling and testing the application, e.g., as it is developed. For example, the simulator may allow the developer to test execution on a simulated gateway device, simulated management server, simulated other device, or any combination (e.g., in order to test distributed execution of the application). Stewart at ¶[0106] **FIG. 6 illustrates an exemplary use of APIs for a developed application**. In the illustration of FIG. 6, the application is a tank application that is configured to determine the fuel level in the fuel tank and present the fuel level information in a user interface. In the embodiment of FIG. 6, the tank application is executed in a distributed fashion, by both the gateway device (on the capture side) and the management server (on the presentation side). Stewart at ¶[0111] As shown, **the tank application, executing on the gateway device**, may receive information from the fuel tank using a communication protocol supported by the fuel tank device (e.g., the device monitoring the fuel level in the fuel tank). **The tank application is also configured to provide the reported information for storage, e.g., in cloud storage on the Internet.** In order to perform this functionality, **the tank application interfaces with the gateway device using an external API, e.g., which may be provided in the SDK**. For example, the tank application may use a set of library functions to communicate and retrieve information from the fuel tank. Similarly, the tank application may use a set of

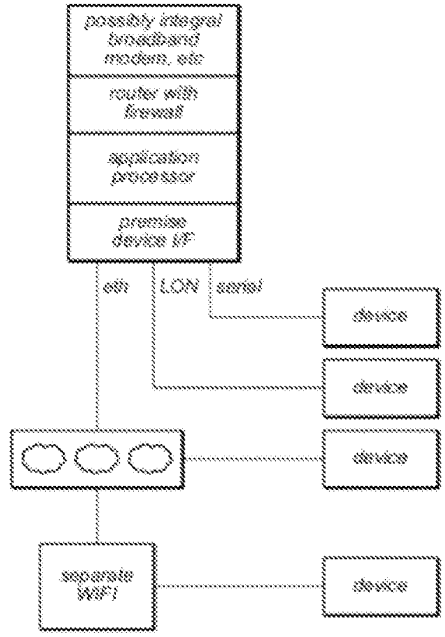
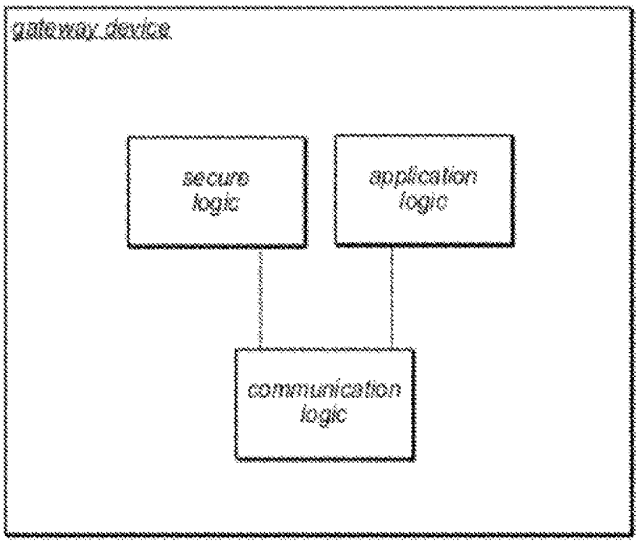
Claims of '823 Patent	Prior Art Disclosure
	library functions to provide the reported data for storage. **Accordingly, the gateway device may be configured to receive the reported information from the tank application and provide that information for storage via an internal API. The internal API, unlike the external API, may not be exposed or known to developers. Instead, the internal API may be the set of functions that the gateway device uses to perform the storage actions requested by the tank application.** Thus, the capture portion of the tank application may receive information from the fuel tank and provide information for storage using an external API. As shown, after provision to the cloud storage by the gateway device (e.g., using the internal API), time series data (e.g., representing the fuel levels of the fuel tank over time), device status, and net status information may be stored in the cloud storage, among other possibilities. Stewart at ¶[0112] On the presentation side, **a second portion of the tank application may execute on the management server. This portion of the tank application may use a presentation external API to request information stored in the cloud storage.** In response, the management server may use an internal API to retrieve the desired information. The tank application may also use the external API to display the retrieved information in a user interface provided by the management server. For example, the tank application may use the external API to generate a graph to provide the time series data of the fuel tank within a user interface framework provided by the management server (e.g., on a web page hosted by the management server). Thus, FIG. 6 illustrates an exemplary embodiment of a distributed application using internal and external APIs. Note that the fuel tank application is exemplary only and further types of applications, configurations, uses of APIs, etc. are envisioned. Additionally, further APIs and application portions may be used to display information on other devices (e.g., devices of users associated with the gateway device or location). Stewart at ¶[0113] Note that there may be applications that overlap with any of the above types of applications described herein. **Additionally, various ones of the applications may be executed only by the gateway device, the server, or user's devices, and various others of the applications may be executed by more than one of the above**. Stewart at ¶[0119] Stewart at Fig, 6:

Claims of '823 Patent	**Prior Art Disclosure**
	 FIG. 6
Claim 2	
The system of claim 1, wherein the programmable network device and the programmable cloud device each include an access network interface unit configured to send and receive communications and a processor with a memory associated with the access network interface unit	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *[t]he system of claim 1. See* claim 1, *supra*. Stewart discloses *wherein the programmable* (e.g., configurable) *network device* (e.g., gateway device) *and the programmable* (e.g., configurable) *cloud device* (e.g., cloud server) *each include an access network interface unit configured to send and receive communications* (e.g., suitable hardware and/or firmware to permit communication over the Internet) *and a processor* (e.g., processor) *with a memory* (e.g., memory or cloud storage) *associated with the access network interface unit* (e.g., suitable hardware and/or firmware to permit communication over the Internet) . Specifically, Stewart discloses that the "gateway device may be in communication with one or more servers… over the Internet connection." Stewart at ¶[0055]; *See also id.* at ¶¶[0017], and Figs. 3 and 4A. Accordingly, Stewart teaches that the gateway device and the cloud server each include an access network interface, since such a component would be necessary in order to connect to and communicate via the Internet. Additionally, the '823 Patent discloses that the network interface "may include any suitable hardware and/or firmware" such as "a wired network adapter, a wireless network adapter, a telephone modem, and/or a wireless modem." '823 Patent at 9:41-51. Likewise, Stewart discloses that the gateway device "may include a broadband modem" or other suitable hardware or firmware to provide an Internet connection. Stewart at ¶¶[0050], [0094] and Fig. 4A. Stewart also discloses that the gateway device and the cloud server each have a memory associated with the access network interface. For example, Stewart discloses that the gateway device may include

Claims of '823 Patent	Prior Art Disclosure
	"circuitry such as an ASIC or FPGA and/or a processor and memory storing program instructions to provide Internet connection," Stewart at ¶[0050], and that the gateway device includes both communication logic and applications logic, each of which include memory. Stewart at ¶¶[0094], [0096]. [0098]-[0099] and Figs. 4A and 4B. Stewart also discloses that the cloud server includes cloud based storage, which is memory, which, among other things, may store information received from the gateway device via the Internet. Stewart at ¶[0085] and Fig. 3. ***See, e.g.*, Stewart** In addition, as noted above, **the server which interfaces to a respective gateway device** may be configured with one or more applications to allow it to manage the gateway device (or multiple gateway devices), manage premise devices coupled to the gateway devices, receive and process data received from one or more gateway devices, etc…. Stewart at ¶[0017] As shown, the gateway device may be in communication with a network, e.g., a wide area network (WAN), such as the Internet. **The gateway device may be in communication with the Internet via a router or modem that provides the Internet connection. Alternatively, the gateway device may itself include logic (e.g., circuitry such as an ASIC or FPGA and/or a processor and memory storing program instructions) to perform these wide area network connectivity functions. For example, the gateway device may include a broadband modem for providing a communication link between the convenience store and the Internet. The gateway device may also act as a router (e.g., a wired and/or wireless router) which couples to the wide area network and premise devices at the location**. Stewart at ¶[0050] As shown in FIG. 1A (and as discussed above), the **gateway device may be in communication with one or more servers (e.g., which may be a single server or a server farm, as desired) over the Internet connection**. The one or more servers (which may be a "cloud server" or "cloud servers") may provide various functionality for the convenience store, the gateway device, and/or other devices or users, as desired. Stewart at ¶[0055] In some embodiments, such as shown in the embodiment shown in FIG. 2, **the gateway device may have a more direct connection to the server than through the Internet. For example, the gateway device may be coupled to the server via a private network or alternate WAN, e.g., that may provide more security than over the Internet. Alternatively, or additionally, the connection to the server may be via a virtual private network (VPN).**

Claims of '823 Patent	Prior Art Disclosure
	Regardless of the type or method of connection, the gateway device may be configured to provide reported information to the server, which may store the information (and/or derived information) in the **cloud based storage that is coupled to the server**. Additionally, one or more clients may be coupled to the Internet and may receive various reported information or notifications from the gateway device and/or the server. Further, as indicated above, the server and/or clients may control various devices via the gateway device. Note that the clients may perform this control by communicating with the server or by directly communicating with the gateway device, as desired. Stewart at ¶[0085] FIGS. 4A and 4B illustrate exemplary block diagrams of the gateway device, according to various embodiments. **As shown in FIG. 4A, the gateway device may include an integrated broadband modem or other circuitry for establishing a connection to the Internet or another network (e.g., a DSL modem, cable modem, and/or fiber modem, among other possibilities).** The gateway device may also include a router for providing a local area network (e.g., a wired and/or wireless LAN) to various devices. Alternatively, the gateway device may be connected to a modem and/or router, e.g., when these are not integrated into the gateway device. Stewart at ¶[0094] The **gateway device may further include** logic for performing the functionality described herein. For example, the gateway device may include one or more field programmable gate arrays (FPGAs), integrated circuits (e.g., ASICs), and **processors and memory to perform the functionality. In some embodiments, the processor and memory may be used to store and execute applications provided by the server**. Stewart at ¶[0096] FIG. 4B illustrates a different block diagram of the gateway device. In the embodiment of FIG. 4B, **the gateway device may include communication logic and applications logic, and possibly secure applications logic. The communication logic may include any processors and memories and/or circuitry for communicating with the Internet, communicating with the premise devices at the location, providing routing functionality, providing modem functionality, etc**. For example, the communication logic may include routing logic, wireless communication logic, wired communication logic, broadband modem logic, firewall logic, etc. **In some embodiments, the communication logic may include a processor and memory, and the memory may store program instructions that are executable to perform the routing functionality or other communication functionality**. Stewart at ¶[0098]

	The application logic may include a processor and memory, and the memory may store applications (e.g., provided by the applications server) that may be executed by the processor. The application logic may include other circuitry, as desired. Stewart at ¶[0099] Stewart at Fig. 3:  FIG. 2 Stewart at Fig. 4A:

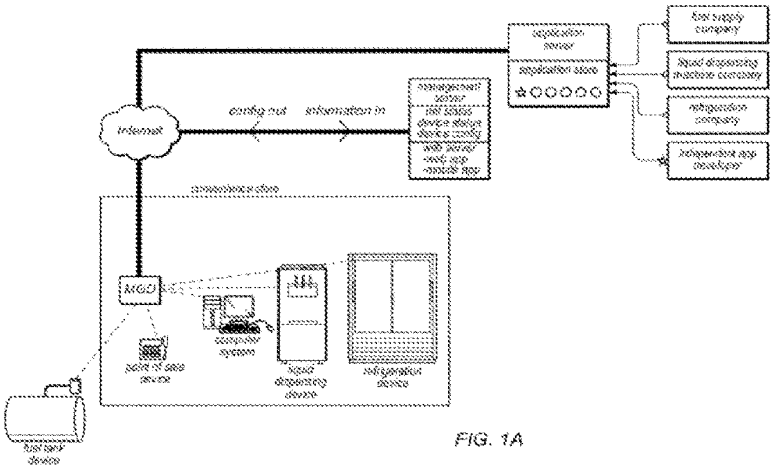
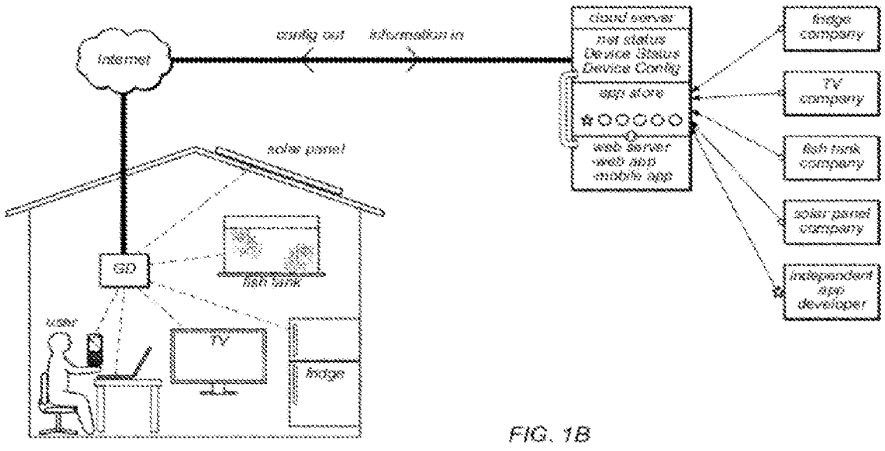
Claims of '823 Patent	Prior Art Disclosure
	 FIG. 4A Stewart at Fig. 4B: 
Claim 3 [P]	

Claims of '823 Patent	**Prior Art Disclosure**
The system of claim 1 further comprising:	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *[t]he system of claim 1. See* claim 1, *supra*.
Claim 3 [a]	
an application management portal capable of managing life cycles of the distributed applications; and	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *an application management portal* (e.g., a management server) *capable of managing life cycles* (e.g., installing, modifying, and removing or replacing applications) *of the of the distributed applications* (e.g., applications or application programs). Requester notes that being "capable of managing life cycles of the distributed applications" is functional language. The recitation of such "functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the prior art", and additionally, if a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation. MPEP §2114; *See also* Request at Section II(D). To the extent that this limitation is given any patentable weight, it is disclosed either expressly or inherently in Stewart. Additionally, to the extent that Stewart does not explicitly disclose the management server managing life cycles of the plurality of network device applications and plurality of cloud applications, it is clear based on the disclosure of Stewart that the management server of Stewart would be capable of such management. Geier Decl. at ¶ 78. Specifically, Stewart discloses a management server and an application server that are in communication with the gateway device. Stewart (Ex. 1004) at ¶¶[0054]-[0055], [0068], [0078]. and Figs. 1A and 1B. "The application server may be the same server as the management server, or may be a different server. Each of the management server and the application server may be cloud servers." *Id.* at ¶ [0068]. As to *managing life cycles of the of the distributed applications* in particular, Stewart discloses that the management server can be used to "perform control actions" and "configure the gateway device," including installing, modifying, and removing or replacing applications. Stewart (Ex. 1004) at ¶[0054]. Stewart discloses that the management server can be used to select one or more applications to be installed and executed by "the gateway device and/or the management server (and possibly by devices associated with the convenience

Claims of '823 Patent	Prior Art Disclosure
	store)". Stewart (Ex. 1004) at ¶¶[0066]-[0069]. For example, Stewart discloses that "a manager may browse the application server and select from a plurality of different applications that may be installed." Stewart (Ex. 1004) at ¶[0069]. Stewart also generally discloses that the installed applications can be modified using the management server. Stewart (Ex. 1004) at ¶¶[0066]. Stewart also discloses that some applications may not be removable (thus indicating that other applications are removable), but that such applications can be replaced. Stewart (Ex. 1004) at ¶¶[0223]. Accordingly, Stewart discloses the management server *managing life cycles of the distributed* applications. ***See, e.g.*, Stewart** The computer system may be in communication with a server, referred to as a **management server (e.g., a cloud server, as described below). The management server may be located on the Internet, or may be in direct communication with the gateway device (e.g., without accessing the Internet), and/or both, as desired. The gateway device may be in direct communication with the management server to** report information gathered by the devices or **perform control actions**, among other possibilities. **A user may be able to configure the gateway device** directly (e.g., via a web site hosted by the gateway device or via an application executing on the computer system) or may be able to configure the gateway device over the Internet, **via the management server**. Stewart at ¶[0054] **As shown in FIG. 1A (and as discussed above), the gateway device may be in communication with one or more servers (e.g., which may be a single server or a server farm, as desired) over the Internet connection**. The one or more servers (which may be a "cloud server" or "cloud servers") may provide various functionality for the convenience store, the gateway device, and/or other devices or users, as desired. Stewart at ¶[0055] The **management server may also allow for configuration of the gateway device or premise devices managed by the gateway device in the convenience store**. For example, a manager may wish to update prices (e.g., of an electronic sign, a fuel dispenser, various food items, etc.) at the convenience store. For gas prices in particular, the manager may access the management server via an application executing on a computer system or device (such as a web browser or custom application) and set the gas prices displayed on electronic signs at the convenience store, the prices of gas at the fuel pumps (e.g., on the POS devices), etc. Thus, instead of the manager calling an employee of the convenience store to change all of these settings, the manager may simply change them remotely without interrupting any work at the convenience store. The manager (or any other authorized user) may be able to modify other settings as well. For example,

Claims of '823 Patent	Prior Art Disclosure
	the manager may change the current temperature of the convenience store by changing the desired temperature of heaters or coolers at the convenience store. The manager may be able to modify the temperatures of various refrigerator units, heating units, or the settings of any controllable device at the convenience store. **The manager may be able to configure any device or setting at the convenience store using the management server**. Stewart at ¶[0066] In one embodiment, the **management server may achieve various features described above by executing one or more "applications" or "packages" (described below as "applications")** for use for the convenience store, in the example of FIG. 1A. Embodiments of the invention provide an improved system for developers to create and sell such applications and an improved system for users to browse and purchase such applications. Stewart at ¶[0067] As described herein, an application server may store and/or provide a plurality of "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. 1A. **The application server may be the same server as the management server, or may be a different server. Each of the management server and the application server may be cloud servers. In the following description, the terms "server" and "cloud server" are used to refer generically either or both of the management server and the application server (and/or other associated servers).** Stewart at ¶[0068] For example, a **manager may browse the application server and select from a plurality of different applications that may be installed and executed by the gateway device and/or the management server (and possibly by devices associated with the convenience store).** There are many different types of applications that are possible and may be desired by a manager or other user of the convenience store. While the following descriptions provide some examples of such applications, these are in no means limiting, and further applications are envisioned. Stewart at ¶[0069] FIG. 1B may operate similar to FIG. 1A except that the gateway device is placed in a home and controls premise devices of the home. Additionally, **in the embodiment of FIG. 1B, the separate management server and application server of FIG. 1A have been combined as a "cloud server"**. Note that these two servers may still be physically separate, or may be implemented by the same one or more servers with separate software functionality, as desired. Thus, the cloud server may be implemented as two separate sets of servers (one for

Claims of '823 Patent	Prior Art Disclosure
	management and one for applications) or may be implemented together, among other possibilities. Stewart at ¶[0078] **Some applications may be default or "irreplaceable"** applications that are always installed on gateway devices (or always installed for specific types of gateway devices). For example, one such application may be a device reporting application that maintains a list of premise devices that are detected and/or controlled by the gateway device. A user may be able to use the application to view or manage premise devices connected to the gateway device. Accordingly, such an application may be installed on every gateway device. **Such applications may not be removable from the device (e.g., a user may not be able to remove the application from the gateway device or management server).** However, third parties may develop an application with a similar purpose as one of these default or irreplaceable applications. **In one embodiment, the third party and/or the application server may mark the application as a replacement (or candidate for replacement) for the default or irreplaceable applications.** Accordingly, when such an application is selected for installation, the original application may be removed and replaced with the new application (e.g., in response to user input confirming the replacement or automatically, as desired). In some embodiments, the new application may be associated with a newer device (or piece of hardware). For example, a user may purchase a new switch which may have nicer features with respect to device management. Accordingly, the user may install a corresponding application that is able to use these nicer features. However, the replacement application may be purely functional (e.g., not associated with new hardware or a specific hardware device) and the user may simply wish to replace the original default software application with a different third party application (e.g., which is better or that the user prefers). Stewart at ¶[0223]

Claims of '823 Patent	**Prior Art Disclosure**
	Stewart at Fig. 1A:  FIG. 1A Stewart at Fig. 1B:  FIG. 1B
Claim 3 [b]	
an infrastructure application marketplace in communication with the application management portal, said	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *an infrastructure application marketplace* (e.g., application server operating as an application store) *in communication with* (e.g., the application server operating as an application store communicates over the Internet with the management server) *the application management portal* (e.g., management server)*, said infrastructure marketplace capable of providing*

Claims of '823 Patent	**Prior Art Disclosure**
infrastructure marketplace capable of providing tested and certified distributed applications to the application management portal.	*tested and certified distributed applications* (e.g., applications can be tested using the SDK and by the server administrator, which are then provided by the application server operating as an application store) *to the application management portal* (e.g., the application server providing the tested applications to the management server). Requester notes that being "capable of providing tested and certified distributed applications to the application management portal" is functional language. The recitation of such "functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the prior art," and additionally, if a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation. MPEP §2114; *See also* Request at Section II(D). To the extent that this limitation is given any patentable weight, it is disclosed either expressly or inherently in Stewart. Moreover, to the extent that Stewart does not disclose this element though explicit disclosure of providing such applications, based on the disclosure of Stewart discussed below, it is clear that the application store of Stewart would be "capable of" providing such applications. Geier Decl. at ¶ 84. As to *an infrastructure application marketplace*, Stewart discloses that "applications may be developed and uploaded to a server, referred to as an application server. The application server may execute software to operate as an 'application store'." Stewart (Ex. 1004) at ¶[0012]. Stewart also discloses a website with respect to which a user may log on and choose from a plurality of applications for installation. *Id*. at [0013]. Applications can be selected manually or automatically from the application store and be installed and executed by the gateway device and/or the management server (and by other devices associated with the gateway device). *Id*. at ¶[0013]; *See also* Id. at ¶¶[0067]-[0069], and [0116] and Figs. 1A and 1B. As to the *infrastructure application marketplace* being *in communication with the application management portal*, Stewart discloses that the application server and the management server can both be cloud servers and that they communicate with each other via the Internet, and even that "The application server may be the same server as the management server". Stewart (Ex. 1004) at ¶[0068], Fig. 1A and Fig. 1B. Stewart also discloses that applications selected in the application store are then installed by the management server. *Id*. at ¶[0069]. As to *said infrastructure marketplace capable of providing tested and certified distributed applications to the application management portal*, Stewart also discloses that the applications can be tested, both by the developer and by an administrator of the application server, prior to being made available in the

Claims of '823 Patent	Prior Art Disclosure
	application store. Stewart (Ex. 1004) at ¶¶[0106] and [0108]. For example, referring to its Figure 5, Stewart discloses a method for developing an application that includes an approval process in which "an administrator of the application server (or other authorized user) may review the source code, test execution of the application, etc." prior to the application being made available for installation. Stewart (Ex. 1004) at ¶[0108]. Such an approval process would constitute both testing and certifying of the applications. Geier Decl. at ¶ 87. After the approval process, the tested application is then made available via the application server operating as an application store for installation on a device. Stewart (Ex. 1004) at ¶[0110]. Accordingly, in view of Stewart's explicit disclosure of testing an application, Stewart's application server is *capable of providing tested and certified distributed applications.* ***See, e.g.*, Stewart** These **applications may be developed and uploaded to a server, referred to as an application server. The application server may execute software to operate as an "application store".** The applications described herein are also referred to as "packages", and thus the server may be said to host a "package store" or "PACKAGE WAREHOUSE™". Thus various different application developers (or "package developers") may create various different applications that the upload to the application server for purchase by customers (e.g., merchants) who are operating gateway devices. Stewart at ¶[0012] **The applications may be selected for installation in various manners. For example, the applications may be selected for installation manually, via user selection**. In one embodiment, the application server hosting the application store maintains a website for the application store (or otherwise presents a user interface) that is accessible by various client systems, e.g., over the Internet. A user may browse (using a web browser) through a plurality of available applications in the application store and then select desired applications for installation, e.g., on the gateway device, the server, and/or other devices. The user may also specify the targets for installation of the selected applications, although this may be determined automatically, as desired. **Alternatively, or additionally, the applications may be selected for installation in an automatic fashion**. Stewart at ¶[0013] In one embodiment, the management server may achieve various features described above by executing one or more "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. 1A. **Embodiments of the invention provide an improved system for developers to create and sell such applications and an improved system for users to browse and purchase such applications**.

Claims of '823 Patent	Prior Art Disclosure
	Stewart at ¶[0067] As described herein, **an application server may store and/or provide a plurality of "applications" or "packages" (described below as "applications")** for use for the convenience store, in the example of FIG. 1A. **The application server may be the same server as the management server**, or may be a different server. Each of the management server and the application server may be cloud servers. In the following description, the terms "server" and "cloud server" are used to refer generically either or both of the management server and the application server (and/or other associated servers) Stewart at ¶[0068] For example, **a manager may browse the application server and select from a plurality of different applications that may be installed and executed by the gateway device and/or the management server (and possibly by devices associated with the convenience store).** There are many different types of applications that are possible and may be desired by a manager or other user of the convenience store. While the following descriptions provide some examples of such applications, these are in no means limiting, and further applications are envisioned. Stewart at ¶[0069] **The SDK and/or API may allow the developer to develop an application that is compatible with the gateway device, the management server, and/or other devices that may execute portions of the application**. In one embodiment, the SDK or API may be downloaded as executable program instructions (e.g., when downloaded as a development environment) and/or as documentation. **For example, the SDK may include a compiler and/or simulator for compiling and testing the application, e.g., as it is developed. For example, the simulator may allow the developer to test execution on a simulated gateway device, simulated management server, simulated other device, or any combination (e.g., in order to test distributed execution of the application)**. Stewart at ¶[0106] In 506, after the application has been developed, the application may be submitted to an application server (or "server"). For example, the developer may upload the application (e.g., in a compiled form and/or in source code form) to the application server. **An approval process may be required before the application is available for installation on gateway devices, the management server, other devices, etc. For example, an administrator of the application server (or other authorized user) may review the source code, test execution of the application, etc**. in order to ensure that the application behaves as it is intended and/or will not cause any issues for users installing the application (e.g., on gateway devices, the management server, or other devices). The testing may

Claims of '823 Patent	Prior Art Disclosure
	occur on each corresponding device that the application is intended to execute on. Stewart at ¶[0108] In 508, the application may be installed on one or more gateway devices. As described further below, users may browse the applications stored on the application server and select ones for installation on the gateway device (or other devices). Alternatively, applications on the application server may be automatically determined as described herein. 508 may be performed in the manner described below. Stewart at ¶[0110] In 704, one or more applications may be selected for installation. The one or more applications may be selected for installation manually or automatically as desired. For example, a user, such as an administrator of a location (e.g., a merchant location, such as a convenience store or a home, among other possibilities) or a plurality of locations (e.g., a chain of stores of a merchant), may wish to install an application from the plurality of available applications. Accordingly, the user may access the application server to browse or search the plurality of applications. For example, in one embodiment, the application server may host a web site and the user may log on to the web site to choose from the plurality of applications for installation. Alternatively, or additionally, the user may use a locally installed program that is in communication with the application server to select one of the applications for installation. For example, the locally installed program may retrieve the list of applications available from the application server and provide them for selection on a display of a client used by the user. In one embodiment, the user may access the application server (via webpage, application, or other mechanism) to visit a "store" of applications, which may be referred to as an application store, app store, PACKAGE WAREHOUSE.TM., PACKAGE STORE.TM., etc. Stewart at ¶[0116]

Stewart at Fig. 1A:

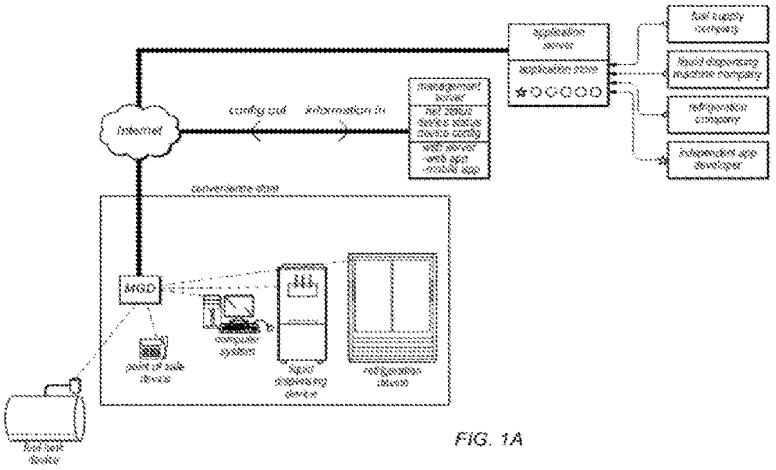


FIG. 1A

Stewart at Fig. 1B:

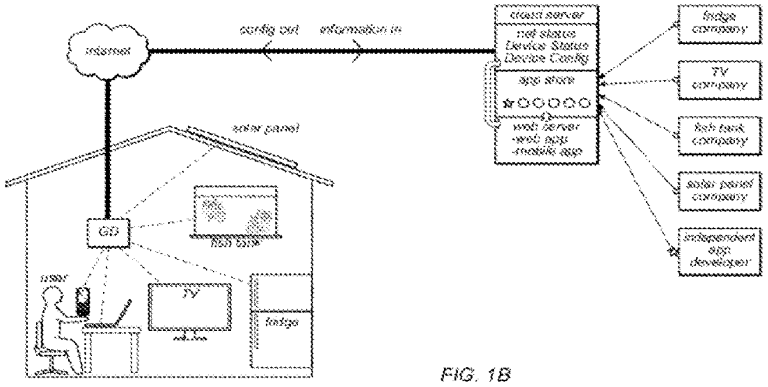
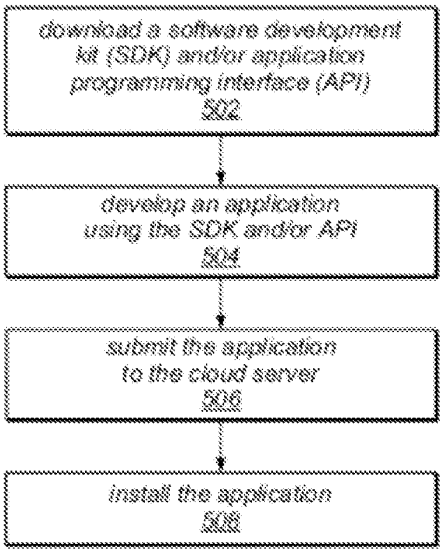


FIG. 1B

Claims of '823 Patent	**Prior Art Disclosure**
	Stewart at Fig. 5:  FIG. 5
Claim 4	
The system of claim 3, wherein the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *[t]he system of claim 3*. *See* claim 3, *supra*. Stewart discloses that *the application management portal* (e.g., management server) *is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace* (e.g., application server operating as an application store); *testing said distributed applications prior to deployment; provisioning said distributed applications* (e.g., management server installing applications)*; and deprovisioning said distributed applications*. Requester notes that this limitation recites a Markush group. MPEP §2117. Accordingly, disclosure of any one of the options within the group anticipates the limitation. Requester also notes that being "capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications" is

Claims of '823 Patent	**Prior Art Disclosure**
said distributed applications.	functional language. The recitation of such "functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the prior art", and additionally, if a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation. MPEP §2114; *See also* Request at Section II(D). To the extent that this limitation is given any patentable weight, it is disclosed either expressly or inherently in Stewart. To the extent that Stewart does not disclose this element though explicit disclosure of providing such applications, based on the disclosure of Stewart discussed below, it is clear that the application store of Stewart would be "capable of" providing such applications. Geier Decl. at ¶ 92. Specifically, Stewart discloses that applications may be developed and uploaded to the application server, and that the management server may obtain and execute one or more of the applications. Stewart at ¶¶ [0012]-[0013] and [0067]-[0068]. Applications can be selected manually or automatically from the application store and be installed and executed by the gateway device and/or the management server (and by other devices associated with the gateway device). *Id*. at ¶[0013]. Thus, the management server of Stewart is capable of at least both *receiving new applications from the infrastructure application marketplace* and *provisioning said distributed applications*. ***See, e.g.*, Stewart** These applications may be developed and uploaded to a server, referred to as an application server. The application server may execute software to operate as an "application store". The applications described herein are also referred to as "packages", and thus the server may be said to host a "package store" or "PACKAGE WAREHOUSE™". Thus various different application developers (or "package developers") may create various different applications that the upload to the application server for purchase by customers (e.g., merchants) who are operating gateway devices. Stewart at ¶[0012] The applications may be selected for installation in various manners. For example, the applications may be selected for installation manually, via user selection. In one embodiment, the application server hosting the application store maintains a website for the application store (or otherwise presents a user interface) that is accessible by various client systems, e.g., over the Internet. A user may browse (using a web browser) through a plurality of available applications in the application store and then select desired applications for

Claims of '823 Patent	**Prior Art Disclosure**
	installation, e.g., on the gateway device, the server, and/or other devices. The user may also specify the targets for installation of the selected applications, although this may be determined automatically, as desired. Alternatively, or additionally, the applications may be selected for installation in an automatic fashion. Stewart at ¶[0013] In one embodiment, the management server may achieve various features described above by executing one or more "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. 1A. Embodiments of the invention provide an improved system for developers to create and sell such applications and an improved system for users to browse and purchase such applications. Stewart at ¶[0067] As described herein, an application server may store and/or provide a plurality of "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. 1A. The application server may be the same server as the management server, or may be a different server. Each of the management server and the application server may be cloud servers. In the following description, the terms "server" and "cloud server" are used to refer generically either or both of the management server and the application server (and/or other associated servers) Stewart at ¶[0068]
Claim 8	
The system of claim 3 further comprising: a distributed notification service wherein notice of a predetermined event is sent from at least one of the plurality of network device applications to at least one of the plurality of cloud applications.	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart disclsoes *[t]he system of claim 3. See* claim 3, *supra*. Stewart discloses *a distributed notification service* (e.g., gateway device sending messages to the management server) *wherein notice of a predetermined event* (e.g., messages indicating specific alarms or events associated with premise devices) *is sent from at least one of the plurality of network device applications* (e.g., applications or application programs on the gateway device, such as the "first portion" of the example fuel tank application) *to at least one of the plurality of cloud applications* (e.g., applications or application programs on the cloud server, such as the "second portion" of the example fuel tank application). Specifically, Stewart discloses that the management server executes applications, and that once the gateway device is configured with one or more applications it may provide information reported by the premise devices to the management server. Stewart at ¶¶[0016], [0067]. For example, the gateway

Claims of '823 Patent	Prior Art Disclosure
	device may send to the management server messages indicating specific alarms or events associated with the premise devices managed by the gateway device. Stewart at ¶[0057]. Stewart discloses an example of such communication between the gateway device and the management server in which a distributed application includes "a first portion (provided to the gateway device) to adequately perform management of the fuel tank device" and "a second portion for user interaction" where "the second portion of the application may be executed by the server to provide the received reported information to various other devices for presentation, provide a user interface (e.g., a graphical user interface or "GUI") to control or modify settings associated with the fuel tank device, provide a user interface to set up fuel level thresholds (for when the fuel tank should be refilled), set up contact information associated with the fuel tank (e.g., whom to contact when the fuel level is below a threshold), purchase order information for requisitioning more fuel when the fuel tank is depleted, etc." Stewart at ¶[0072]. As noted in Stewart, the server can be a cloud server. Stewart at ¶[0068]. Accordingly, in the example above, the first portion of the distributed application on the gateway device sends information to the second portion of the distributed application on the cloud server, such as notifications regarding fuel level. ***See, e.g.*, Stewart** **Once the gateway device has been configured with one or more applications, and hence has the ability to manage the coupled premise devices (e.g., via the installed applications), it may provide information reported by the premise devices to a server and/or control the premise devices in response to messages.** For example, the reported information may be presented to a user, e.g., via the server. In one embodiment, the server may provide that information to a user that has logged in to the server (e.g., a website of the server). Alternatively, or additionally, the server may provide reported information to a device of the user (e.g., such as a cell phone or device controlled by the user), which may execute an application (or a plurality of applications) for providing an interface to the server or gateway device. In further embodiments, the user may be able to receive reported information and control various devices via a device coupled to the gateway device, e.g., via local area connection, such as when the Internet is down. In some embodiments, the applications described above may be executed in a distributed fashion, e.g., among the gateway device, the server, and/or other devices, such as user devices. Stewart at ¶[0016] **In some embodiments, the management server may receive messages indicating specific alarms or events associated with the premise devices managed by the gateway device.** For example, the gateway device may send information indicating when a temperature of the refrigeration device exceeds a threshold (e.g., based on food safety requirements or otherwise); when the fuel

Claims of '823 Patent	Prior Art Disclosure
	level in the fuel tank is below a threshold level; when the liquid dispensing device is low on a respective liquid, out of ice, etc.; when the POS performs or attempts to perform a transaction above a certain level, etc. The gateway device may also send information when a security alarm has been detected (e.g., when the convenience store is broken in to), when an employee triggers an alarm (e.g., when the convenience store is being robbed), etc. Stewart at ¶[0057] In one embodiment, the **management server may achieve various features described above by executing one or more "applications" or "packages" (described below as "applications")** for use for the convenience store, in the example of FIG. 1A. Embodiments of the invention provide an improved system for developers to create and sell such applications and an improved system for users to browse and purchase such applications. Stewart at ¶[0067] As described herein, an application server may store and/or provide a plurality of "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. 1A. The application server may be the same server as the management server, or may be a different server. **Each of the management server and the application server may be cloud servers. In the following description, the terms "server" and "cloud server" are used to refer generically either or both of the management server and the application server (and/or other associated servers)**. Stewart at ¶[0068] In some embodiments, a portion of the application may be installed on the management server. For example, **the application may include a first portion (provided to the gateway device) to adequately perform management of the fuel tank device, and the application may include a second portion for user interaction. For example, the second portion of the application may be executed by the server to provide the received reported information to various other devices for presentation, provide a user interface (e.g., a graphical user interface or "GUI") to control or modify settings associated with the fuel tank device, provide a user interface to set up fuel level thresholds (for when the fuel tank should be refilled), set up contact information associated with the fuel tank (e.g., whom to contact when the fuel level is below a threshold), purchase order information for requisitioning more fuel when the fuel tank is depleted, etc.** The second portion may be a pluggable module into a management application (e.g., within a website) provided by the server. In one embodiment, the management server may include a database of available interfaces (e.g., associated with each application that is configured for execution on the management server) and, in response to an application being installed, associate the interface (or application

Claims of '823 Patent	**Prior Art Disclosure**
	portion) with the gateway device or user for which the application was installed. Thus, the application portion may already be present on the management server and may simply be activated upon installation (e.g., by associating a user or location with the application in a database). Thus, the second portion may "add on" to a management interface for the convenience store provided by the server. Stewart at ¶[0072]
Claim 15	
The system of claim 1 further comprising: a distributed content service wherein contents generated by the at least one of the plurality of network device applications are shared with at least one of the plurality of cloud applications.	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *[t]he system of claim 1. See* claim 1, *supra*. Stewart discloses *a distributed notification service* (e.g., gateway device sending messages to the management server messages) *wherein contents* (e.g., information) *generated by the at least one of the plurality of network device applications* (e.g., applications or application programs on the gateway device, such as the "first portion" of the example fuel tank application) *are shared with at least one of the plurality of cloud applications* (e.g., applications or application programs on the cloud server, such as the "second portion" of the example fuel tank application). Specifically, Stewart discloses that the management server executes applications, and that once the gateway device is configured with one or more applications it may provide information reported by the premise devices to the management server. Stewart at ¶¶[0016], [0067]. For example, the gateway device may send to the management server messages comprising information gathered from each of (or at least a subset of) the plurality of premise devices managed by the gateway device. Stewart at ¶[0057]. Stewart discloses an example of such communication between the gateway device and the management server in which a distributed application includes "a first portion (provided to the gateway device) to adequately perform management of the fuel tank device" and "a second portion for user interaction" where "the second portion of the application may be executed by the server to provide the received reported information to various other devices for presentation, provide a user interface (e.g., a graphical user interface or "GUI") to control or modify settings associated with the fuel tank device, provide a user interface to set up fuel level thresholds (for when the fuel tank should be refilled), set up contact information associated with the fuel tank (e.g., whom to contact when the fuel level is below a threshold), purchase order information for requisitioning more fuel when the fuel tank is depleted, etc." Stewart at ¶[0072]. As noted in Stewart, the server

Claims of '823 Patent	Prior Art Disclosure
	can be a cloud server. Stewart at ¶[0068]. Accordingly, in the example above, the first portion of the distributed application on the gateway device sends information to the second portion of the distributed application on the cloud server, such as content regarding fuel level. ***See, e.g.*, Stewart** **Once the gateway device has been configured with one or more applications, and hence has the ability to manage the coupled premise devices (e.g., via the installed applications), it may provide information reported by the premise devices to a server and/or control the premise devices in response to messages.** For example, the reported information may be presented to a user, e.g., via the server. In one embodiment, the server may provide that information to a user that has logged in to the server (e.g., a website of the server). Alternatively, or additionally, the server may provide reported information to a device of the user (e.g., such as a cell phone or device controlled by the user), which may execute an application (or a plurality of applications) for providing an interface to the server or gateway device. In further embodiments, the user may be able to receive reported information and control various devices via a device coupled to the gateway device, e.g., via local area connection, such as when the Internet is down. In some embodiments, the applications described above may be executed in a distributed fashion, e.g., among the gateway device, the server, and/or other devices, such as user devices. Stewart at ¶[0016] **For example, the management server may be configured to receive and store information regarding the premise devices managed by the gateway device and possibly status information of the gateway device or the convenience store**. For example, the management server may store configuration information of the various premise devices, network status information of the connection between the management server and the gateway device, etc. **In one embodiment, the management server may be configured to receive messages comprising information gathered from each of (or at least a subset of) the plurality of premise devices at the convenience store and store reported information for the premise devices, e.g., in a database.** The reported information may indicate a current condition of the premise device, a condition of the environment of the location near the premise device, any data measured by the premise device, etc. For example, the management server may store the current temperature (and/or a history of temperatures) for a first refrigeration device at the convenience store. The management server may also receive messages regarding various transactions performed by a first POS device at the convenience store and store information regarding those transactions. The management server may also receive messages regarding the fuel level in a first fuel tank at the convenience store and store the current level (or a history of fuel

Claims of '823 Patent	Prior Art Disclosure
	levels) in response to the message. Other types of reported information or messages may be received by the management server. Stewart at ¶[0057] In one embodiment, the **management server may achieve various features described above by executing one or more "applications" or "packages" (described below as "applications")** for use for the convenience store, in the example of FIG. 1A. Embodiments of the invention provide an improved system for developers to create and sell such applications and an improved system for users to browse and purchase such applications. Stewart at ¶[0067] As described herein, an application server may store and/or provide a plurality of "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. 1A. The application server may be the same server as the management server, or may be a different server. **Each of the management server and the application server may be cloud servers. In the following description, the terms "server" and "cloud server" are used to refer generically either or both of the management server and the application server (and/or other associated servers)**. Stewart at ¶[0068] In some embodiments, a portion of the application may be installed on the management server. For example, **the application may include a first portion (provided to the gateway device) to adequately perform management of the fuel tank device, and the application may include a second portion for user interaction. For example, the second portion of the application may be executed by the server to provide the received reported information to various other devices for presentation, provide a user interface (e.g., a graphical user interface or "GUI") to control or modify settings associated with the fuel tank device, provide a user interface to set up fuel level thresholds (for when the fuel tank should be refilled), set up contact information associated with the fuel tank (e.g., whom to contact when the fuel level is below a threshold), purchase order information for requisitioning more fuel when the fuel tank is depleted, etc.** The second portion may be a pluggable module into a management application (e.g., within a website) provided by the server. In one embodiment, the management server may include a database of available interfaces (e.g., associated with each application that is configured for execution on the management server) and, in response to an application being installed, associate the interface (or application portion) with the gateway device or user for which the application was installed. Thus, the application portion may already be present on the management server and may simply be activated upon installation (e.g., by associating a user or location with the application in a database). Thus, the second portion may "add on" to a management interface for the convenience store provided by the server.

Claims of '823 Patent	**Prior Art Disclosure**
	Stewart at ¶[0072]
Claim 19 [P]	
A system comprising:	To the extent that the preamble is limiting, Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *a system*. *See* Claim 1[P], *supra.*
Claim 19 [a]	
a programmable network device adapted to host a plurality of network device applications;	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *a programmable* (e.g., configurable) *network device* (e.g., gateway device) *adapted to host a plurality of network device applications* (e.g., applications or application programs on the gateway device). *See* Claim 1[a], *supra.*
Claim 19 [b]	
a programmable cloud device adapted to host a plurality of cloud applications,	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses *a programmable* (e.g., configurable) *cloud device* (e.g., cloud server) *adapted to host a plurality of cloud applications* (e.g., applications or application programs on the cloud server). *See* Claim 1[b], *supra.*
Claim 19[c]	
wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications;	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Stewart discloses that *the plurality of network device applications* (e.g., applications or application programs on the gateway device) *and the plurality of cloud applications* (e.g., applications or application programs on the cloud server) *are in secure communication with each other to form distributed applications*. *See* Claim 1[c], *supra.*
Claim 19 [d]	
wherein the plurality of network device	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation.

Claims of '823 Patent	**Prior Art Disclosure**
applications and plurality of cloud applications form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware;	Requester notes that the only difference between Claim 1[d], *supra*, and Claim 19[d] is that Claim 1[d], *supra*, includes the word "device" after the first recitation of "cloud applications", which appears to be a typographical error. Stewart discloses that *the plurality of network device applications* (e.g., applications or application programs on the gateway device) *and plurality of cloud applications* (e.g., applications or application programs on the cloud server) *form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device* (e.g., gateway device) *hardware and cloud device* (e.g., clud server) *hardware. See* Claim 1[d], *supra.*
Claim 19 [e]	
an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications; and	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation. Requester notes that the only difference between Claim 3[a], *supra*, and Claim 19[e] is that Claim 3[a], *supra*, recites "distributed applications" while 19[e] recites "the plurality of network device applications and plurality of cloud applications." However, because "the plurality of network device applications and plurality of cloud applications" are what forms the "distributed applications", the same disclosures in Stewart are equally applicable to both limitations. Geier Decl. at ¶ 76. Stewart discloses *an application management portal* (e.g., a management server) *capable of managing life cycles of the plurality of network device applications* (e.g., applications or application programs on the gateway device) *and plurality of cloud applications* (e.g., applications or application programs on the cloud server). *See* Claim 3[a], *supra.*
Claim 19 [f]	
an infrastructure application marketplace in communication	Stewart anticipates (Ground 1) or at least renders obvious (Ground 2) this limitation.

Claims of '823 Patent	**Prior Art Disclosure**
with the application management portal, said infrastructure marketplace capable of providing tested distributed applications to the application management portal.	Stewart discloses *an infrastructure application marketplace* (e.g., application server operating as an application store) *in communication with the application management portal* (e.g., management server)*, said infrastructure marketplace capable of providing tested distributed applications to the application management portal*. *See* Claim 3[b], *supra.* Applicant notes that claim 3 recites providing "tested and certified" distributed applications, while claim 19 recites providing "tested" distributed applications. However, the difference does not provide any distinction, and the same disclosure in Stewart is applicable to both terms. Geier Decl. at ¶ 82.

US005691897A

United States Patent [19]

Brown et al.

[11] Patent Number: 5,691,897

[45] Date of Patent: Nov. 25, 1997

[54] **MOTION CONTROL SYSTEMS**

[75] Inventors: **David W. Brown**, White Salmon; **Jay S. Clark**, Seattle, both of Wash.

[73] Assignee: **Roy-G-Biv Corporation**, White Salmon, Wash.

[21] Appl. No.: **454,736**

[22] Filed: **May 30, 1995**

[51] **Int. Cl.**[6] **G05B 19/18**; G05B 19/42

[52] **U.S. Cl.** **364/167.01**; 364/191; 318/568.1

[58] **Field of Search** 364/167.01, 474.21, 364/474.22, 474.23, 474.24, 474.25, 474.26, 191, 138; 395/500; 318/567, 568.1

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,937,737	6/1990	Schwane et al. .	
5,095,445	3/1992	Sekiguchi .	
5,126,932	6/1992	Wolfson et al. .	
5,175,817	12/1992	Adams et al. .	
5,245,703	9/1993	Hubert .	
5,247,650	9/1993	Judd et al. .	
5,453,933	9/1995	Wright et al.	364/474.23
5,491,813	2/1996	Bondy et al.	395/500

OTHER PUBLICATIONS

WOSA Backgrounder: Delivering Enterprise Services to the Windows-based Desktop, Jul. 1993, Microsoft Development Library.
WOSA Inside Windows 95, pp. 348-351.
Software Products For Industrial Automation, ICONICS.
The Complete, Computer-Based Automation Tool (IGSS), Seven Technologies A/S.
OpenBatch Product Brief, PID, Inc.
FIX Product Brochure, Intellution (1994).
AIMAX-WIN Product Brochure, TA Engineering Co., Inc.
XANALOG Model NL-SIM Product Borchure, XANALOG.
ExperTune PID Tuning Software, Gerry Engineering Software.
Paragon TNT Product Brochure, Intec Controls Corp.
Compumotor Digiplan 1993-94 catalog, pp. 10-11.
Aerotech Motion Control Product Guide, pp. 233-234.
PMAC Product Catalog, p. 43.
PC/DSP-Series Motion Controller C Programming Guide.
Precision Motion Controls Product Guide, p. 17.
WEB 3.0 Product Brochure, Trihedral Engineering Ltd. (1994).
Oregon Micro Systems Product Guide.

Primary Examiner—Paul P. Gordon
Attorney, Agent, or Firm—Michael R. Schacht; Hughes, Multer & Schacht, P.S.

[57] **ABSTRACT**

A system for motion control in which an application is developed that is independent from the actual motion control hardware used to implement the system. The system comprises a software system that employs an application programming interface comprising component functions and a service provider interface comprising driver functions. A system programmer writes an application that calls the component functions. Code associated with the component functions relates these functions to the driver functions. A hardware designer writes driver code that implements the driver functions on a given motion control hardware product. The driver functions are separated into core and extended driver functions. All software drivers implement the core driver functions, while the software drivers need not contain code for implementing the extended driver functions. If the software driver does not contain code to implement an extended driver function, the functionality of the extended driver function is obtained through a combination of core driver functions. The system programmer may also select one or more streams that allow the control commands to be communicated to, and response data to be communicated from, motion control hardware.

25 Claims, 58 Drawing Sheets

FIG. 1A

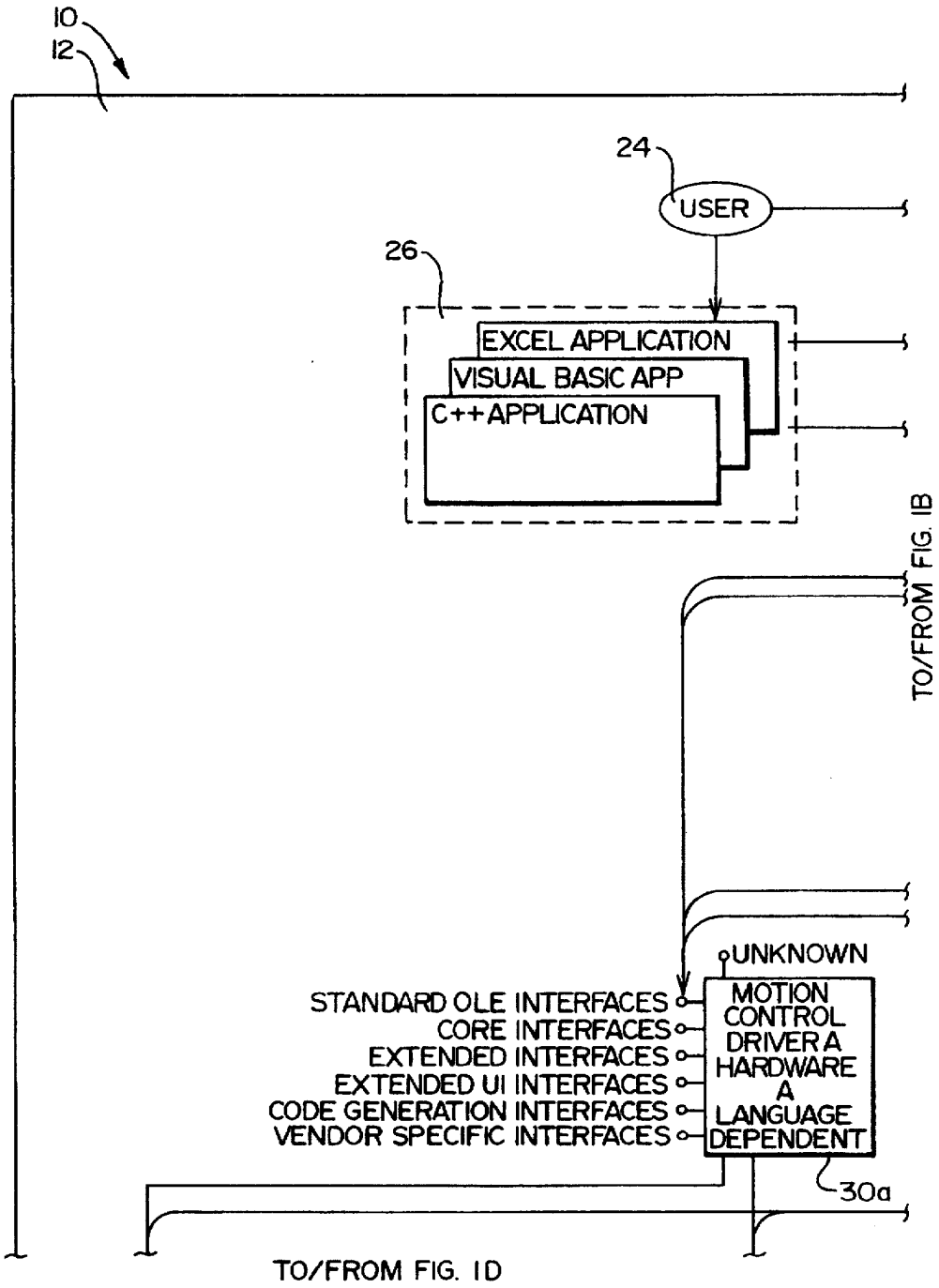

10
12
24
USER
26
EXCEL APPLICATION
VISUAL BASIC APP
C++APPLICATION
TO/FROM FIG. 1B
UNKNOWN
STANDARD OLE INTERFACES
CORE INTERFACES
EXTENDED INTERFACES
EXTENDED UI INTERFACES
CODE GENERATION INTERFACES
VENDOR SPECIFIC INTERFACES
MOTION CONTROL DRIVER A
HARDWARE A
LANGUAGE DEPENDENT
30a
TO/FROM FIG. 1D

FIG. 1B

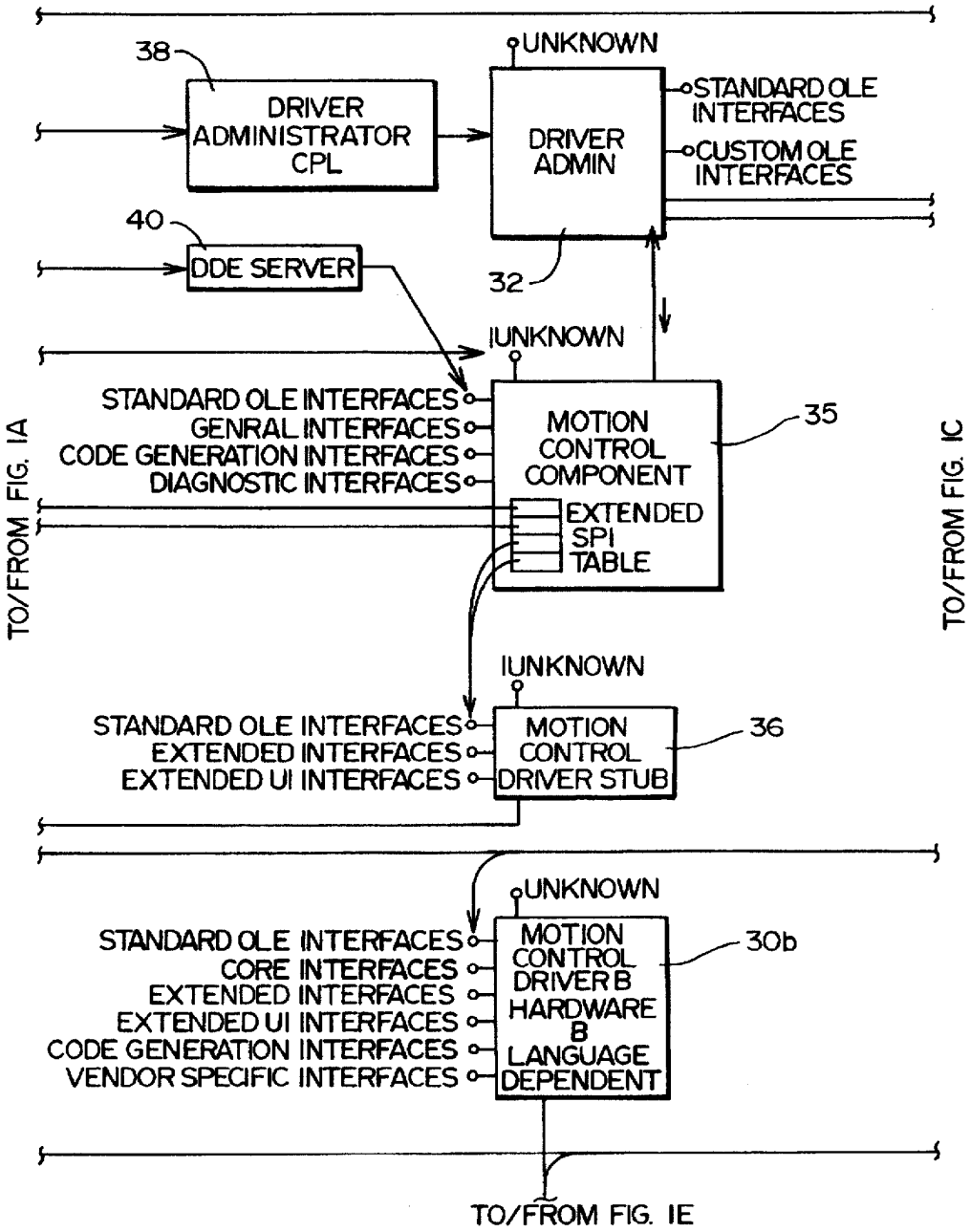

38
DRIVER ADMINISTRATOR CPL
UNKNOWN
DRIVER ADMIN
STANDARD OLE INTERFACES
CUSTOM OLE INTERFACES
40
DDE SERVER
32
IUNKNOWN
STANDARD OLE INTERFACES
GENRAL INTERFACES
CODE GENERATION INTERFACES
DIAGNOSTIC INTERFACES
MOTION CONTROL COMPONENT
35
EXTENDED SPI TABLE
TO/FROM FIG. IA
TO/FROM FIG. IC
IUNKNOWN
STANDARD OLE INTERFACES
EXTENDED INTERFACES
EXTENDED UI INTERFACES
MOTION CONTROL DRIVER STUB
36
UNKNOWN
STANDARD OLE INTERFACES
CORE INTERFACES
EXTENDED INTERFACES
EXTENDED UI INTERFACES
CODE GENERATION INTERFACES
VENDOR SPECIFIC INTERFACES
MOTION CONTROL DRIVER B HARDWARE B LANGUAGE DEPENDENT
30b
TO/FROM FIG. IE

FIG. IC

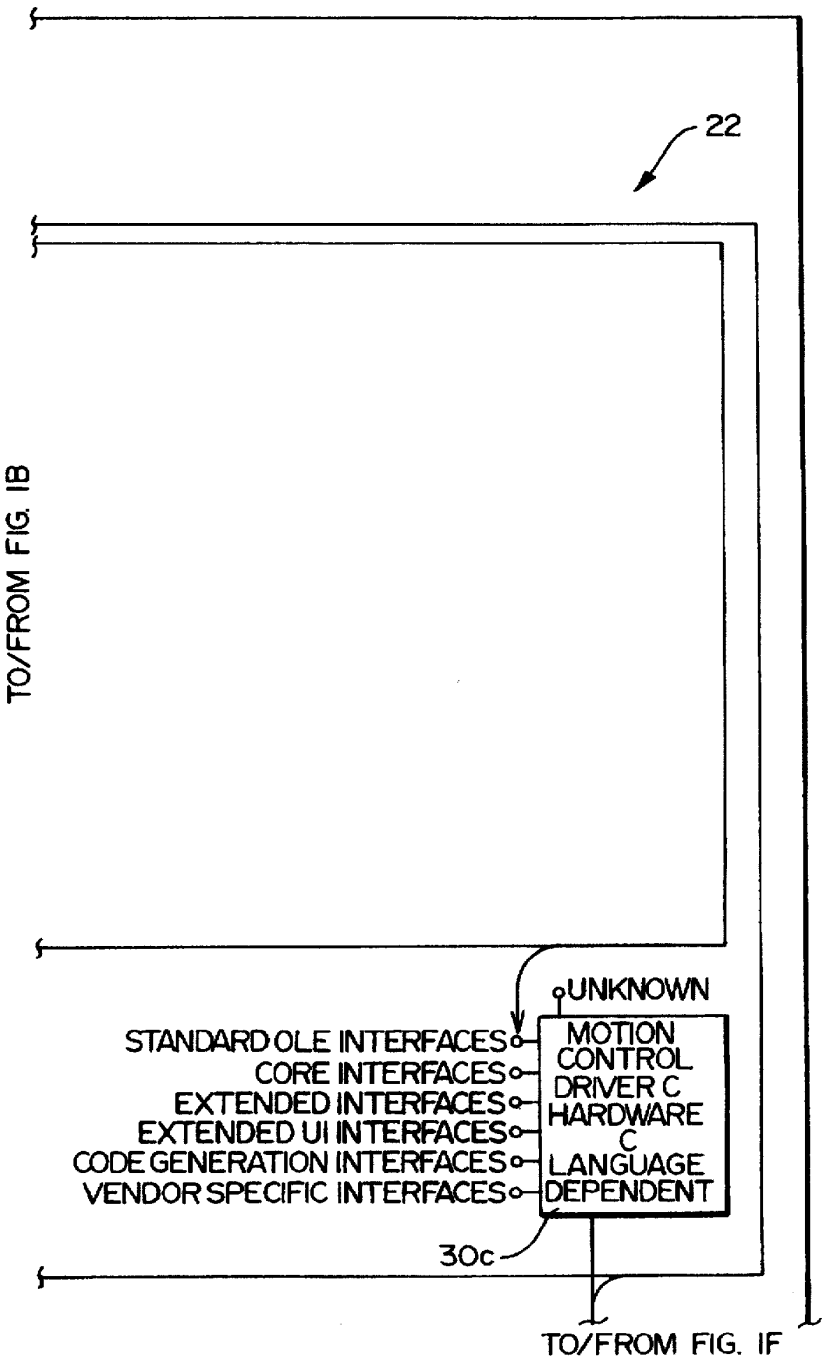

22
TO/FROM FIG. IB
UNKNOWN
STANDARD OLE INTERFACES
CORE INTERFACES
EXTENDED INTERFACES
EXTENDED UI INTERFACES
CODE GENERATION INTERFACES
VENDOR SPECIFIC INTERFACES
MOTION CONTROL DRIVER C HARDWARE C LANGUAGE DEPENDENT
30c
TO/FROM FIG. IF

FIG. ID

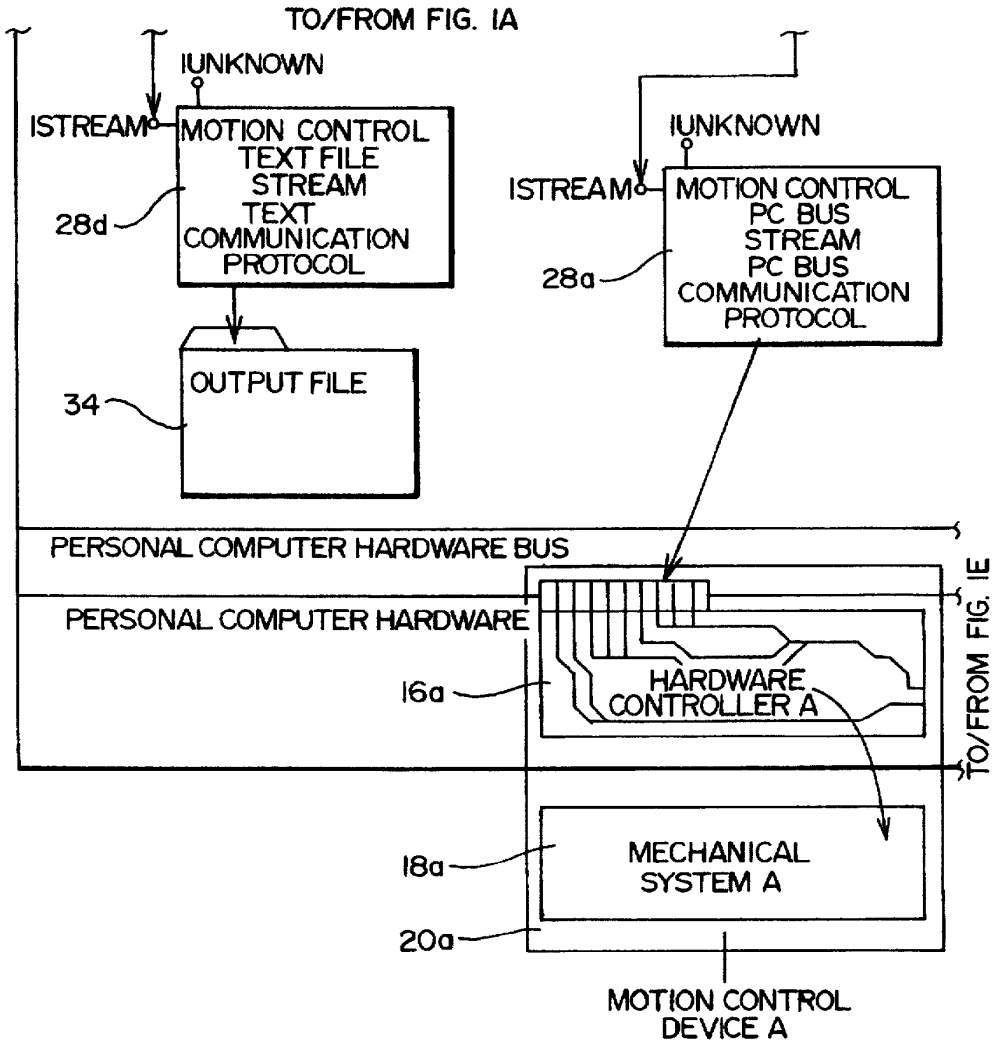

TO/FROM FIG. IA
IUNKNOWN
ISTREAM
MOTION CONTROL TEXT FILE STREAM TEXT COMMUNICATION PROTOCOL
28d
OUTPUT FILE
34
IUNKNOWN
ISTREAM
MOTION CONTROL PC BUS STREAM PC BUS COMMUNICATION PROTOCOL
28a
PERSONAL COMPUTER HARDWARE BUS
PERSONAL COMPUTER HARDWARE
16a
HARDWARE CONTROLLER A
TO/FROM FIG. IE
18a
MECHANICAL SYSTEM A
20a
MOTION CONTROL DEVICE A

FIG. 1E

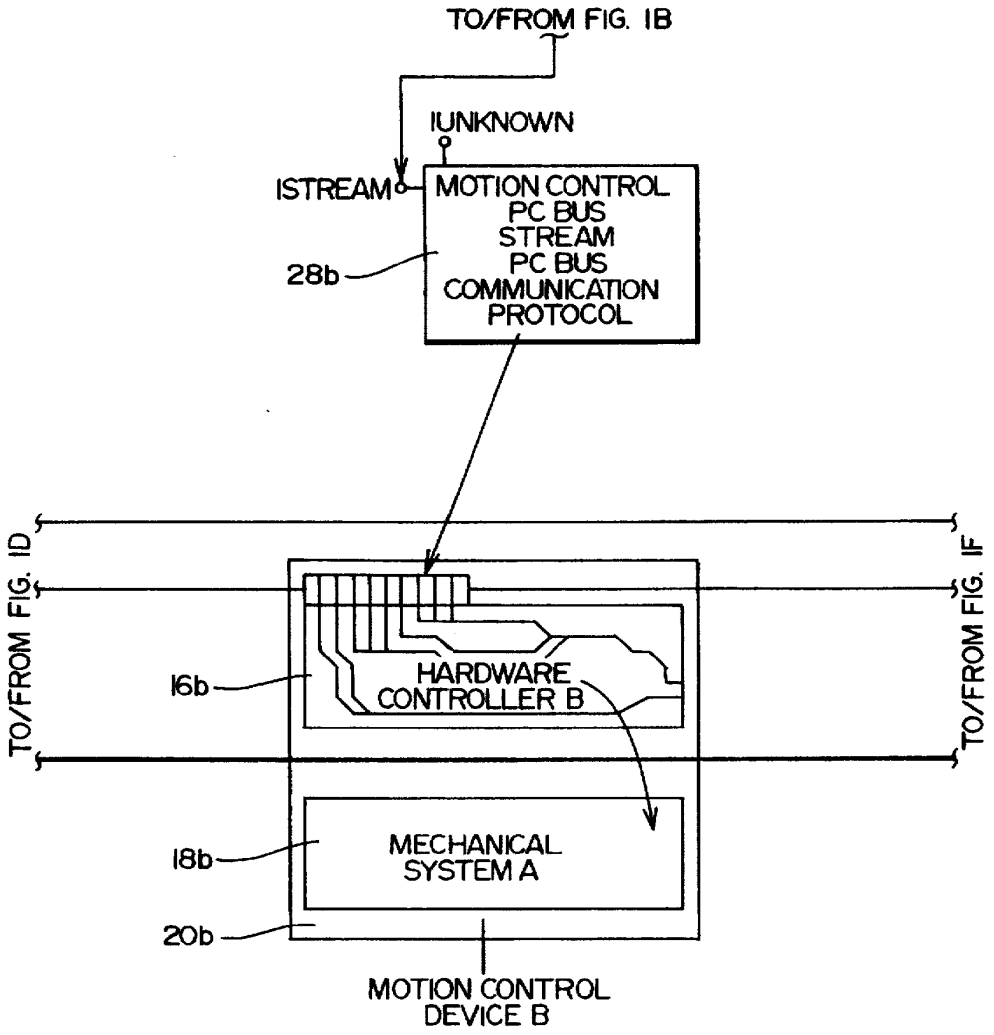

TO/FROM FIG. 1B
IUNKNOWN
ISTREAM
MOTION CONTROL PC BUS STREAM PC BUS COMMUNICATION PROTOCOL
28b
TO/FROM FIG. 1D
TO/FROM FIG. 1F
16b
HARDWARE CONTROLLER B
18b
MECHANICAL SYSTEM A
20b
MOTION CONTROL DEVICE B

FIG. IF

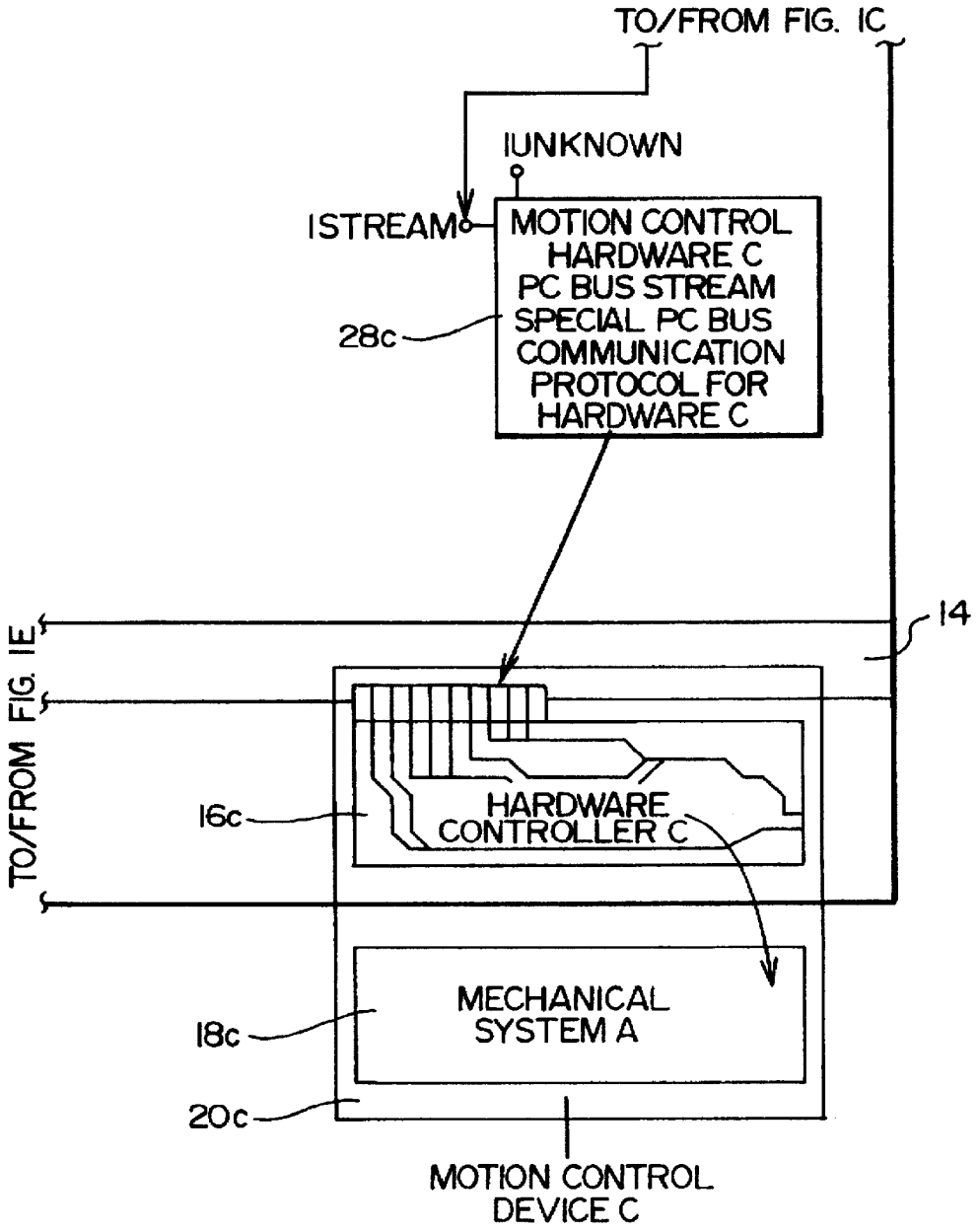

TO/FROM FIG. IC
IUNKNOWN
ISTREAM
MOTION CONTROL HARDWARE C PC BUS STREAM SPECIAL PC BUS COMMUNICATION PROTOCOL FOR HARDWARE C
28c
14
TO/FROM FIG. IE
16c
HARDWARE CONTROLLER C
MECHANICAL SYSTEM A
18c
20c
MOTION CONTROL DEVICE C

FIG. 2

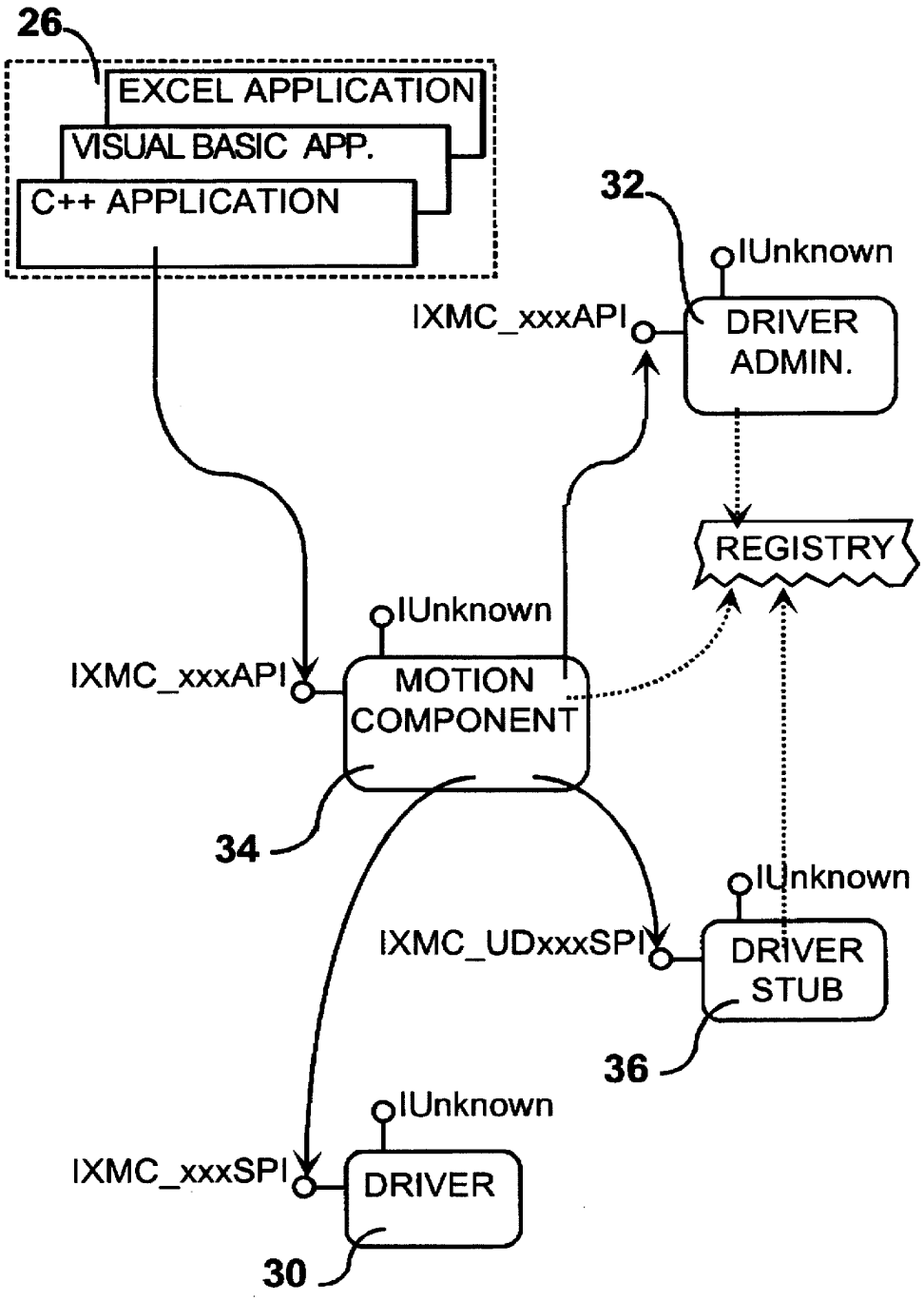

26
EXCEL APPLICATION
VISUAL BASIC APP.
C++ APPLICATION
32
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
REGISTRY
IUnknown
IXMC_xxxAPI
MOTION COMPONENT
34
IUnknown
IXMC_UDxxxSPI
DRIVER STUB
36
IUnknown
IXMC_xxxSPI
DRIVER
30

FIG. 3

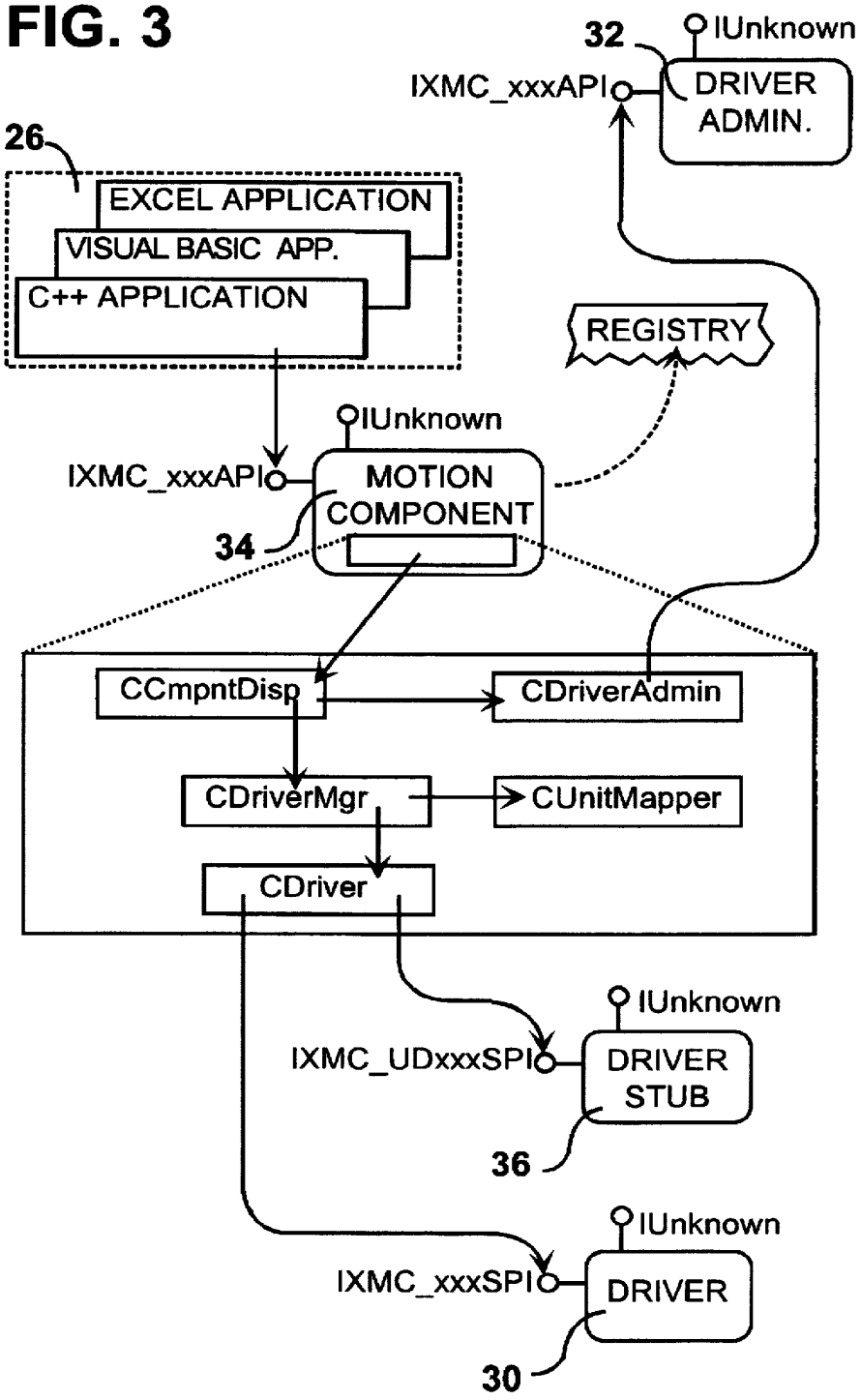

32
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
26
EXCEL APPLICATION
VISUAL BASIC APP.
C++ APPLICATION
REGISTRY
IUnknown
IXMC_xxxAPI
MOTION COMPONENT
34
CCmpntDisp
CDriverAdmin
CDriverMgr
CUnitMapper
CDriver
IUnknown
IXMC_UDxxxSPI
DRIVER STUB
36
IUnknown
IXMC_xxxSPI
DRIVER
30

FIG. 4

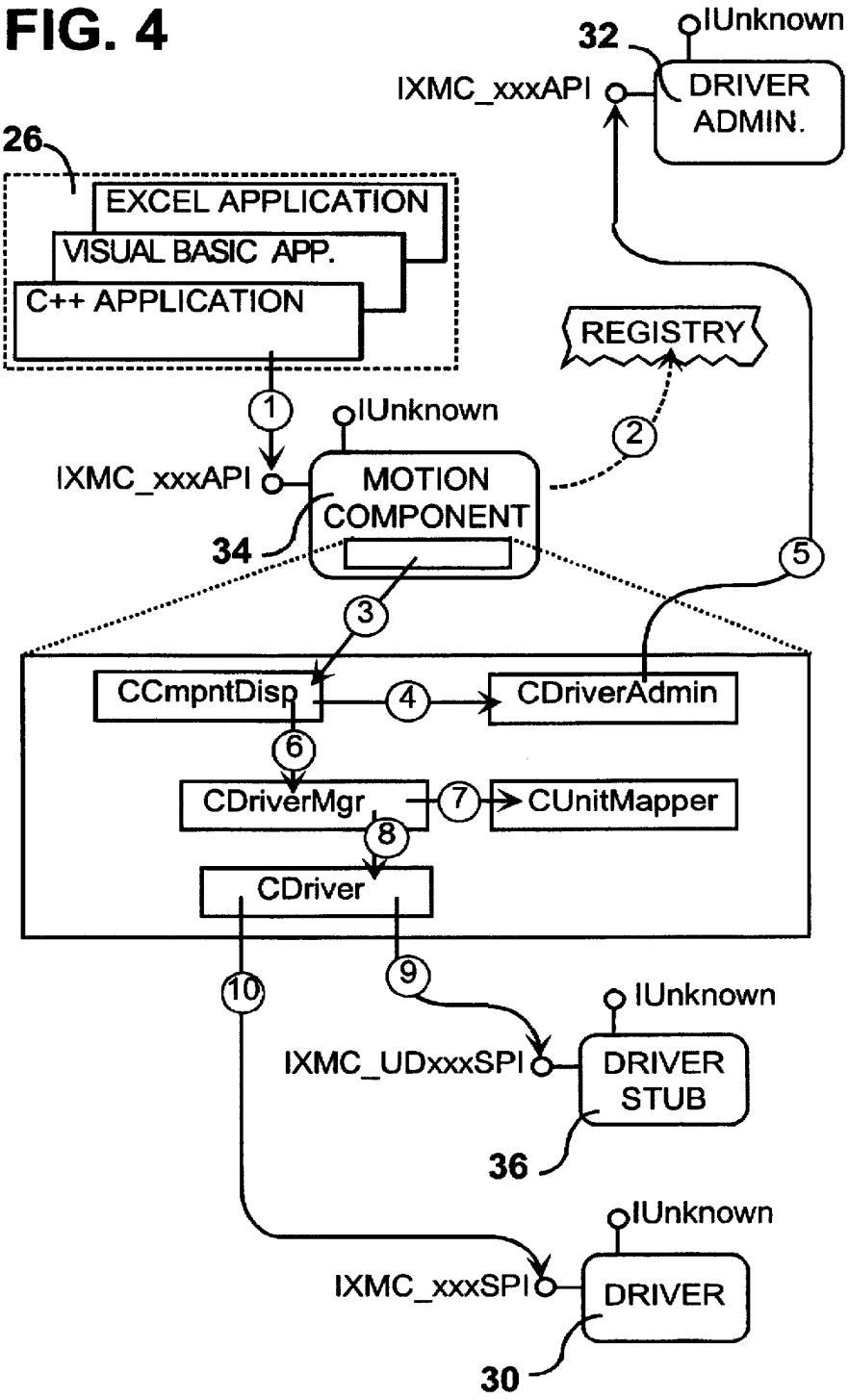

32
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
26
EXCEL APPLICATION
VISUAL BASIC APP.
C++ APPLICATION
REGISTRY
1
IUnknown
2
IXMC_xxxAPI
MOTION COMPONENT
34
5
3
CCmpntDisp
4
CDriverAdmin
6
CDriverMgr
7
CUnitMapper
8
CDriver
10
9
IUnknown
IXMC_UDxxxSPI
DRIVER STUB
36
IUnknown
IXMC_xxxSPI
DRIVER
30

FIG. 5

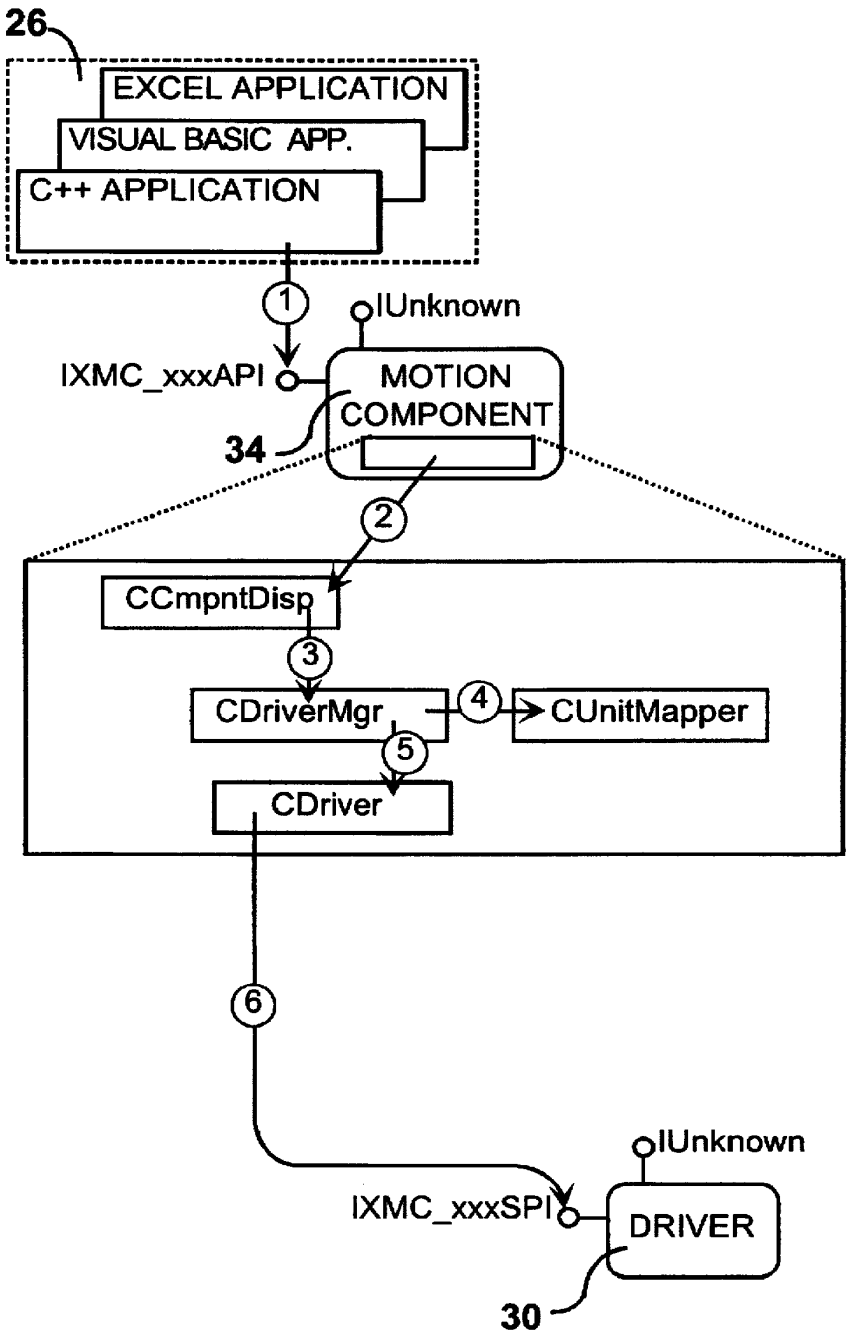

26
EXCEL APPLICATION
VISUAL BASIC APP.
C++ APPLICATION
1
IUnknown
IXMC_xxxAPI
MOTION COMPONENT
34
2
CCmpntDisp
3
CDriverMgr
4
CUnitMapper
5
CDriver
6
IUnknown
IXMC_xxxSPI
DRIVER
30

FIG. 6

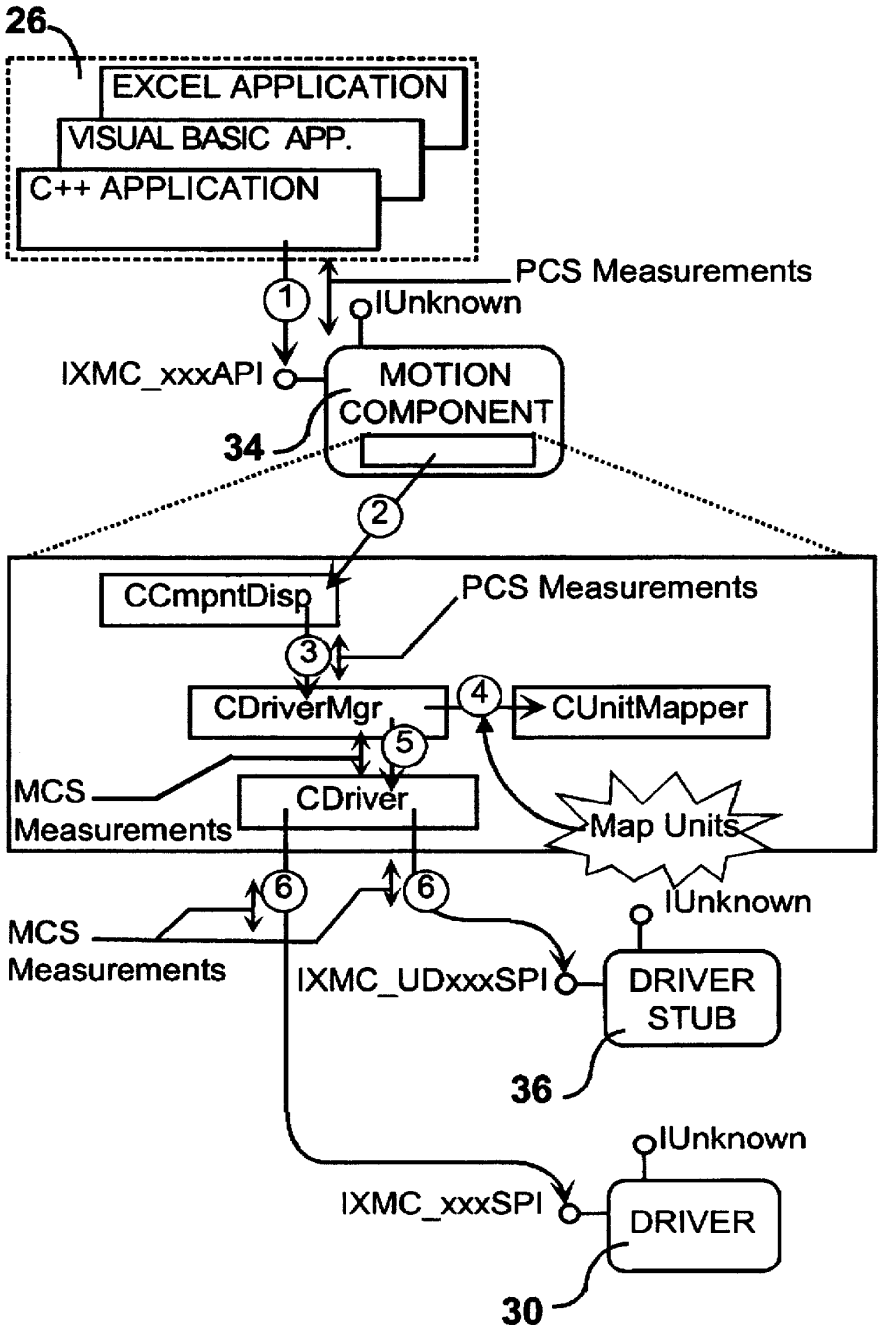

26
EXCEL APPLICATION
VISUAL BASIC APP.
C++ APPLICATION
PCS Measurements
1
IUnknown
IXMC_xxxAPI
MOTION COMPONENT
34
2
CCmpntDisp
PCS Measurements
3
CDriverMgr
4
CUnitMapper
5
MCS Measurements
CDriver
Map Units
6
6
IUnknown
MCS Measurements
IXMC_UDxxxSPI
DRIVER STUB
36
IUnknown
IXMC_xxxSPI
DRIVER
30

FIG. 7

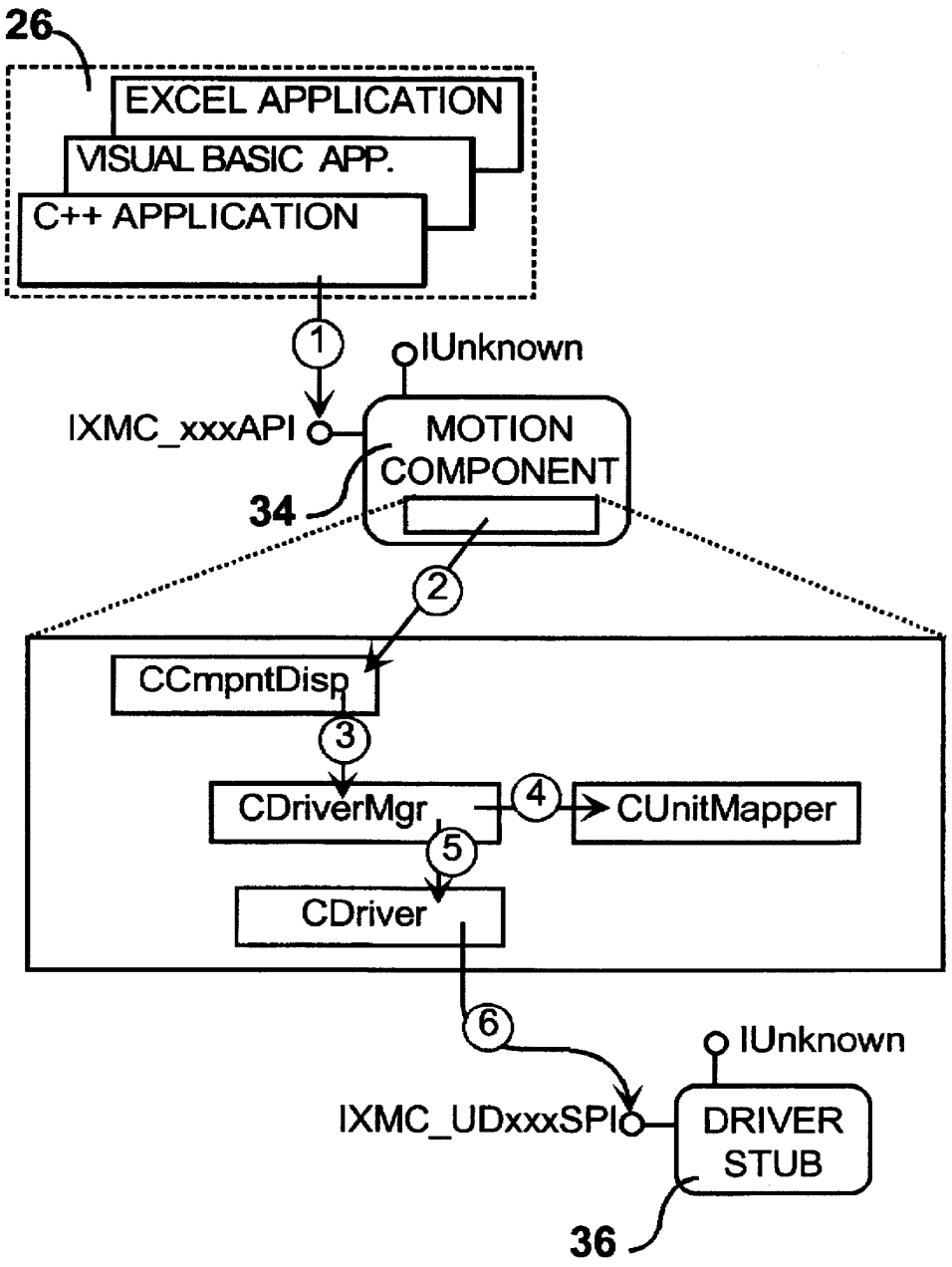

26
EXCEL APPLICATION
VISUAL BASIC APP.
C++ APPLICATION
1
IUnknown
IXMC_xxxAPI
MOTION COMPONENT
34
2
CCmpntDisp
3
CDriverMgr
4
CUnitMapper
5
CDriver
6
IUnknown
IXMC_UDxxxSPI
DRIVER STUB
36

FIG. 8

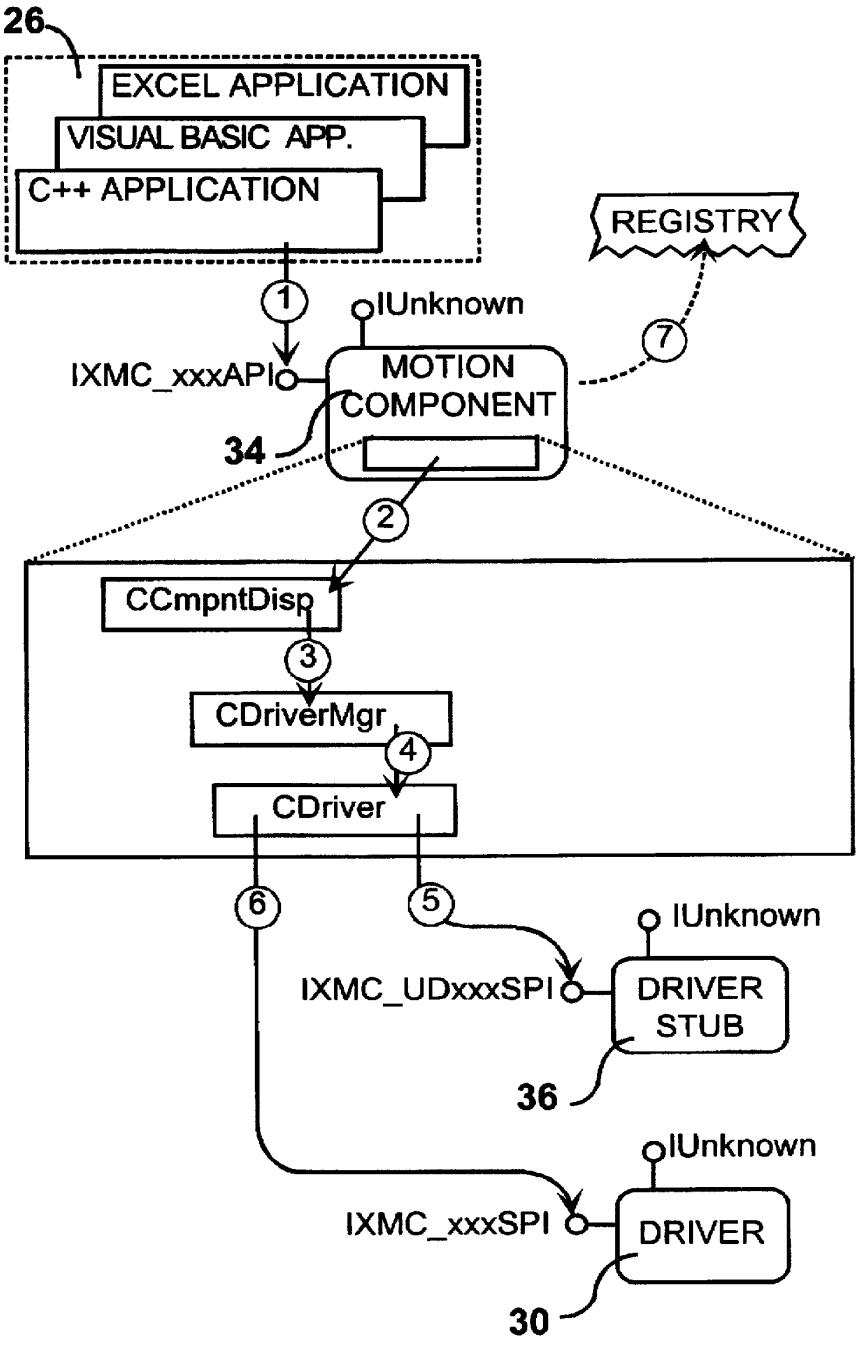

26
EXCEL APPLICATION
VISUAL BASIC APP.
C++ APPLICATION
REGISTRY
1
IUnknown
IXMC_xxxAPI
MOTION COMPONENT
7
34
2
CCmpntDisp
3
CDriverMgr
4
CDriver
6
5
IUnknown
IXMC_UDxxxSPI
DRIVER STUB
36
IUnknown
IXMC_xxxSPI
DRIVER
30

FIG. 9

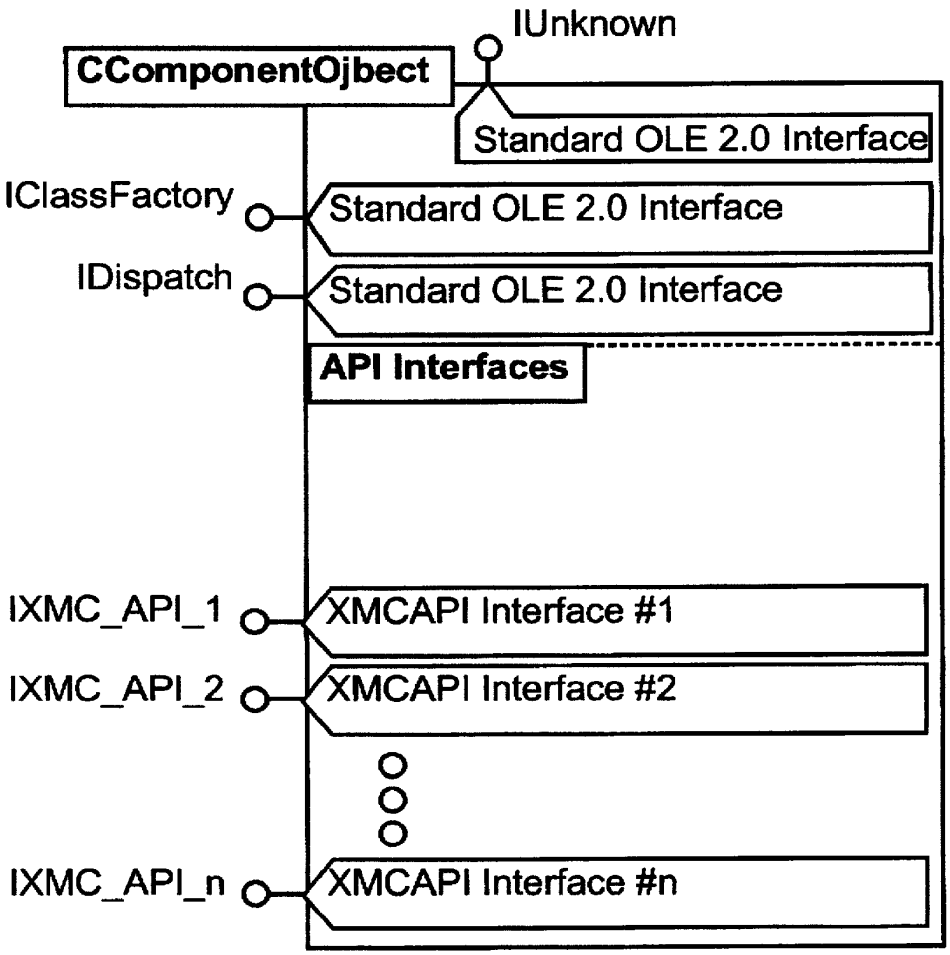

IUnknown
CComponentOjbect
Standard OLE 2.0 Interface
IClassFactory
Standard OLE 2.0 Interface
IDispatch
Standard OLE 2.0 Interface
API Interfaces
IXMC_API_1
XMCAPI Interface #1
IXMC_API_2
XMCAPI Interface #2
IXMC_API_n
XMCAPI Interface #n

FIG. 10

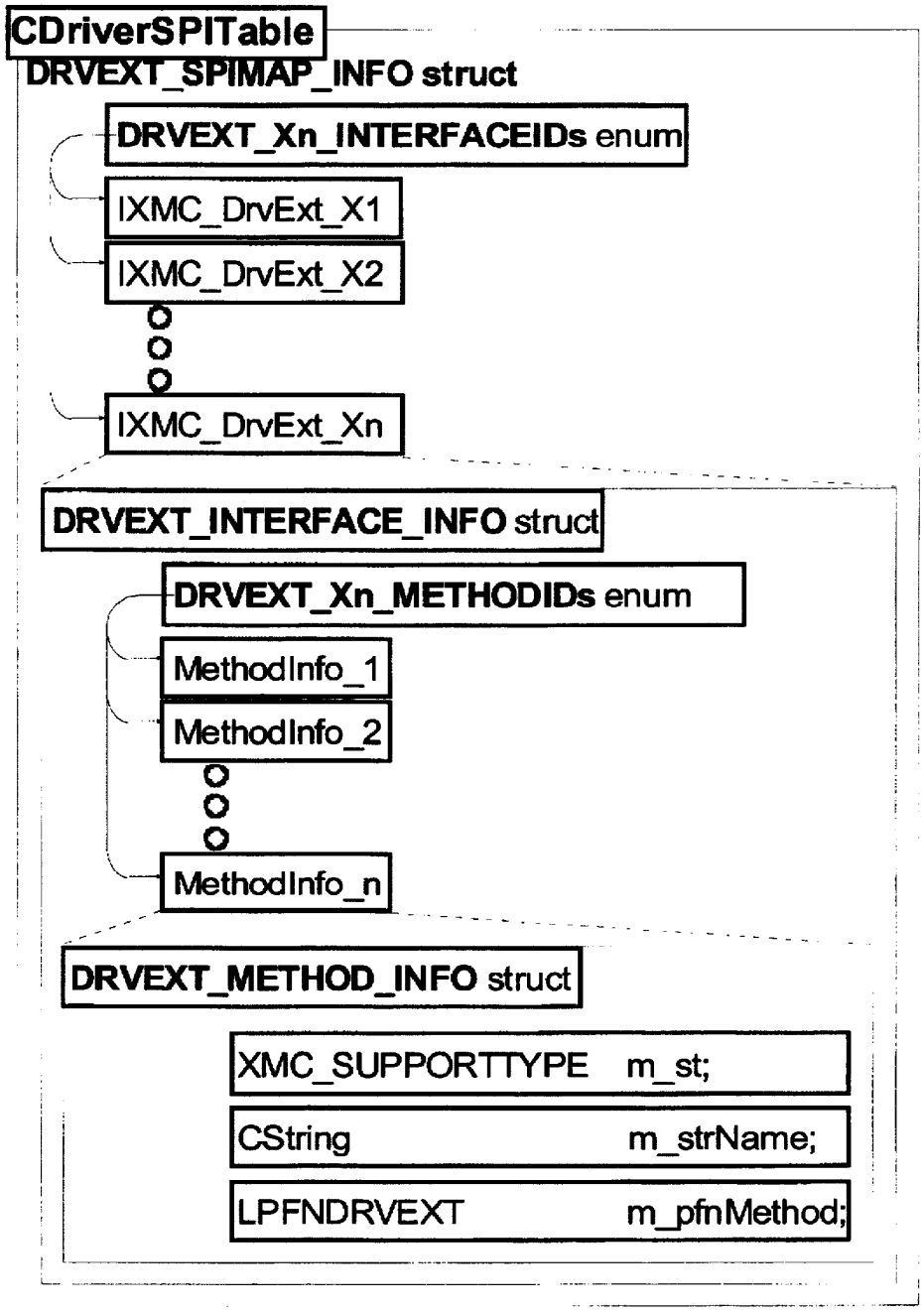

CDriverSPITable
DRVEXT_SPIMAP_INFO struct
DRVEXT_Xn_INTERFACEIDs enum
IXMC_DrvExt_X1
IXMC_DrvExt_X2
IXMC_DrvExt_Xn
DRVEXT_INTERFACE_INFO struct
DRVEXT_Xn_METHODIDs enum
MethodInfo_1
MethodInfo_2
MethodInfo_n
DRVEXT_METHOD_INFO struct
XMC_SUPPORTTYPE m_st;
CString m_strName;
LPFNDRVEXT m_pfnMethod;

FIG. 11

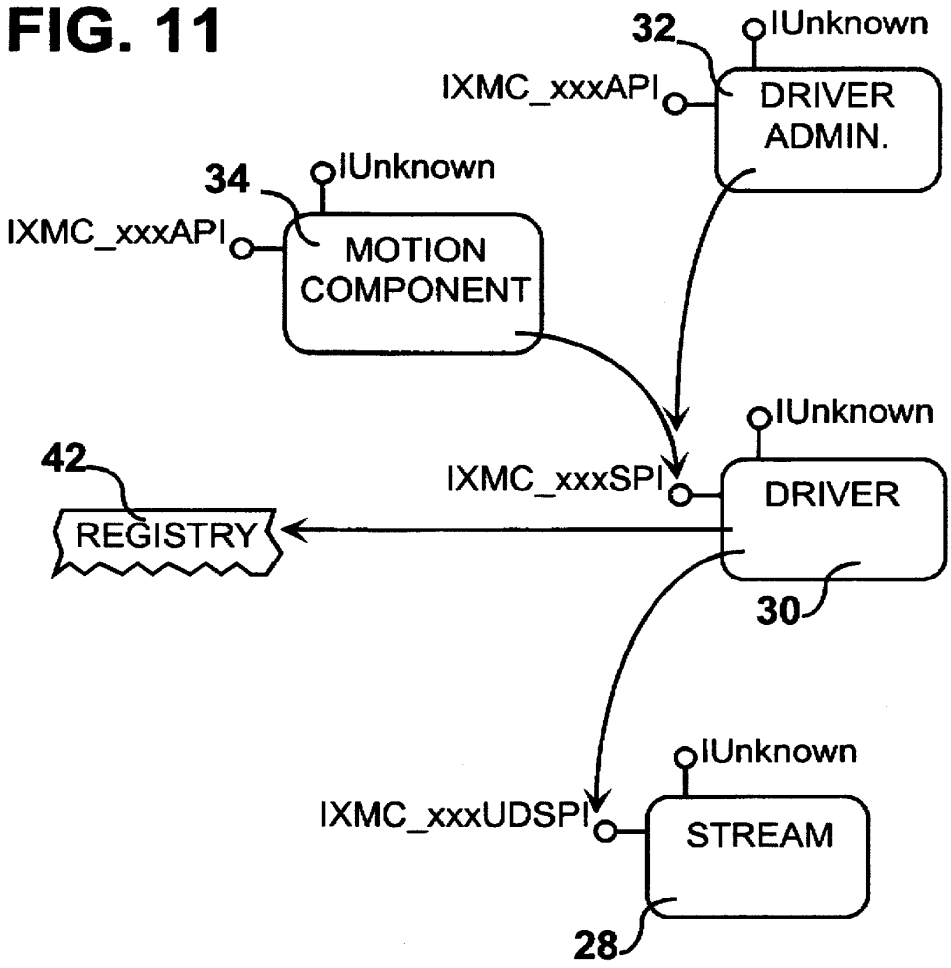

32
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
34
IUnknown
IXMC_xxxAPI
MOTION COMPONENT
IUnknown
42
IXMC_xxxSPI
DRIVER
REGISTRY
30
IUnknown
IXMC_xxxUDSPI
STREAM
28

FIG. 12

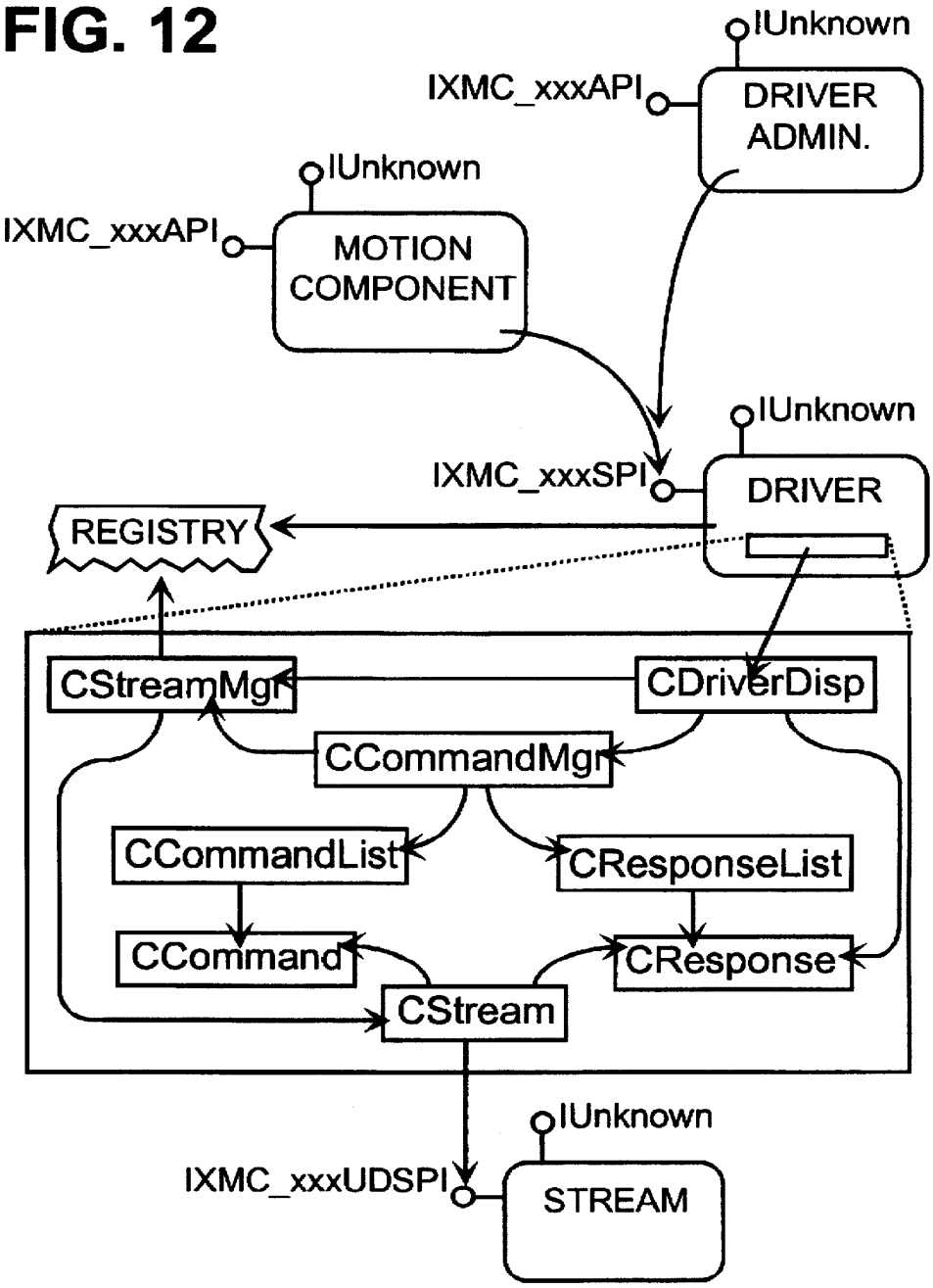

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
IUnknown
IXMC_xxxAPI
MOTION COMPONENT
IUnknown
IXMC_xxxSPI
DRIVER
REGISTRY
CStreamMgr
CDriverDisp
CCommandMgr
CCommandList
CResponseList
CCommand
CResponse
CStream
IUnknown
IXMC_xxxUDSPI
STREAM

FIG. 13

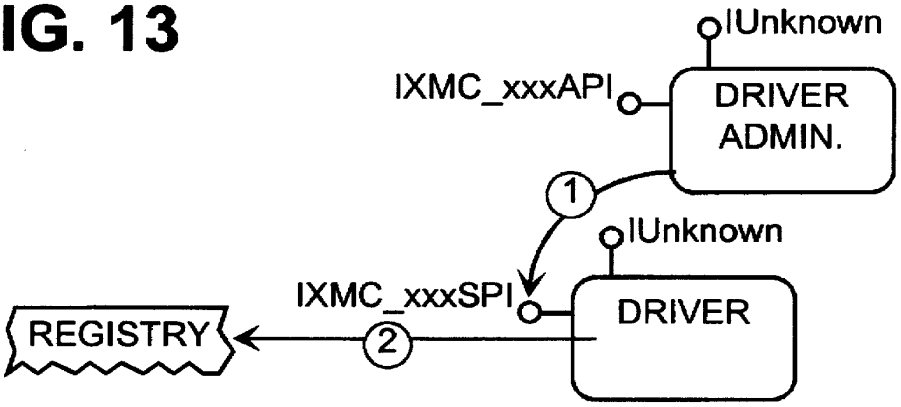

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
1
IUnknown
IXMC_xxxSPI
DRIVER
REGISTRY
2

FIG. 14

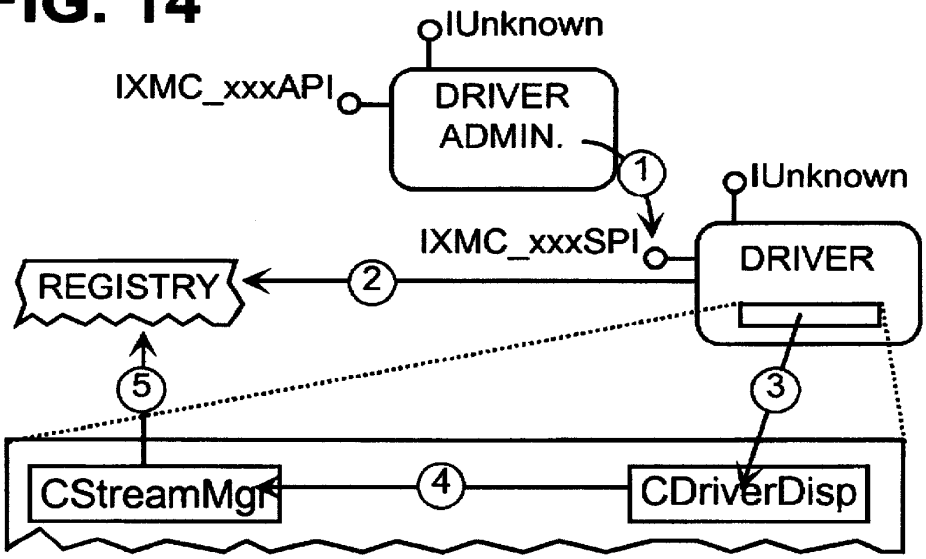

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
1
IUnknown
IXMC_xxxSPI
DRIVER
REGISTRY
2
5
3
CStreamMgr
4
CDriverDisp

FIG. 15

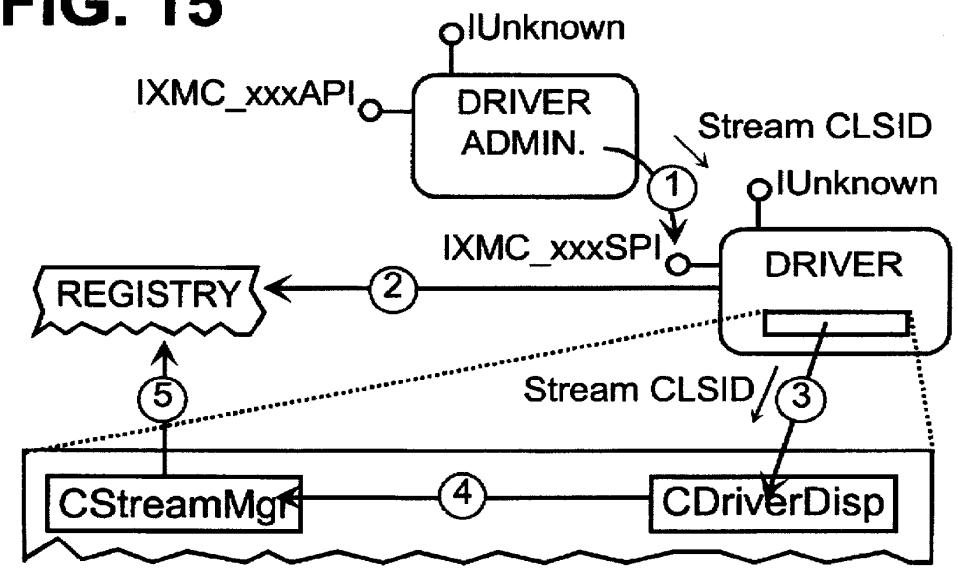

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
Stream CLSID
1
IUnknown
IXMC_xxxSPI
DRIVER
REGISTRY
2
5
Stream CLSID
3
CStreamMgr
4
CDriverDisp

FIG. 16

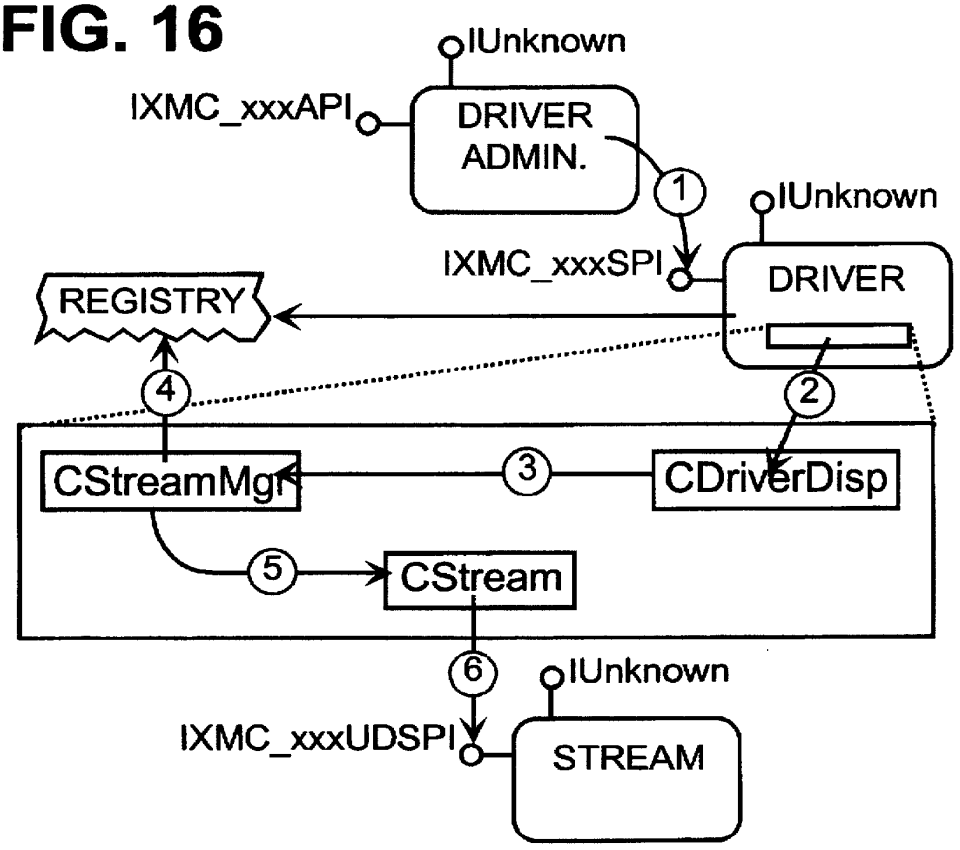

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
1
IUnknown
IXMC_xxxSPI
DRIVER
REGISTRY
4
2
CStreamMgr
3
CDriverDisp
5
CStream
6
IUnknown
IXMC_xxxUDSPI
STREAM

FIG. 17

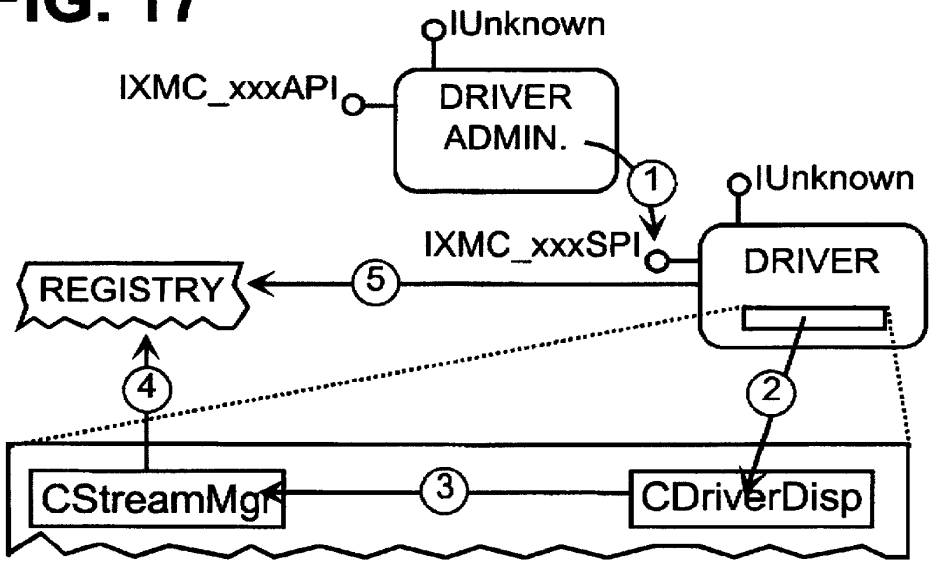

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
1
IUnknown
IXMC_xxxSPI
DRIVER
REGISTRY
5
4
2
CStreamMgr
3
CDriverDisp

FIG. 18

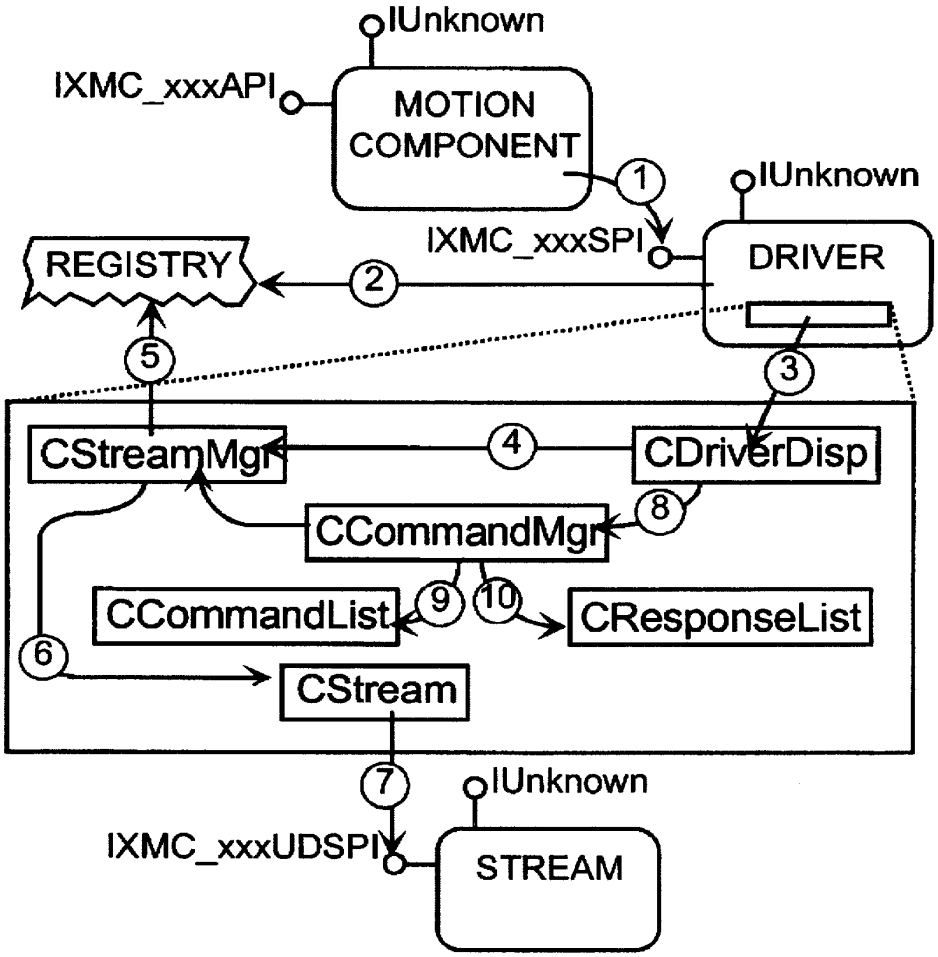

IUnknown
IXMC_xxxAPI
MOTION COMPONENT
1
IUnknown
IXMC_xxxSPI
DRIVER
REGISTRY
2
5
3
CStreamMgr
4
CDriverDisp
8
CCommandMgr
CCommandList
9
10
CResponseList
6
CStream
7
IUnknown
IXMC_xxxUDSPI
STREAM

FIG. 19

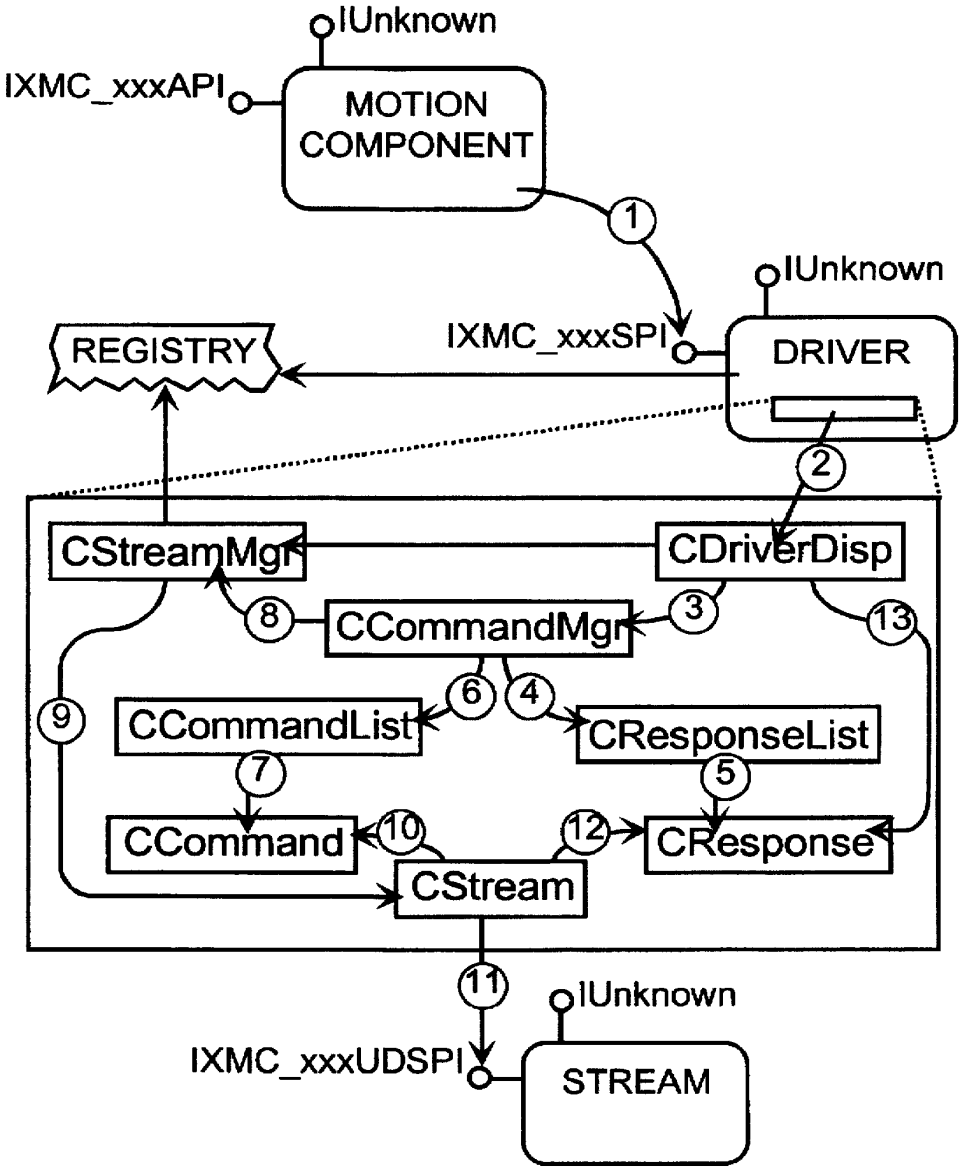

IUnknown
IXMC_xxxAPI
MOTION COMPONENT
1
IUnknown
REGISTRY
IXMC_xxxSPI
DRIVER
2
CStreamMgr
CDriverDisp
8
CCommandMgr
3
13
9
6
4
CCommandList
CResponseList
7
5
CCommand
10
12
CResponse
CStream
11
IUnknown
IXMC_xxxUDSPI
STREAM

FIG. 20

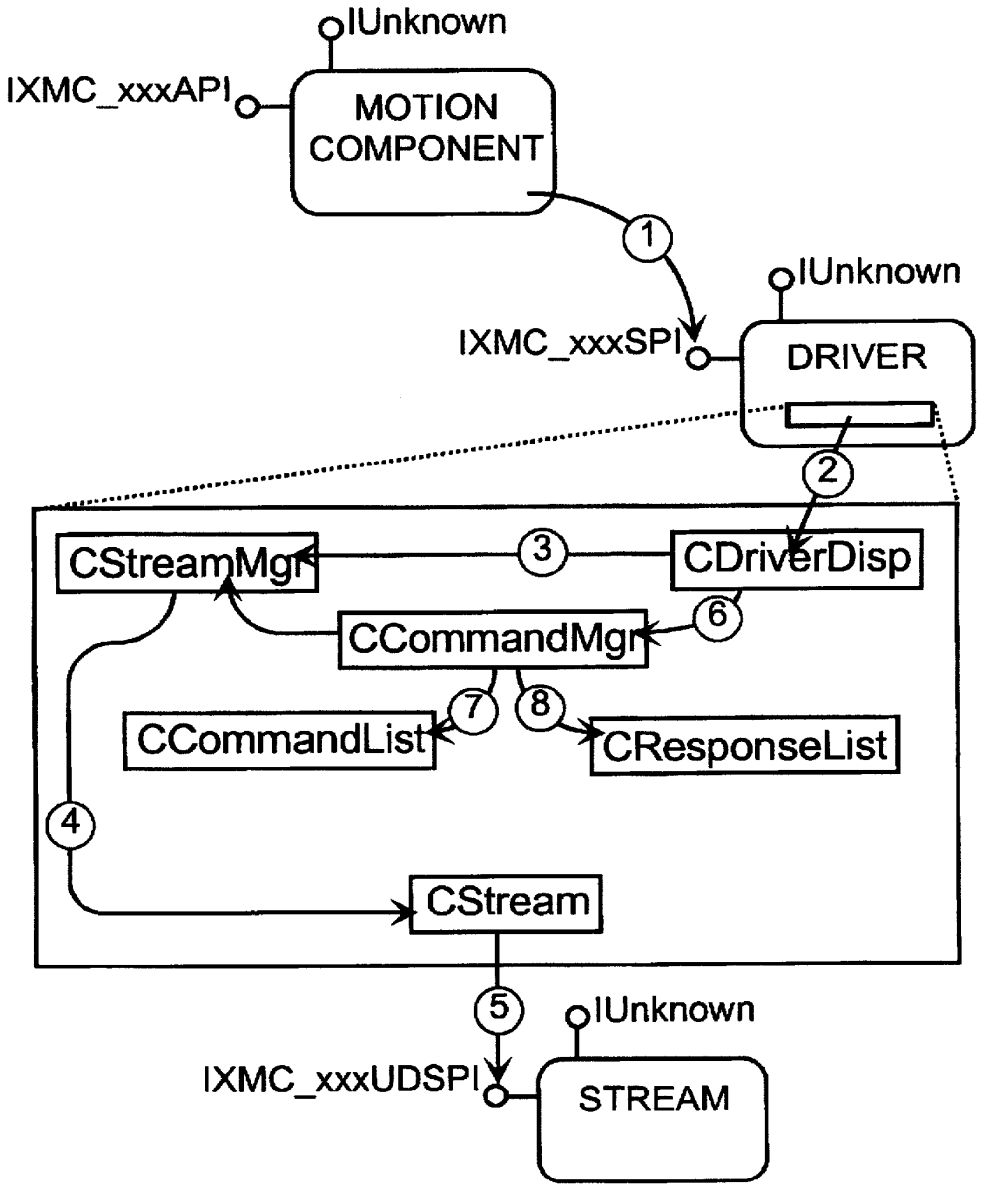

IUnknown
IXMC_xxxAPI
MOTION COMPONENT
1
IUnknown
IXMC_xxxSPI
DRIVER
2
CStreamMgr
3
CDriverDisp
6
CCommandMgr
7
8
CCommandList
CResponseList
4
CStream
5
IUnknown
IXMC_xxxUDSPI
STREAM

FIG. 21

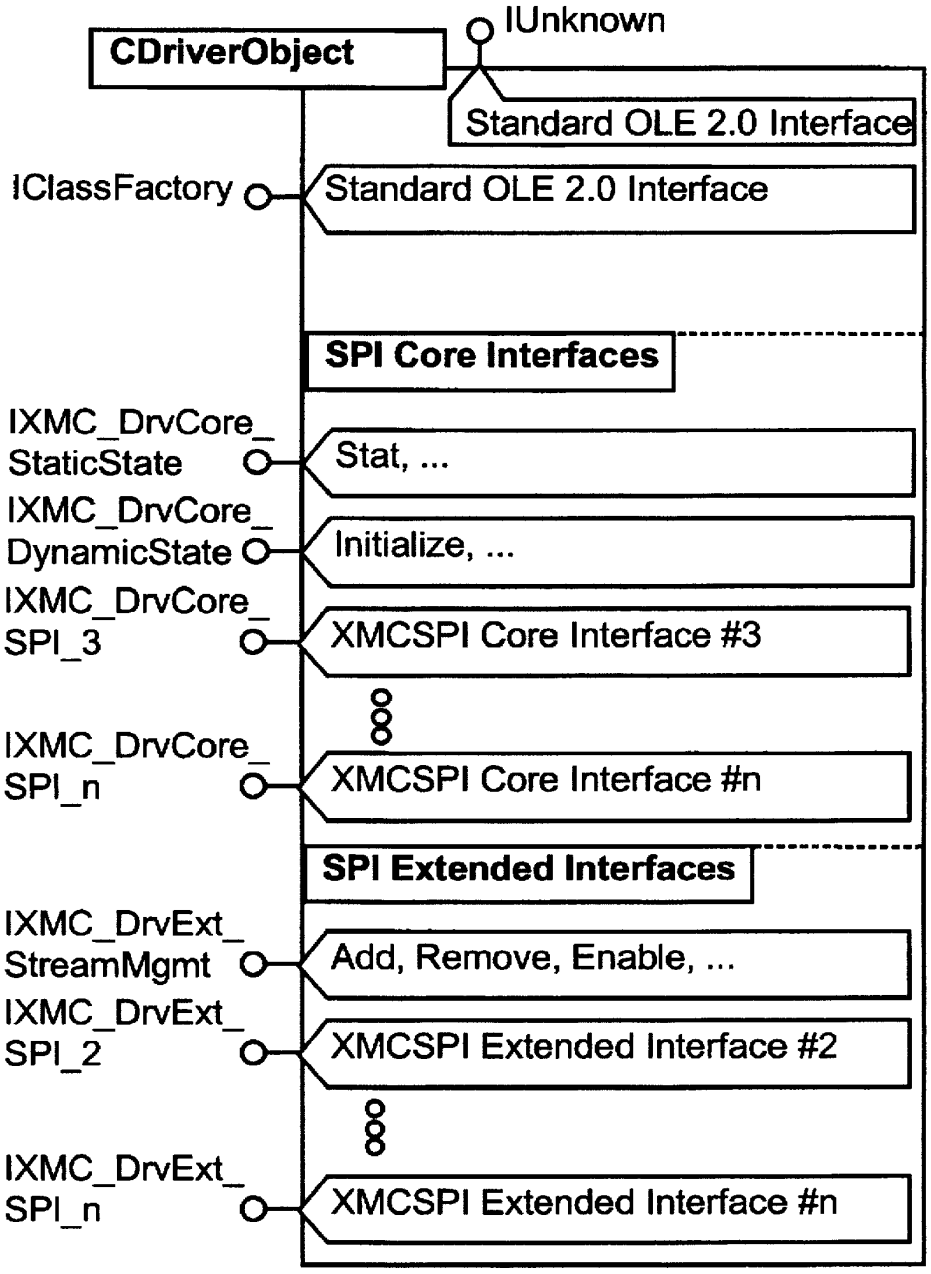

CDriverObject
IUnknown
Standard OLE 2.0 Interface
IClassFactory
Standard OLE 2.0 Interface
SPI Core Interfaces
IXMC_DrvCore_ StaticState
Stat, ...
IXMC_DrvCore_ DynamicState
Initialize, ...
IXMC_DrvCore_ SPI_3
XMCSPI Core Interface #3
IXMC_DrvCore_ SPI_n
XMCSPI Core Interface #n
SPI Extended Interfaces
IXMC_DrvExt_ StreamMgmt
Add, Remove, Enable, ...
IXMC_DrvExt_ SPI_2
XMCSPI Extended Interface #2
IXMC_DrvExt_ SPI_n
XMCSPI Extended Interface #n

FIG. 22

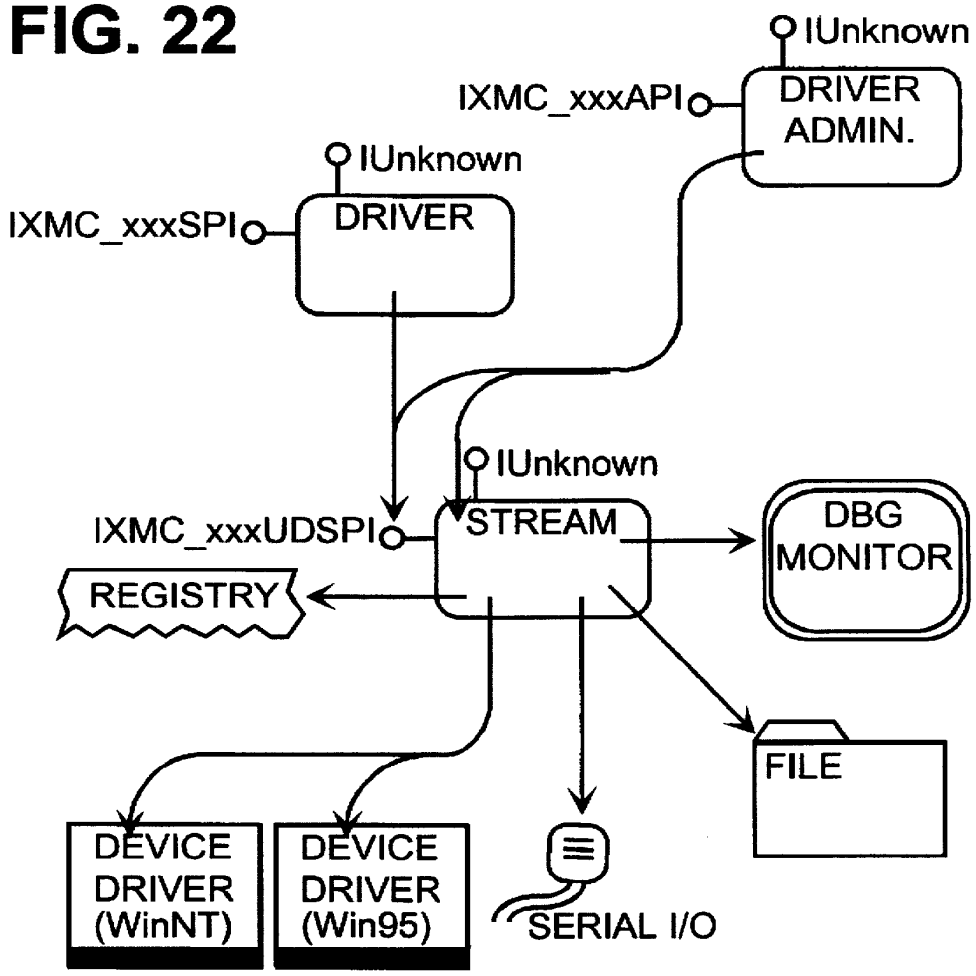

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
IUnknown
IXMC_xxxSPI
DRIVER
IUnknown
IXMC_xxxUDSPI
STREAM
DBG MONITOR
REGISTRY
FILE
DEVICE DRIVER (WinNT)
DEVICE DRIVER (Win95)
SERIAL I/O

FIG. 23

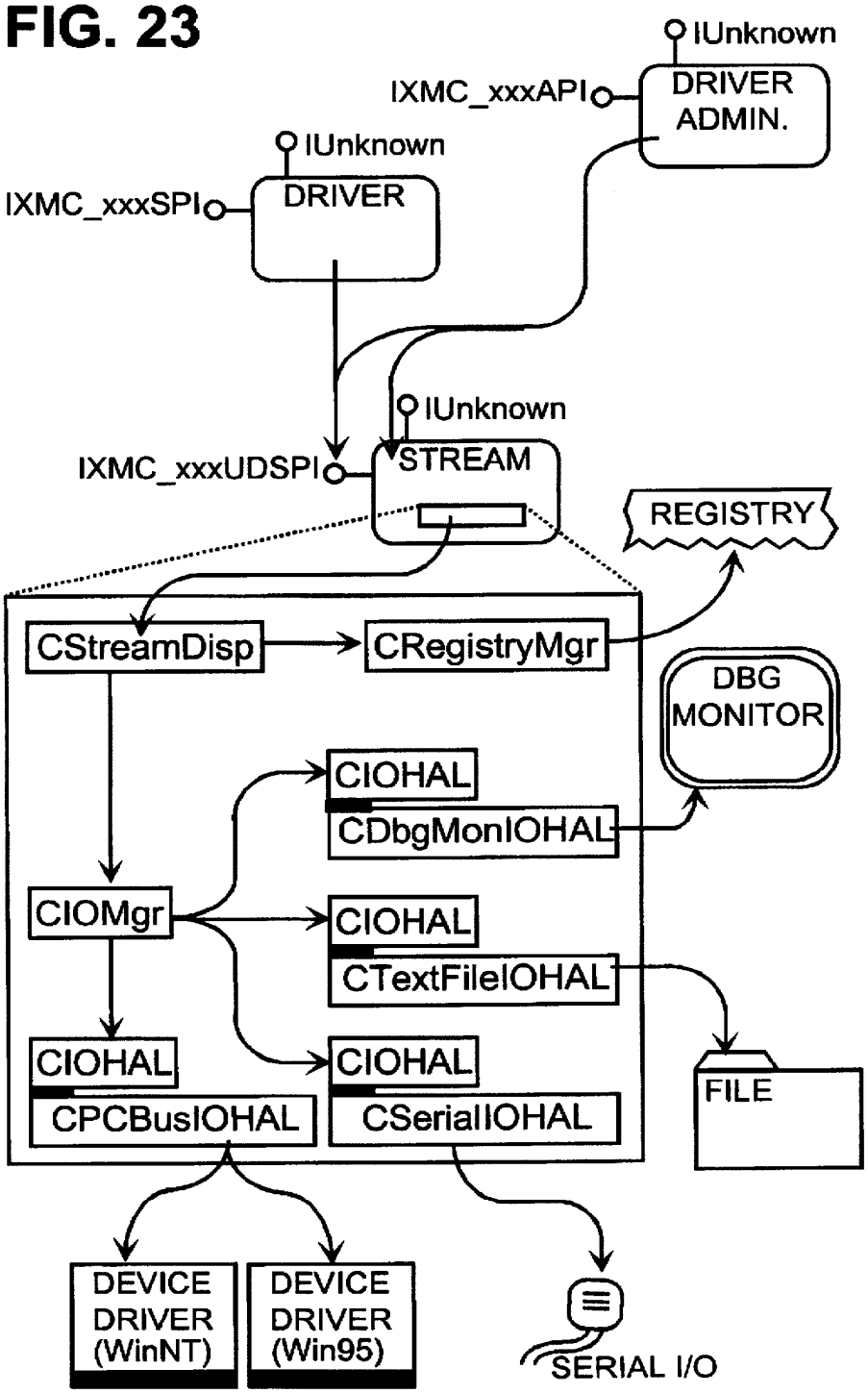

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
IUnknown
IXMC_xxxSPI
DRIVER
IUnknown
IXMC_xxxUDSPI
STREAM
REGISTRY
CStreamDisp
CRegistryMgr
DBG MONITOR
CIOHAL
CDbgMonIOHAL
CIOMgr
CIOHAL
CTextFileIOHAL
CIOHAL
CIOHAL
CPCBusIOHAL
CSerialIOHAL
FILE
DEVICE DRIVER (WinNT)
DEVICE DRIVER (Win95)
SERIAL I/O

FIG. 24

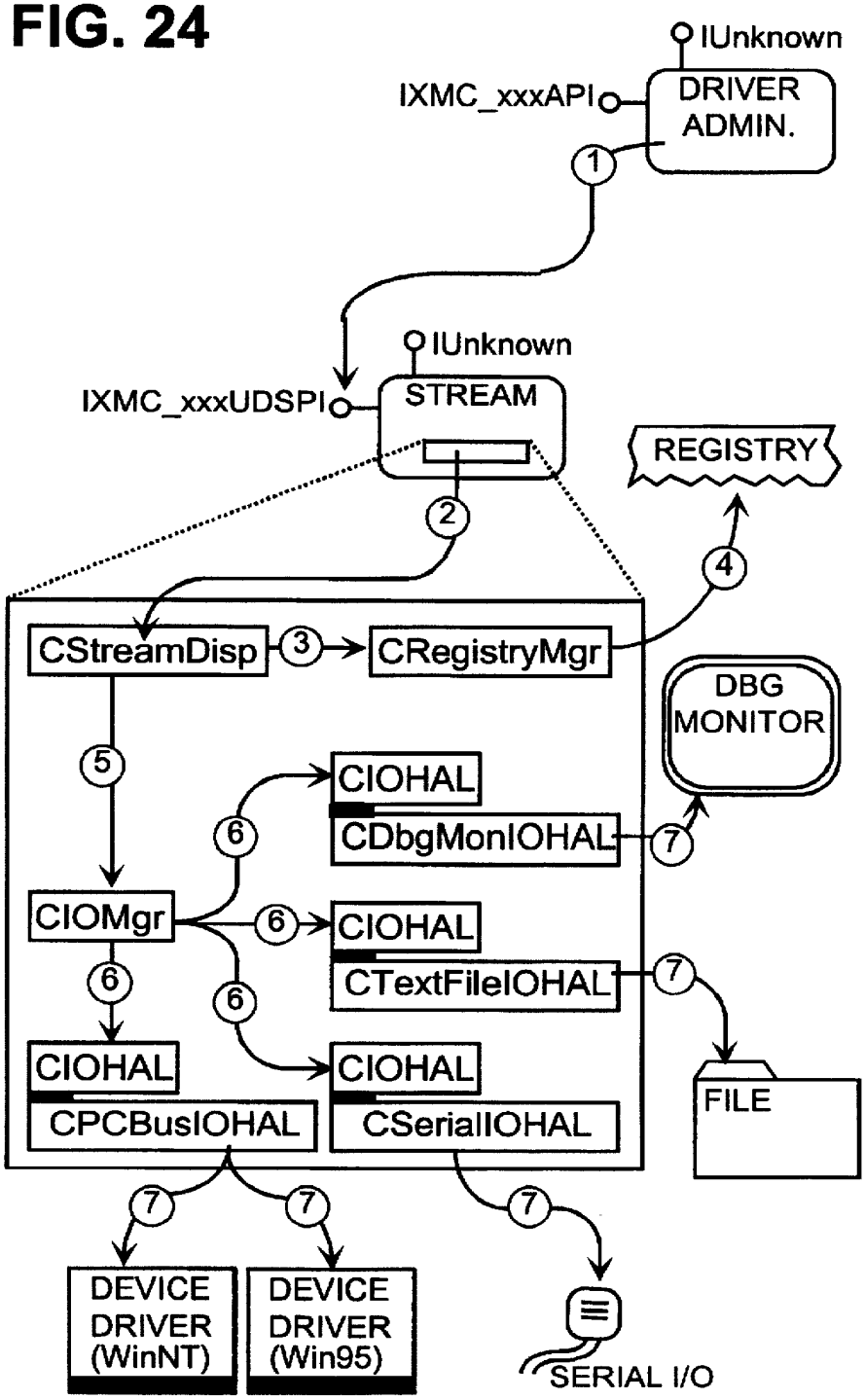

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
1
IUnknown
IXMC_xxxUDSPI
STREAM
REGISTRY
2
4
CStreamDisp
3
CRegistryMgr
DBG MONITOR
5
CIOHAL
CDbgMonIOHAL
7
6
CIOMgr
6
CIOHAL
CTextFileIOHAL
7
6
6
CIOHAL
CPCBusIOHAL
CIOHAL
CSerialIOHAL
FILE
7
7
7
DEVICE DRIVER (WinNT)
DEVICE DRIVER (Win95)
SERIAL I/O

FIG. 25

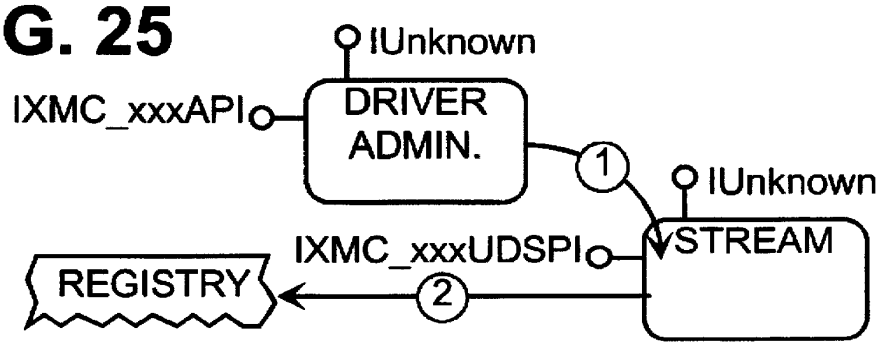

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
1
IUnknown
STREAM
IXMC_xxxUDSPI
REGISTRY
2

FIG. 26

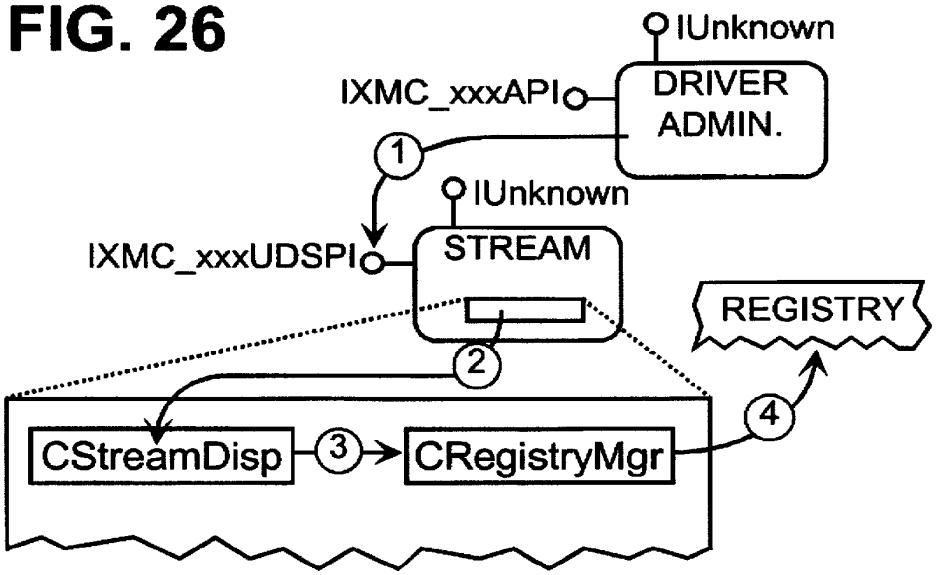

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
1
IUnknown
IXMC_xxxUDSPI
STREAM
REGISTRY
2
4
CStreamDisp
3
CRegistryMgr

FIG. 27

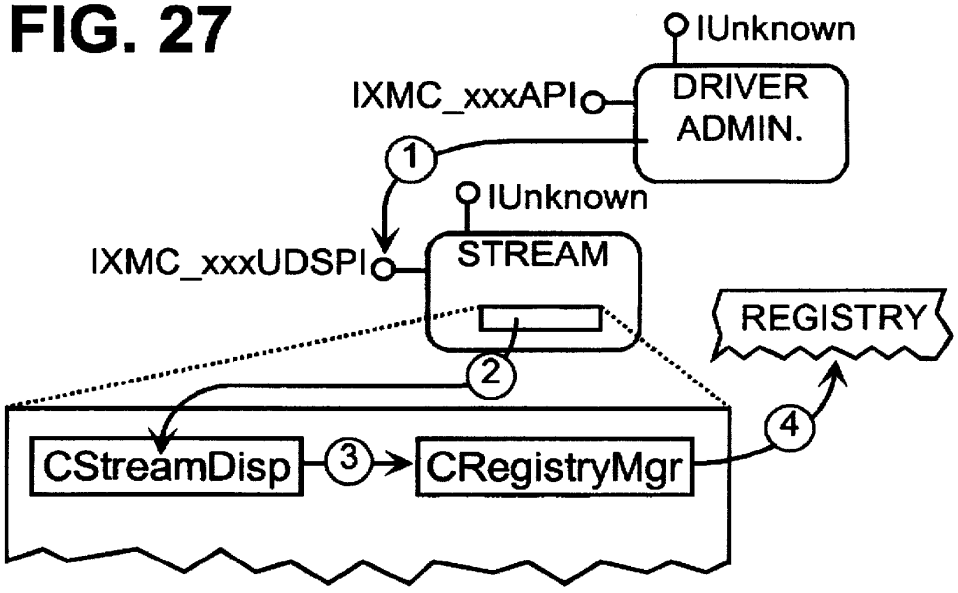

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
1
IUnknown
IXMC_xxxUDSPI
STREAM
REGISTRY
2
CStreamDisp
3
CRegistryMgr
4

FIG. 28

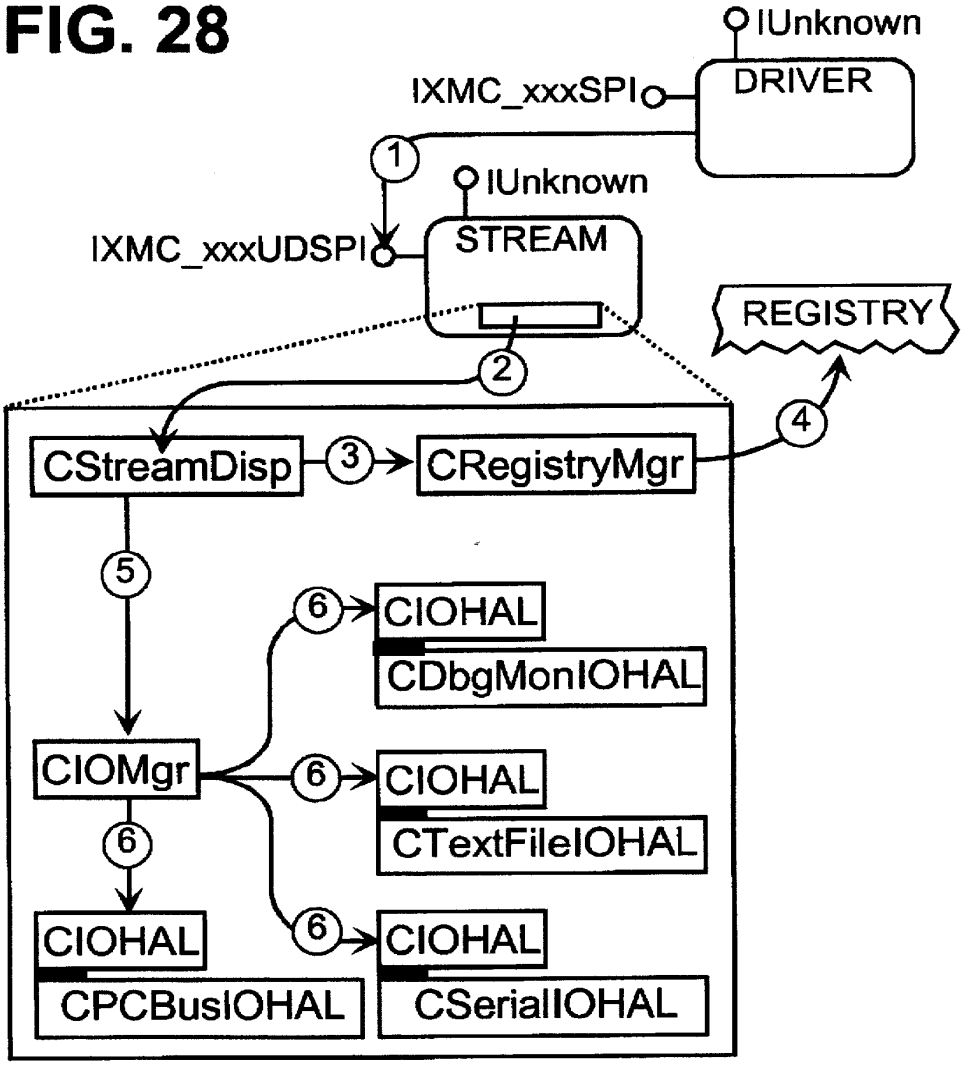

IUnknown
DRIVER
IXMC_xxxSPI
1
IUnknown
STREAM
IXMC_xxxUDSPI
REGISTRY
2
4
CStreamDisp
3
CRegistryMgr
5
6
CIOHAL
CDbgMonIOHAL
CIOMgr
6
CIOHAL
CTextFileIOHAL
6
CIOHAL
CIOHAL
CPCBusIOHAL
CSerialIOHAL

FIG. 29

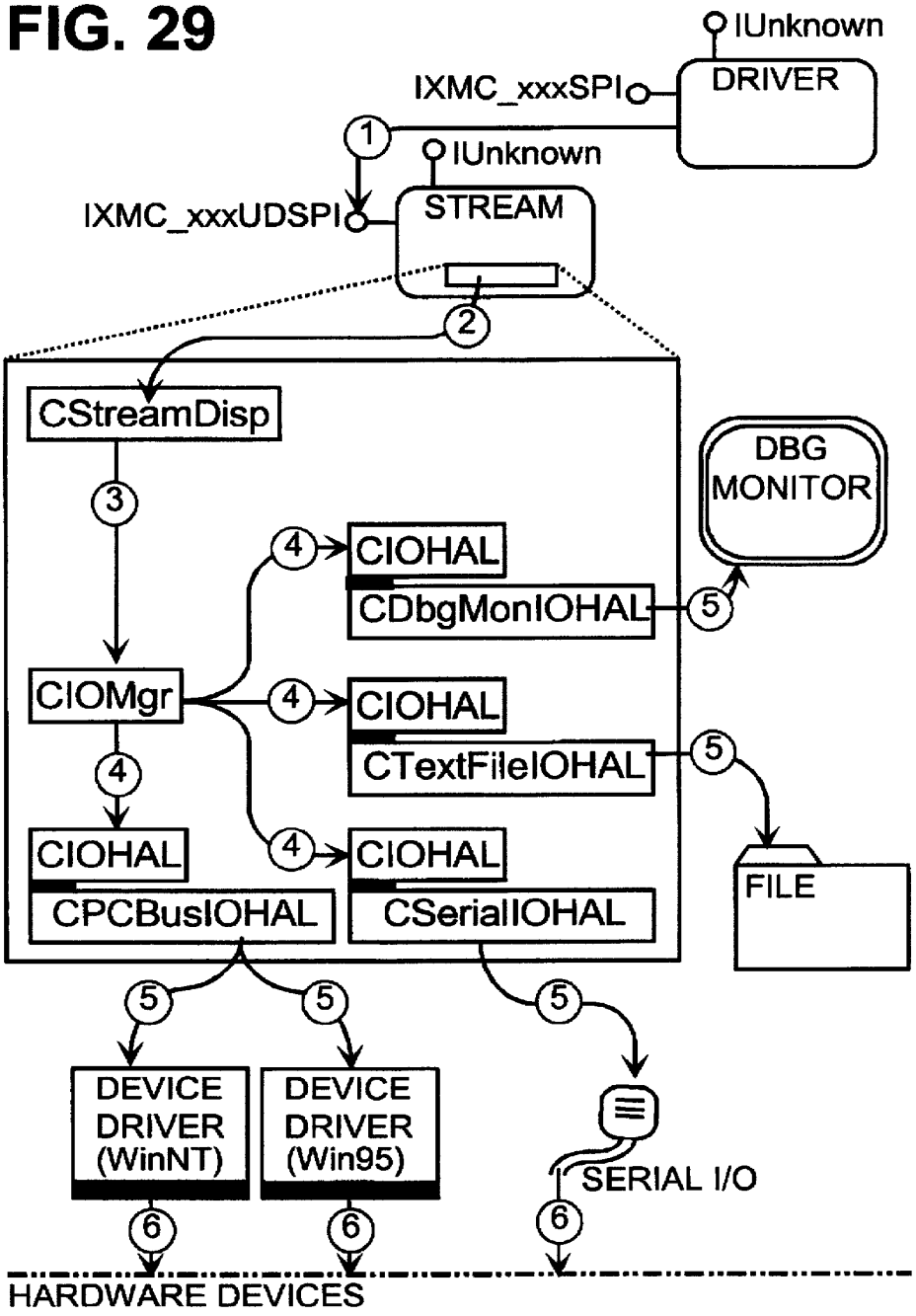

IUnknown
DRIVER
IXMC_xxxSPI
1
IUnknown
STREAM
IXMC_xxxUDSPI
2
CStreamDisp
DBG MONITOR
3
4
CIOHAL
CDbgMonIOHAL
5
CIOMgr
4
CIOHAL
CTextFileIOHAL
5
4
CIOHAL
CPCBusIOHAL
4
CIOHAL
CSerialIOHAL
FILE
5
5
5
DEVICE DRIVER (WinNT)
DEVICE DRIVER (Win95)
SERIAL I/O
6
6
6
HARDWARE DEVICES

FIG. 30

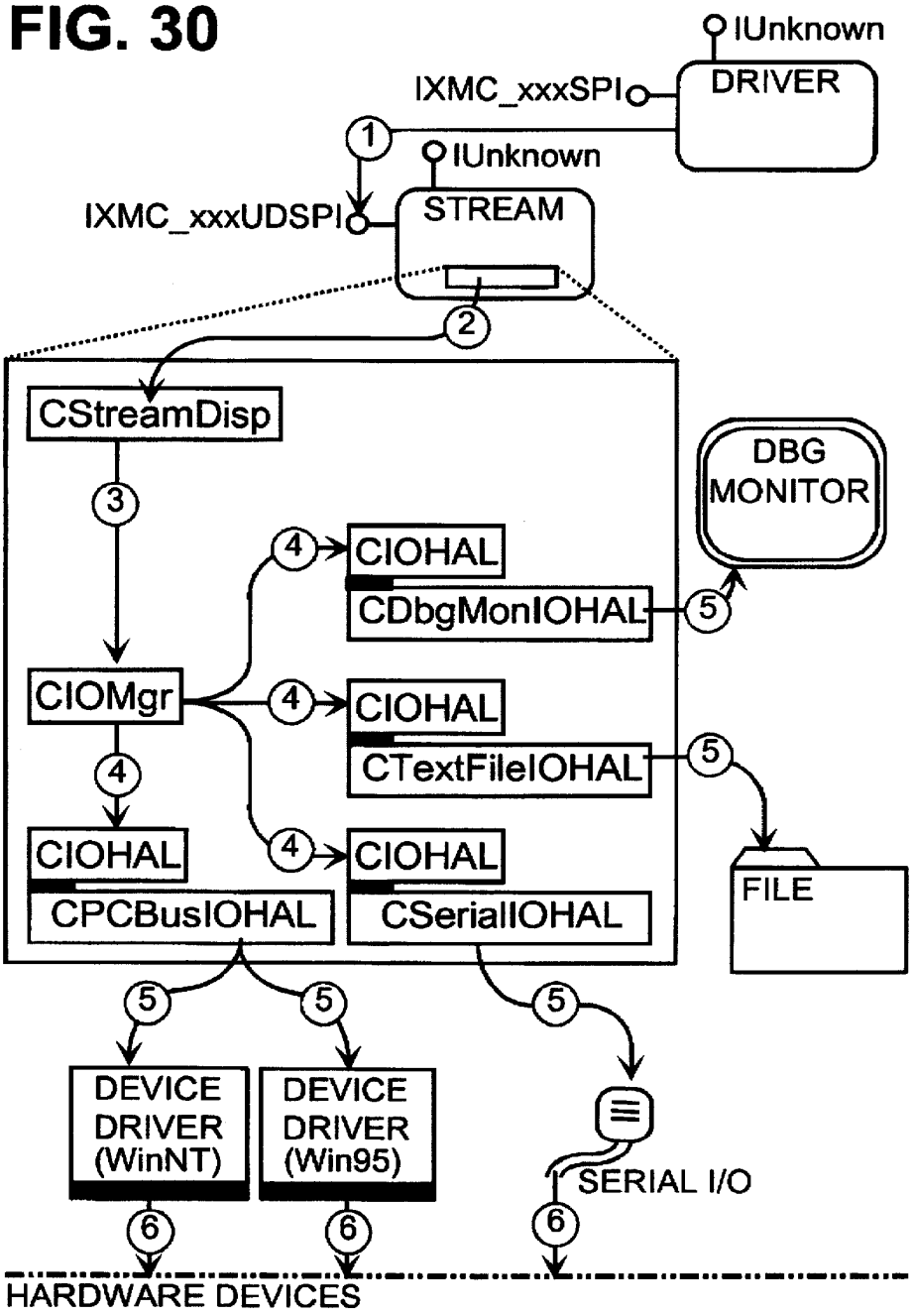

IUnknown
DRIVER
IXMC_xxxSPI
1
IUnknown
STREAM
IXMC_xxxUDSPI
2
CStreamDisp
DBG MONITOR
3
4
CIOHAL
CDbgMonIOHAL
5
CIOMgr
4
CIOHAL
CTextFileIOHAL
5
4
CIOHAL
CIOHAL
CPCBusIOHAL
CSerialIOHAL
FILE
5
5
5
DEVICE DRIVER (WinNT)
DEVICE DRIVER (Win95)
SERIAL I/O
6
6
6
HARDWARE DEVICES

FIG. 31

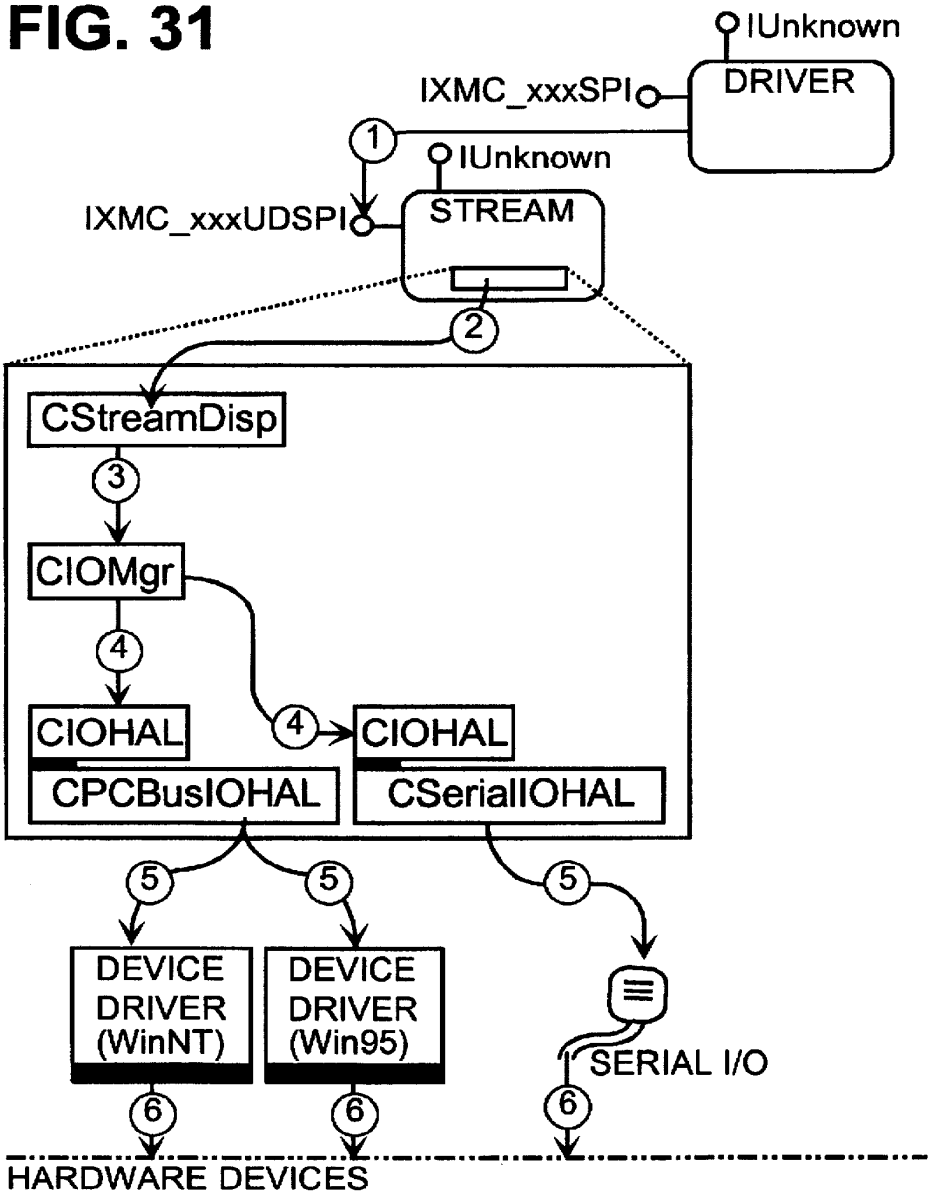

IUnknown
DRIVER
IXMC_xxxSPI
1
IUnknown
IXMC_xxxUDSPI
STREAM
2
CStreamDisp
3
CIOMgr
4
CIOHAL
CPCBusIOHAL
4
CIOHAL
CSerialIOHAL
5
5
5
DEVICE DRIVER (WinNT)
DEVICE DRIVER (Win95)
SERIAL I/O
6
6
6
HARDWARE DEVICES

FIG. 32

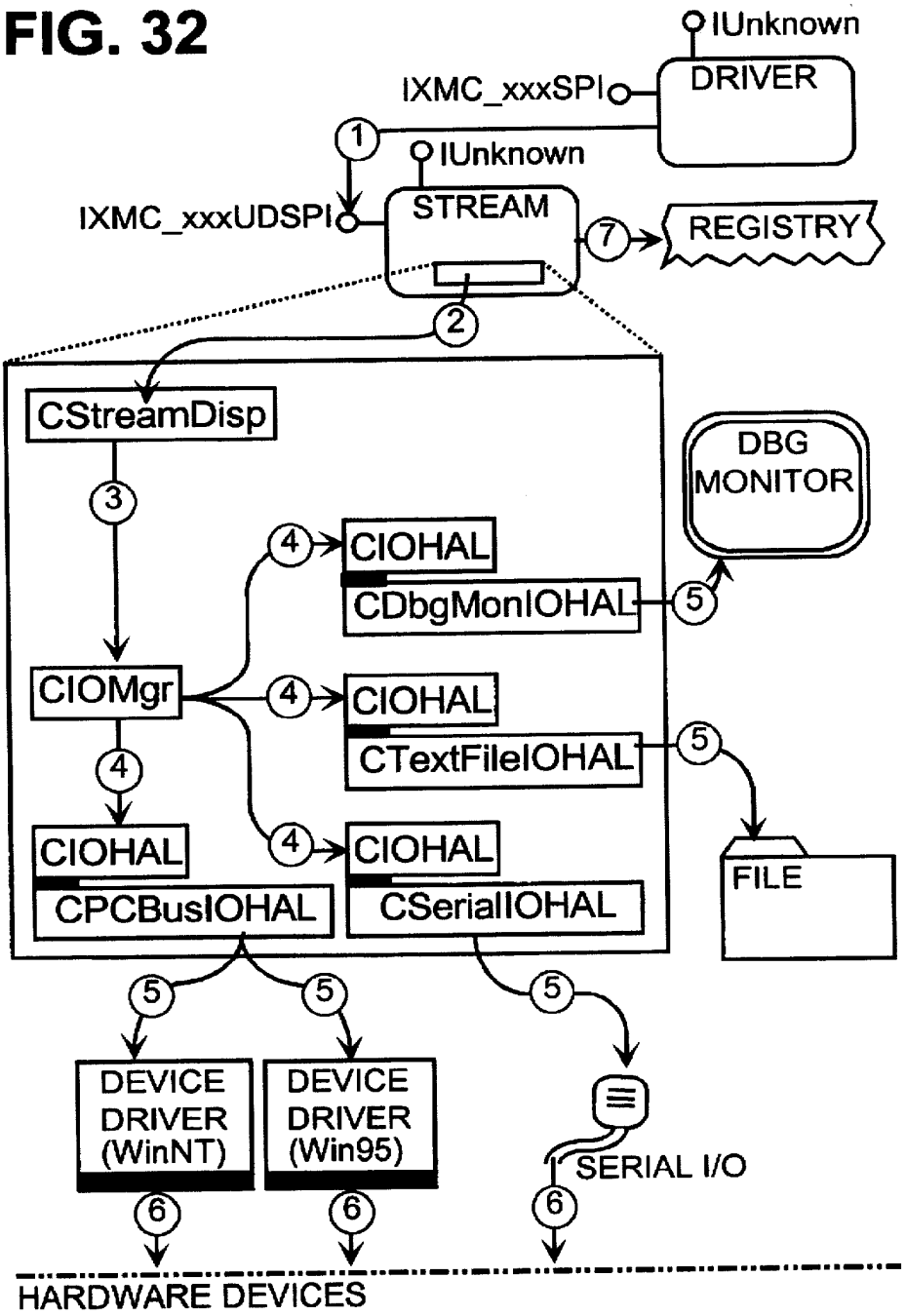

IUnknown
DRIVER
IXMC_xxxSPI
1
IUnknown
STREAM
IXMC_xxxUDSPI
7
REGISTRY
2
CStreamDisp
DBG MONITOR
3
4
CIOHAL
CDbgMonIOHAL
5
CIOMgr
4
CIOHAL
CTextFileIOHAL
5
4
CIOHAL
CPCBusIOHAL
4
CIOHAL
CSerialIOHAL
FILE
5
5
5
DEVICE DRIVER (WinNT)
DEVICE DRIVER (Win95)
SERIAL I/O
6
6
6
HARDWARE DEVICES

FIG. 33

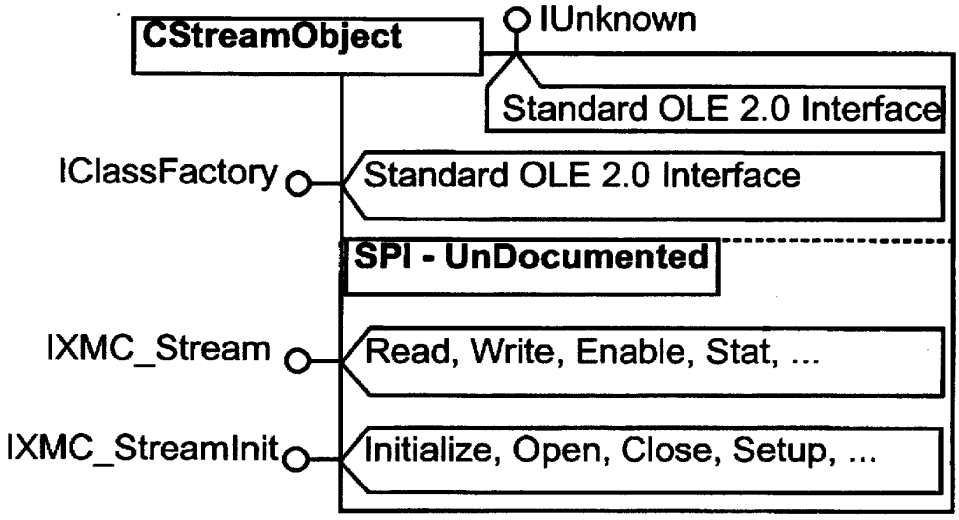

CStreamObject
IUnknown
Standard OLE 2.0 Interface
IClassFactory
Standard OLE 2.0 Interface
SPI - UnDocumented
IXMC_Stream
Read, Write, Enable, Stat, ...
IXMC_StreamInit
Initialize, Open, Close, Setup, ...

FIG. 34

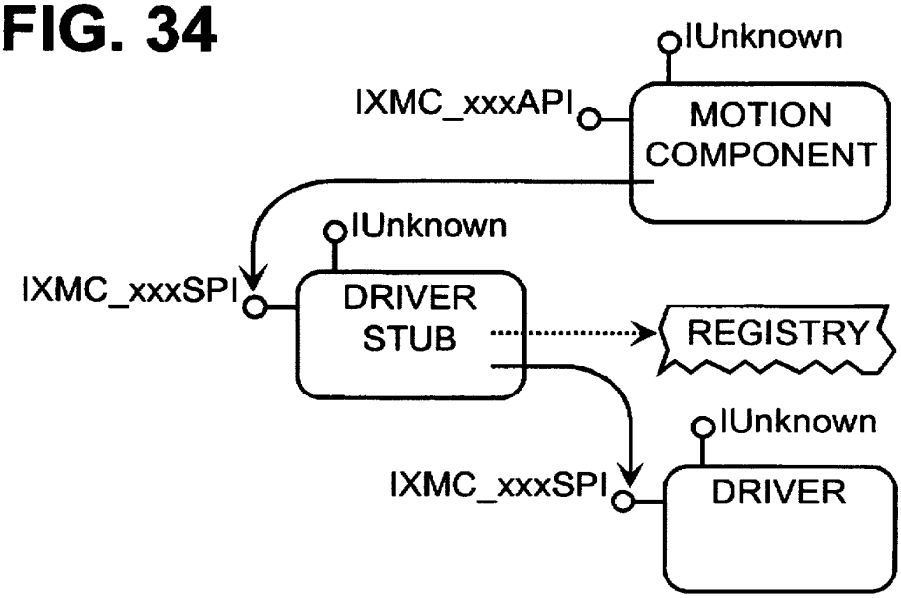

IUnknown
IXMC_xxxAPI
MOTION COMPONENT
IUnknown
IXMC_xxxSPI
DRIVER STUB
REGISTRY
IUnknown
IXMC_xxxSPI
DRIVER

FIG. 35

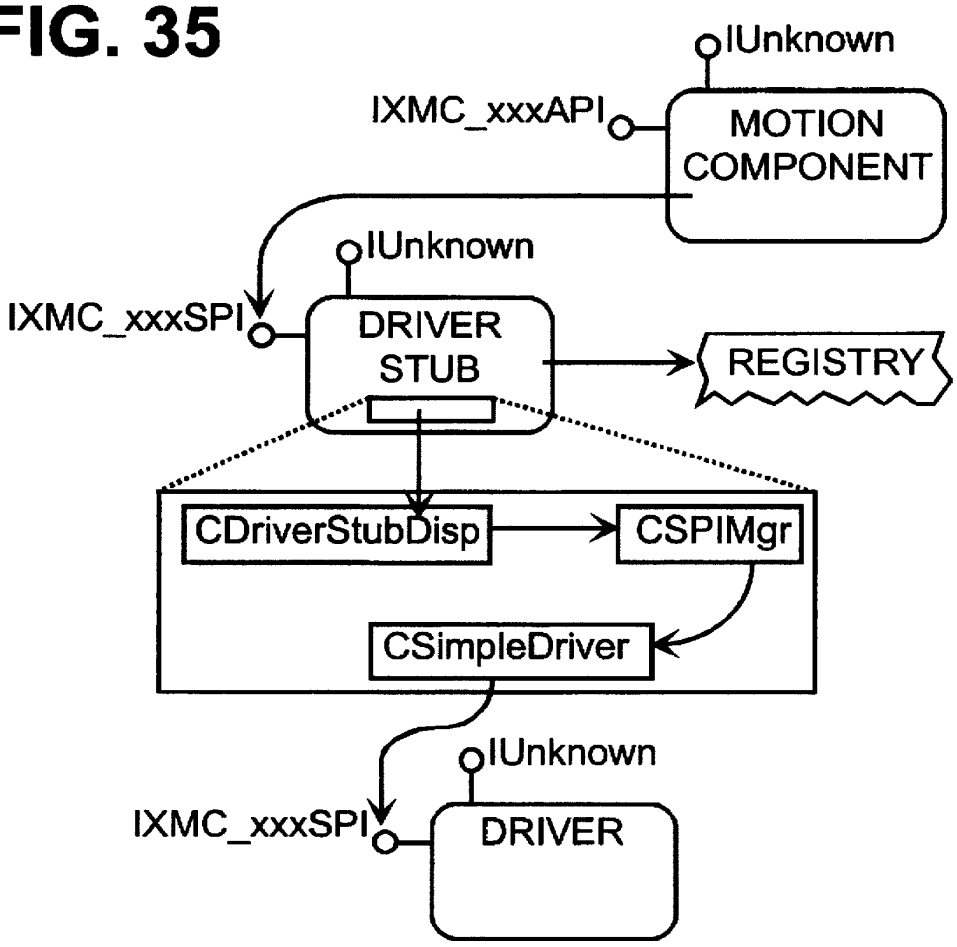

IUnknown
IXMC_xxxAPI
MOTION COMPONENT
IUnknown
IXMC_xxxSPI
DRIVER STUB
REGISTRY
CDriverStubDisp
CSPIMgr
CSimpleDriver
IUnknown
IXMC_xxxSPI
DRIVER

FIG. 36

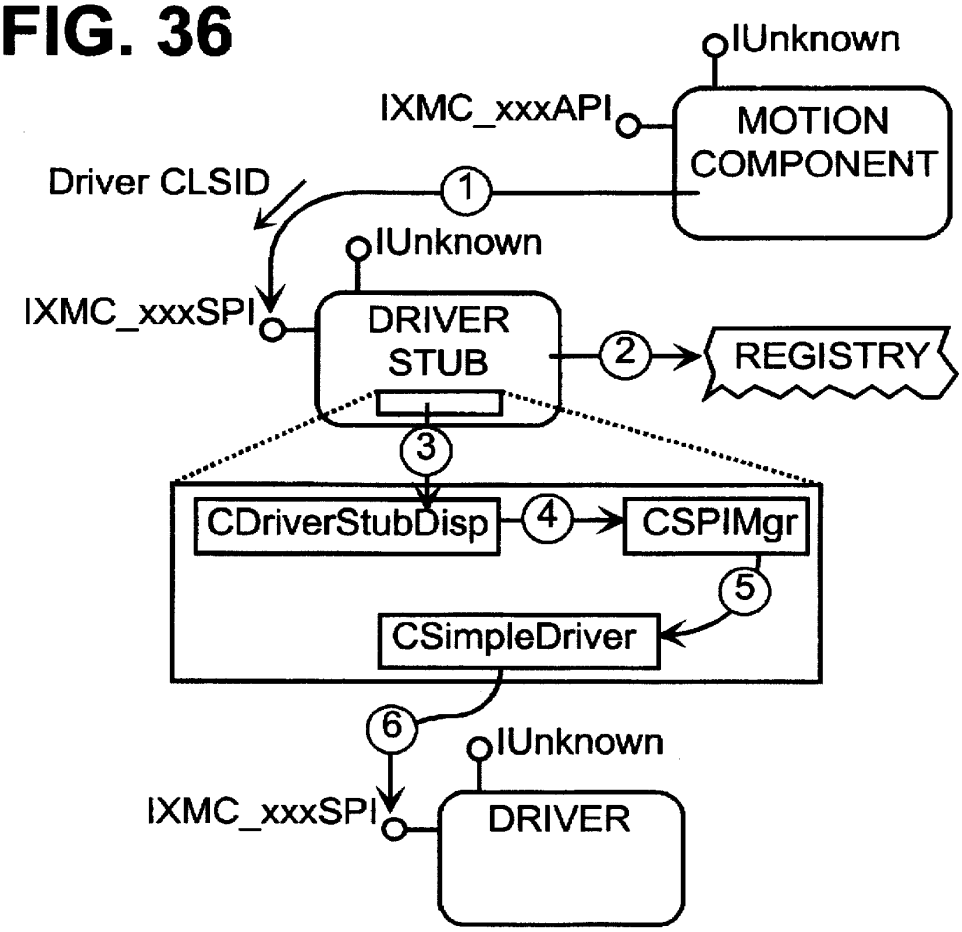

IUnknown
IXMC_xxxAPI
MOTION COMPONENT
Driver CLSID
1
IUnknown
IXMC_xxxSPI
DRIVER STUB
2
REGISTRY
3
CDriverStubDisp
4
CSPIMgr
5
CSimpleDriver
6
IUnknown
IXMC_xxxSPI
DRIVER

FIG. 37

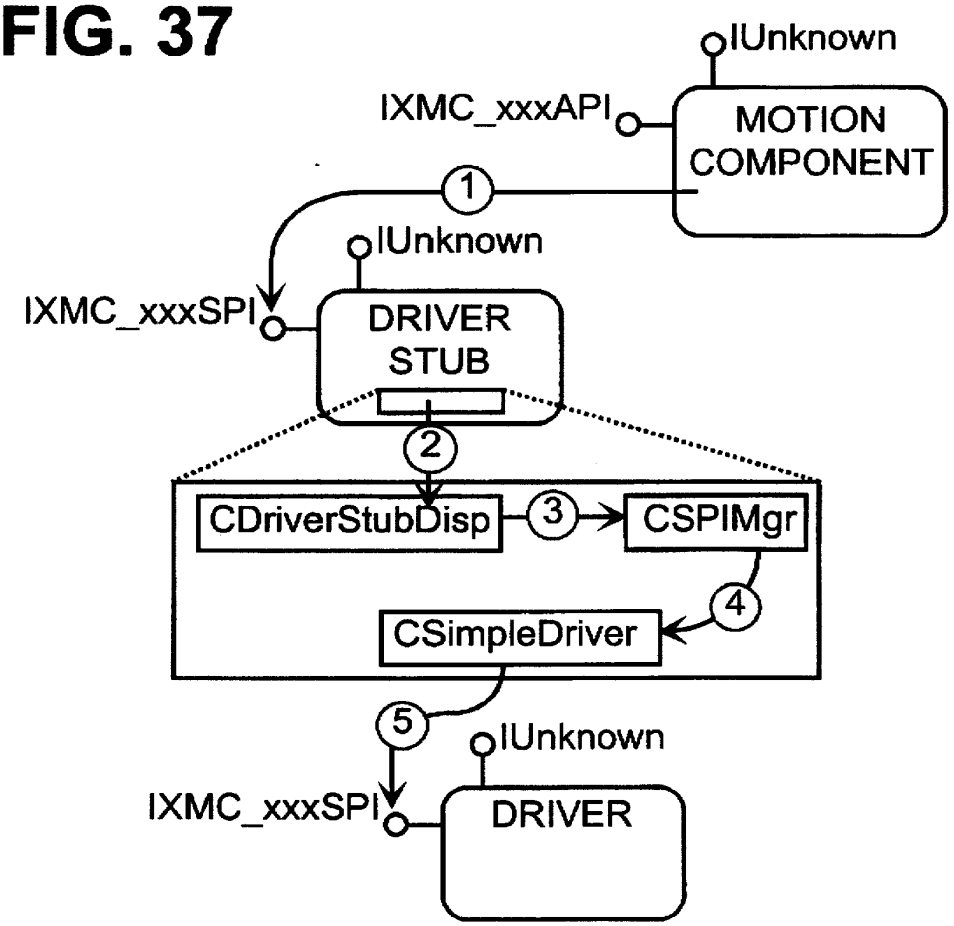

IUnknown
IXMC_xxxAPI
MOTION COMPONENT
1
IUnknown
IXMC_xxxSPI
DRIVER STUB
2
CDriverStubDisp
3
CSPIMgr
4
CSimpleDriver
5
IUnknown
IXMC_xxxSPI
DRIVER

FIG. 38

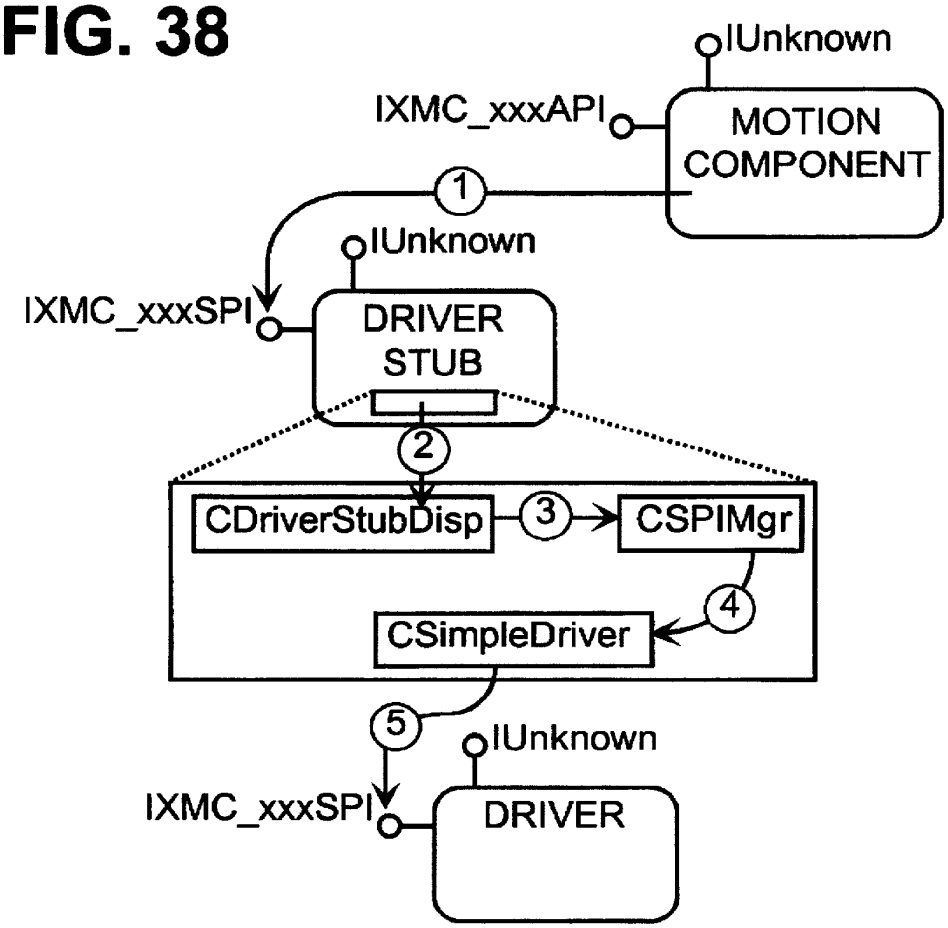

IUnknown
IXMC_xxxAPI
MOTION COMPONENT
1
IUnknown
IXMC_xxxSPI
DRIVER STUB
2
CDriverStubDisp
3
CSPIMgr
4
CSimpleDriver
5
IUnknown
IXMC_xxxSPI
DRIVER

FIG. 39

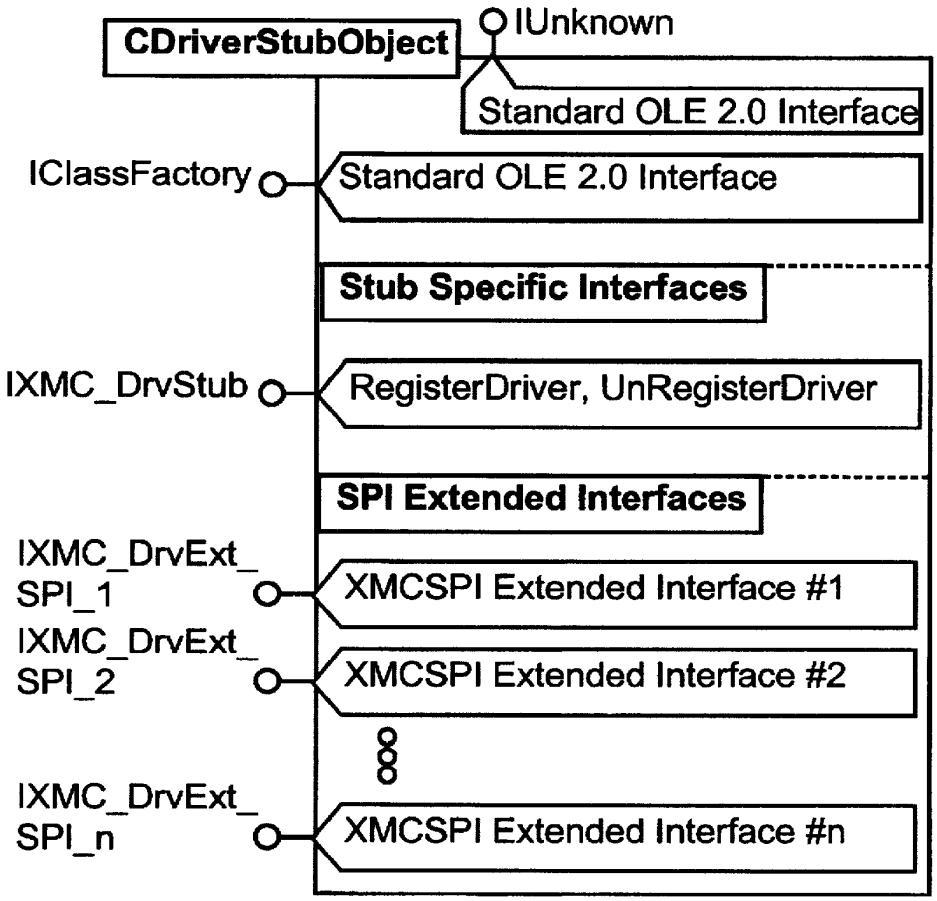

CDriverStubObject
IUnknown
Standard OLE 2.0 Interface
IClassFactory
Standard OLE 2.0 Interface
Stub Specific Interfaces
IXMC_DrvStub
RegisterDriver, UnRegisterDriver
SPI Extended Interfaces
IXMC_DrvExt_SPI_1
XMCSPI Extended Interface #1
IXMC_DrvExt_SPI_2
XMCSPI Extended Interface #2
IXMC_DrvExt_SPI_n
XMCSPI Extended Interface #n

FIG. 40

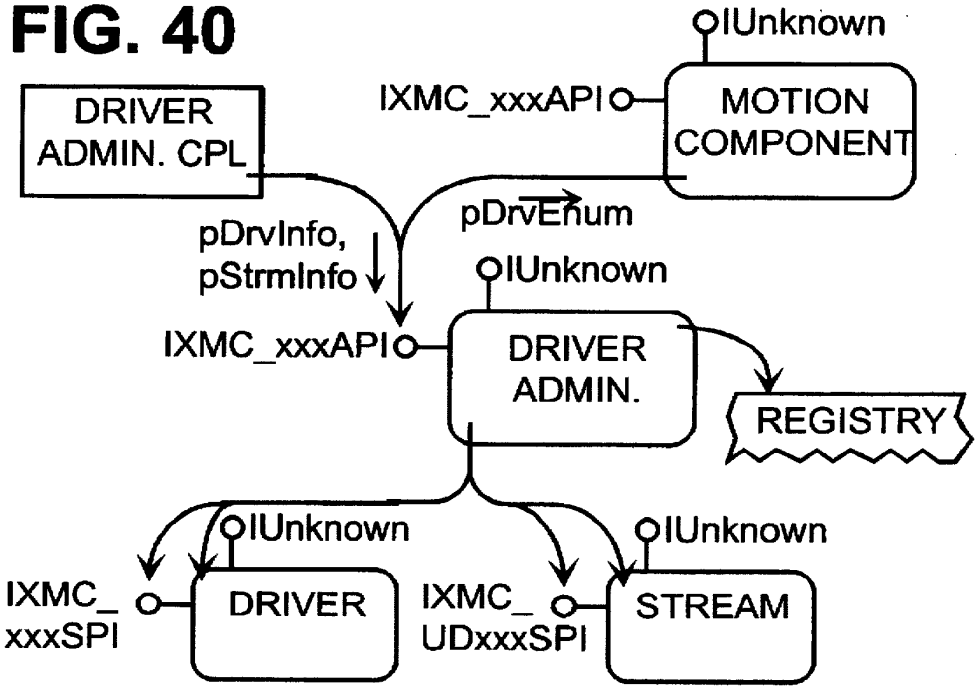

IUnknown
IXMC_xxxAPI
MOTION COMPONENT
DRIVER ADMIN. CPL
pDrvEnum
pDrvInfo,
pStrmInfo
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
REGISTRY
IUnknown
IUnknown
IXMC_ xxxSPI
DRIVER
IXMC_ UDxxxSPI
STREAM

FIG. 41

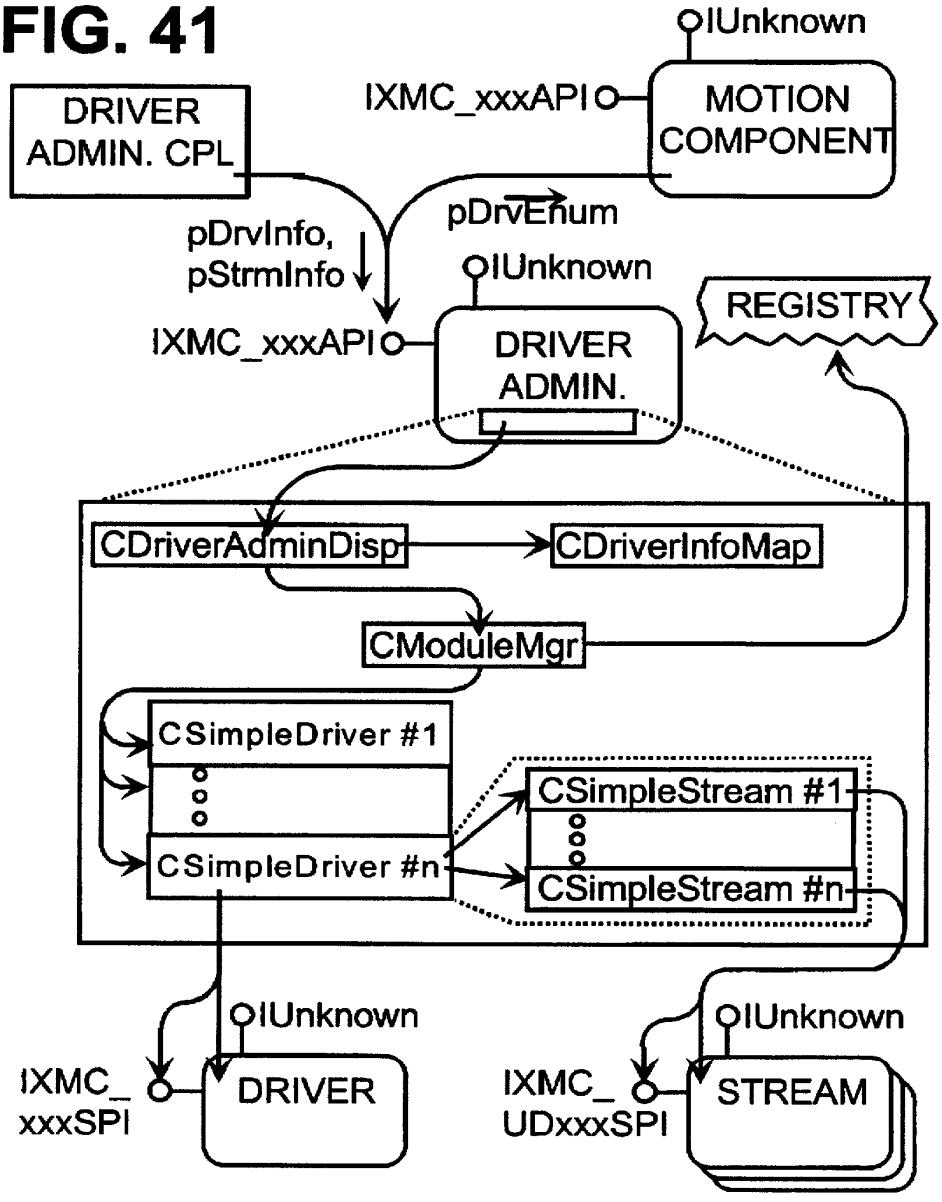

IUnknown
IXMC_xxxAPI
MOTION COMPONENT
DRIVER ADMIN. CPL
pDrvEnum
pDrvInfo, pStrmInfo
IUnknown
REGISTRY
IXMC_xxxAPI
DRIVER ADMIN.
CDriverAdminDisp
CDriverInfoMap
CModuleMgr
CSimpleDriver #1
CSimpleStream #1
CSimpleDriver #n
CSimpleStream #n
IUnknown
IUnknown
IXMC_ xxxSPI
DRIVER
IXMC_ UDxxxSPI
STREAM

FIG. 42

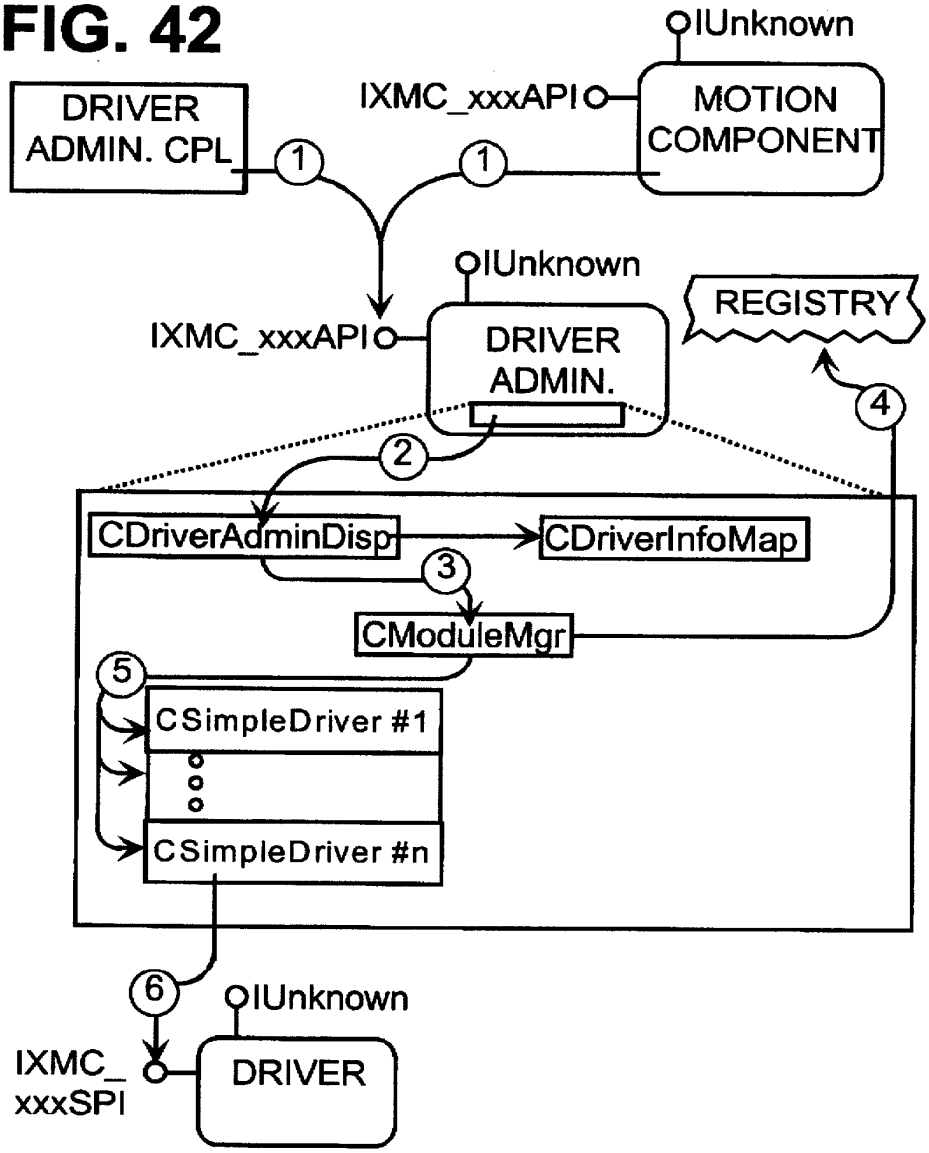

IUnknown
DRIVER ADMIN. CPL
IXMC_xxxAPI
MOTION COMPONENT
1
1
IUnknown
REGISTRY
IXMC_xxxAPI
DRIVER ADMIN.
4
2
CDriverAdminDisp
CDriverInfoMap
3
CModuleMgr
5
CSimpleDriver #1
CSimpleDriver #n
6
IUnknown
IXMC_ xxxSPI
DRIVER

FIG. 43

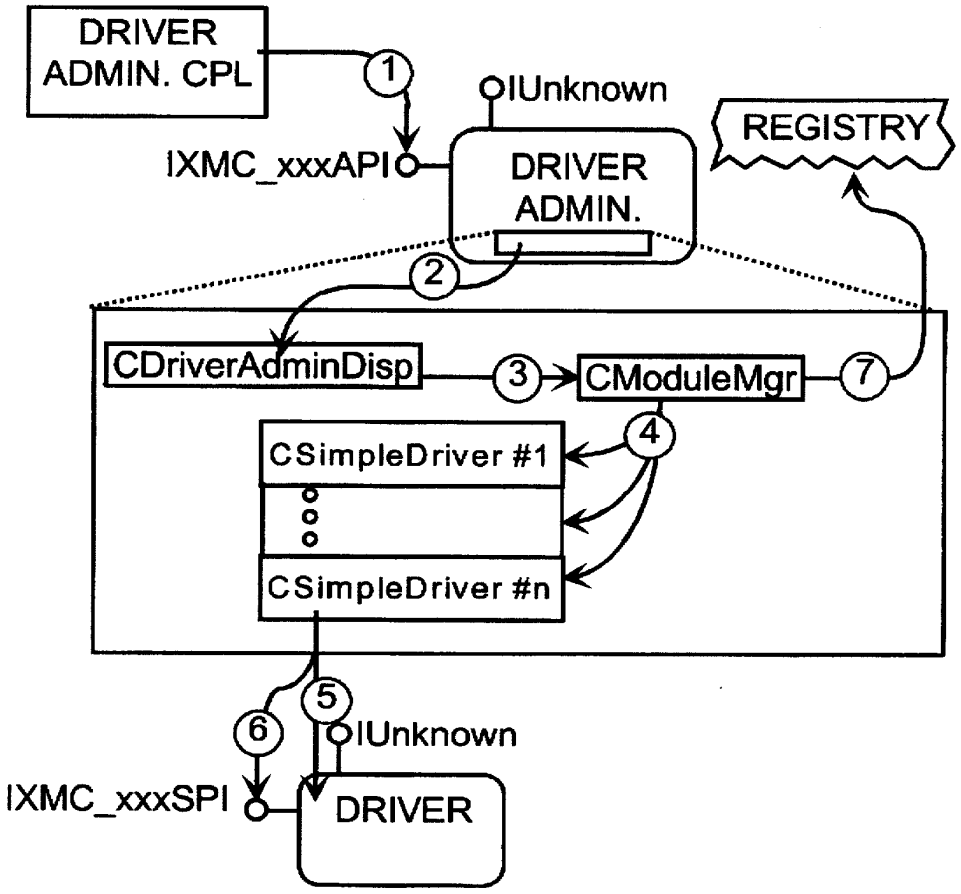

DRIVER ADMIN. CPL
1
IUnknown
REGISTRY
IXMC_xxxAPI
DRIVER ADMIN.
2
CDriverAdminDisp
3
CModuleMgr
7
4
CSimpleDriver #1
CSimpleDriver #n
5
6
IUnknown
IXMC_xxxSPI
DRIVER

FIG. 44

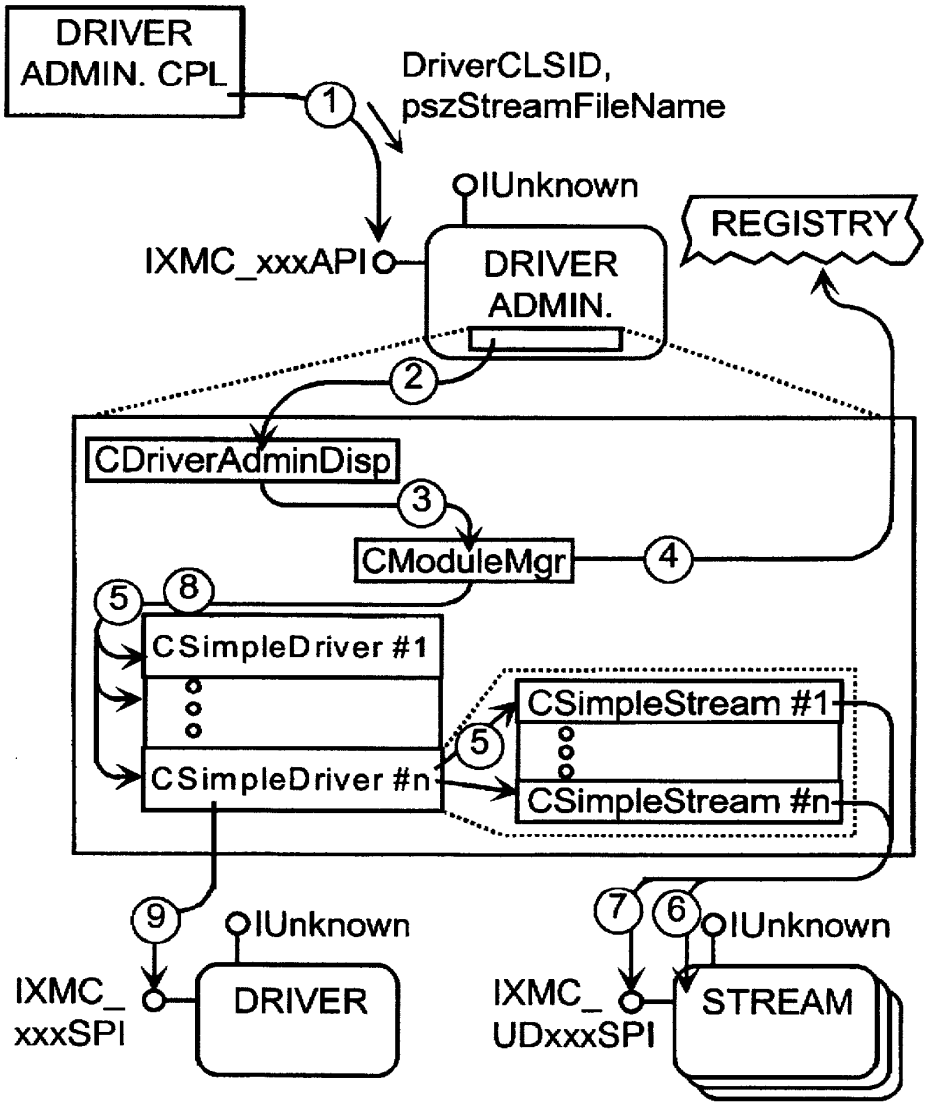

DRIVER ADMIN. CPL
1
DriverCLSID, pszStreamFileName
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
REGISTRY
2
CDriverAdminDisp
3
CModuleMgr
4
5
8
CSimpleDriver #1
CSimpleDriver #n
CSimpleStream #1
CSimpleStream #n
5
9
IUnknown
IXMC_ xxxSPI
DRIVER
7
6
IUnknown
IXMC_ UDxxxSPI
STREAM

FIG. 45

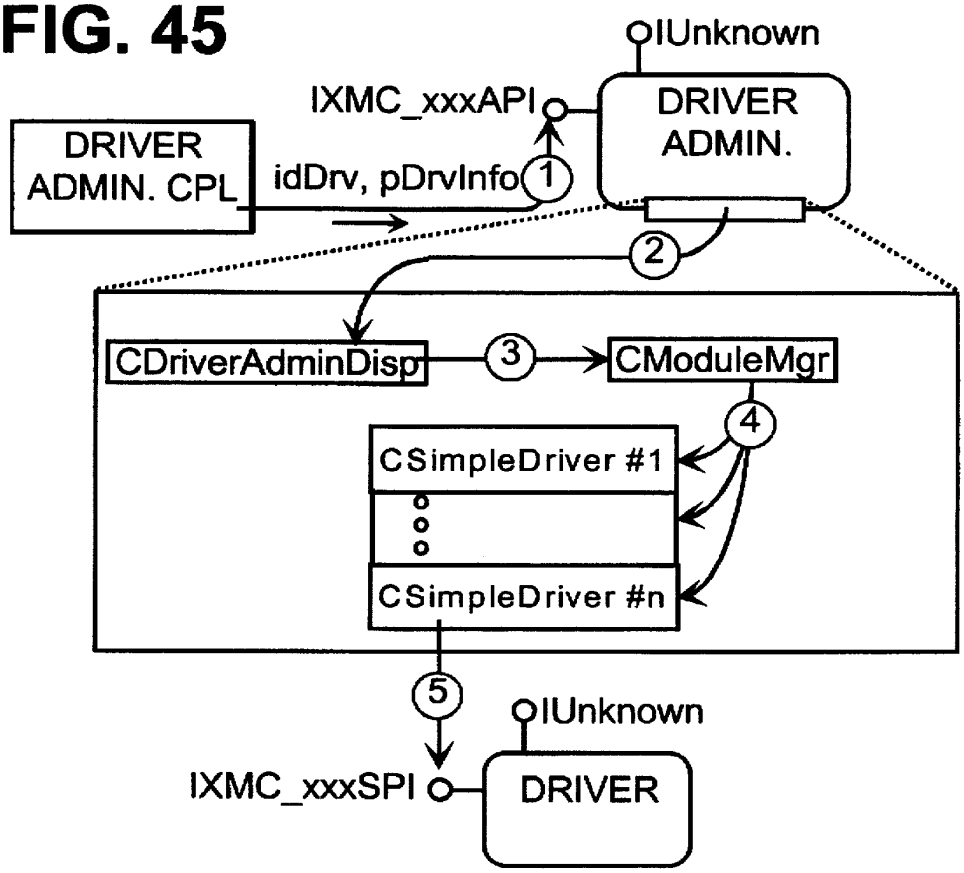

IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
DRIVER ADMIN. CPL
idDrv, pDrvInfo
1
2
CDriverAdminDisp
3
CModuleMgr
4
CSimpleDriver #1
CSimpleDriver #n
5
IUnknown
IXMC_xxxSPI
DRIVER

FIG. 46

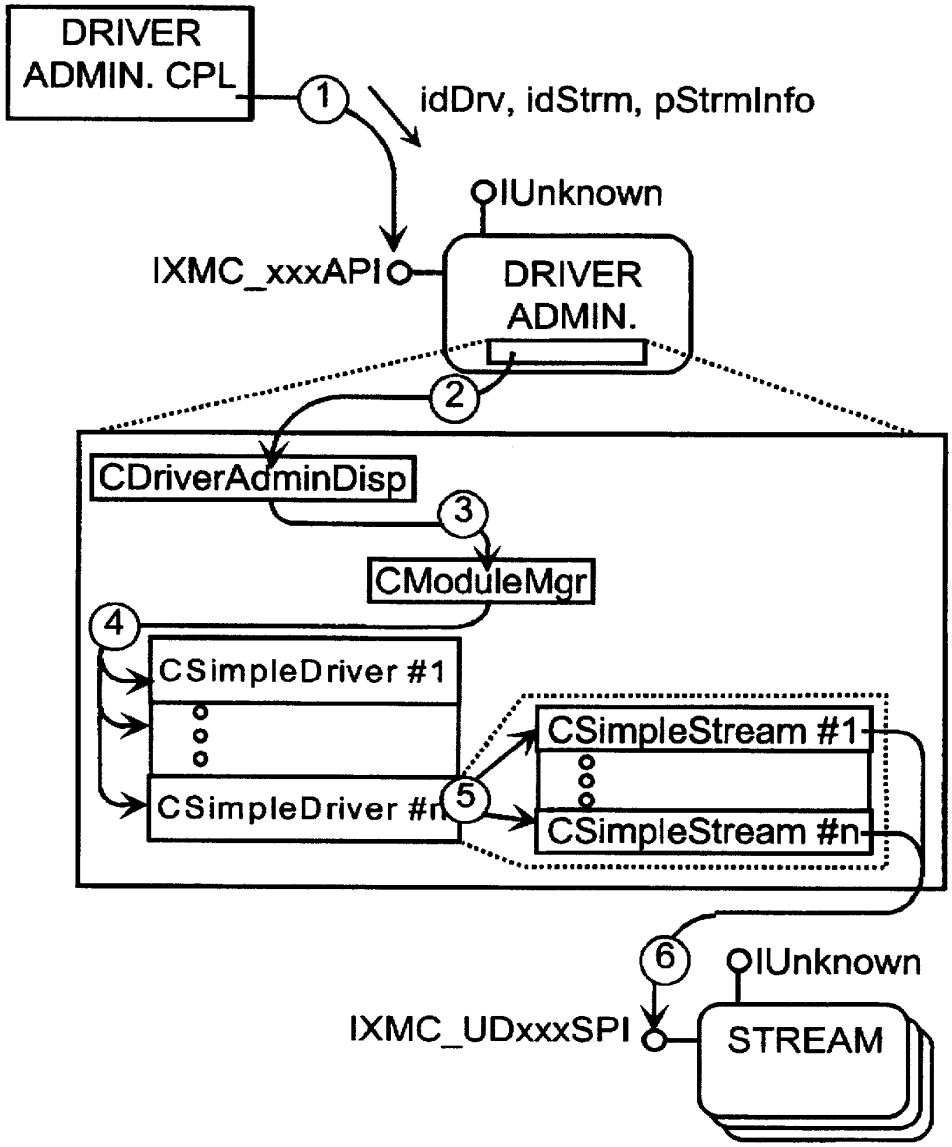

DRIVER ADMIN. CPL
1
idDrv, idStrm, pStrmInfo
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
2
CDriverAdminDisp
3
CModuleMgr
4
CSimpleDriver #1
CSimpleDriver #n
5
CSimpleStream #1
CSimpleStream #n
6
IUnknown
IXMC_UDxxxSPI
STREAM

FIG. 47

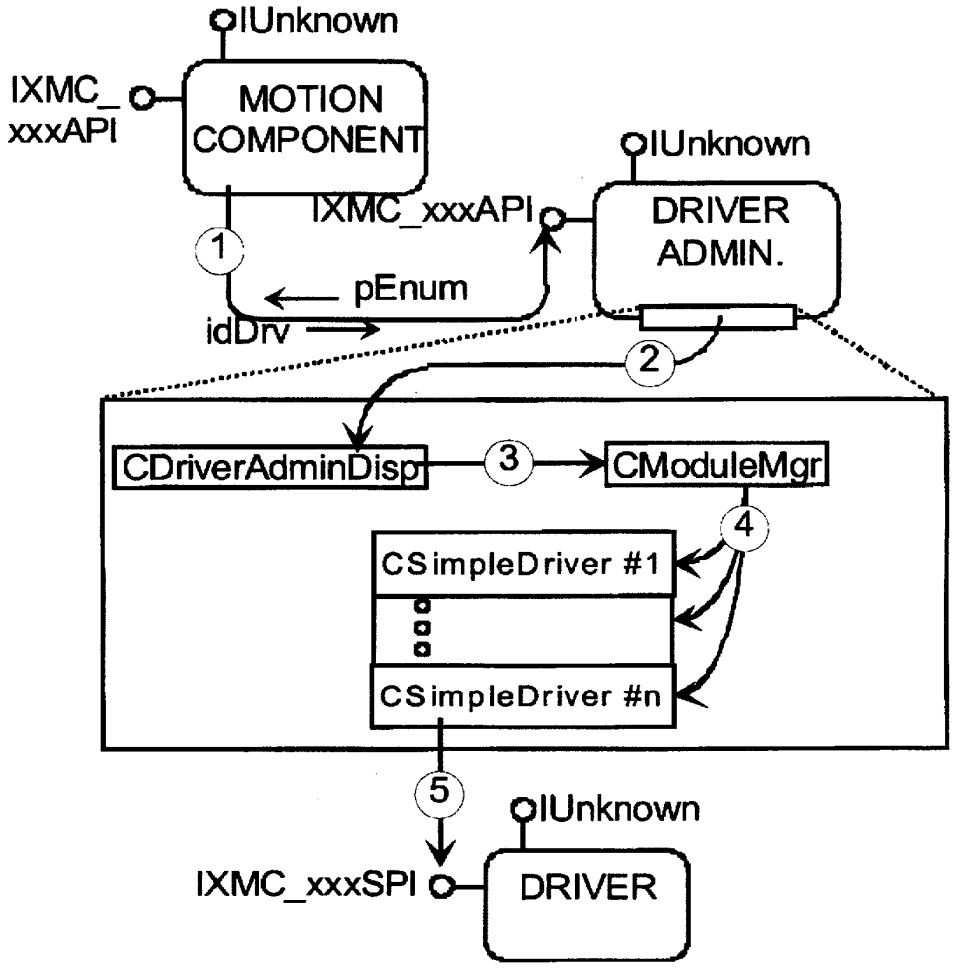

IUnknown
IXMC_
xxxAPI
MOTION
COMPONENT
IUnknown
IXMC_xxxAPI
DRIVER
ADMIN.
1
pEnum
idDrv
2
CDriverAdminDisp
3
CModuleMgr
4
CSimpleDriver #1
CSimpleDriver #n
5
IUnknown
IXMC_xxxSPI
DRIVER

FIG. 48

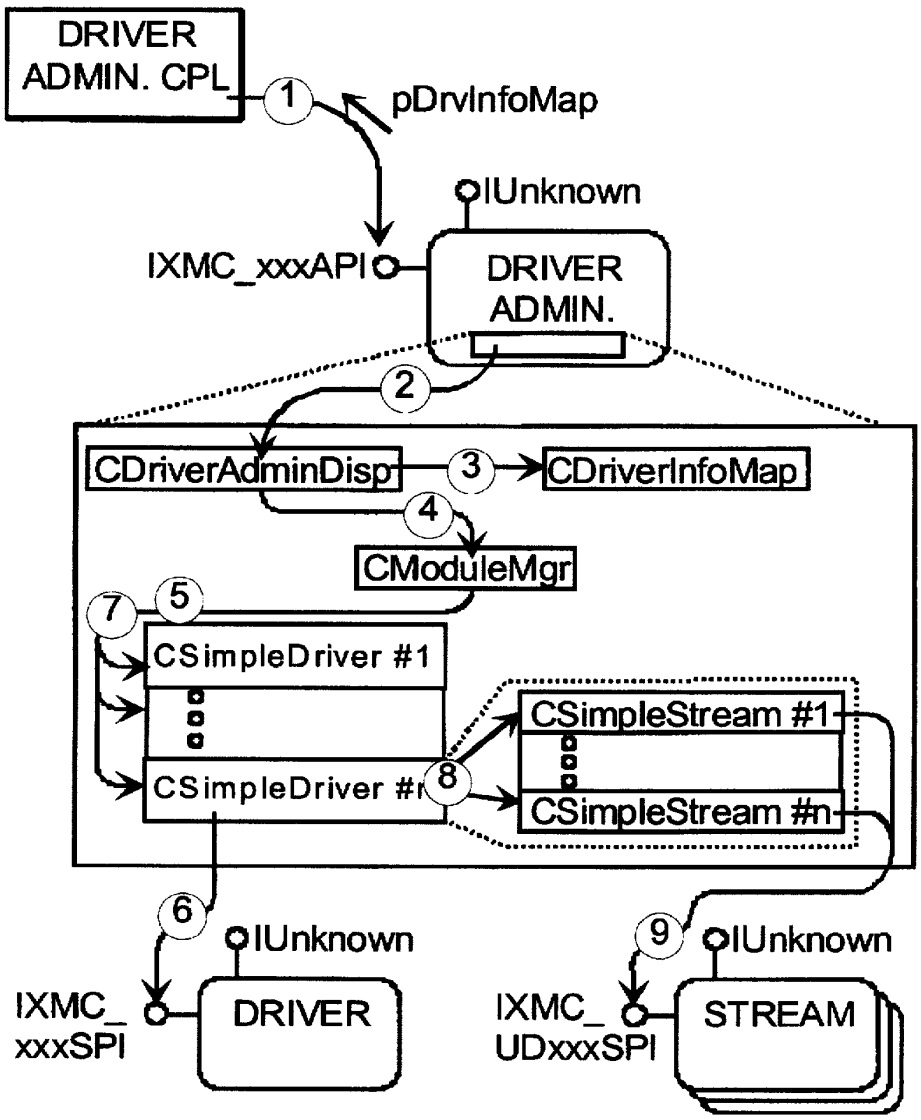

DRIVER ADMIN. CPL
1
pDrvInfoMap
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
2
CDriverAdminDisp
3
CDriverInfoMap
4
CModuleMgr
7
5
CSimpleDriver #1
CSimpleDriver #n
8
CSimpleStream #1
CSimpleStream #n
6
IUnknown
IXMC_ xxxSPI
DRIVER
9
IUnknown
IXMC_ UDxxxSPI
STREAM

FIG. 49

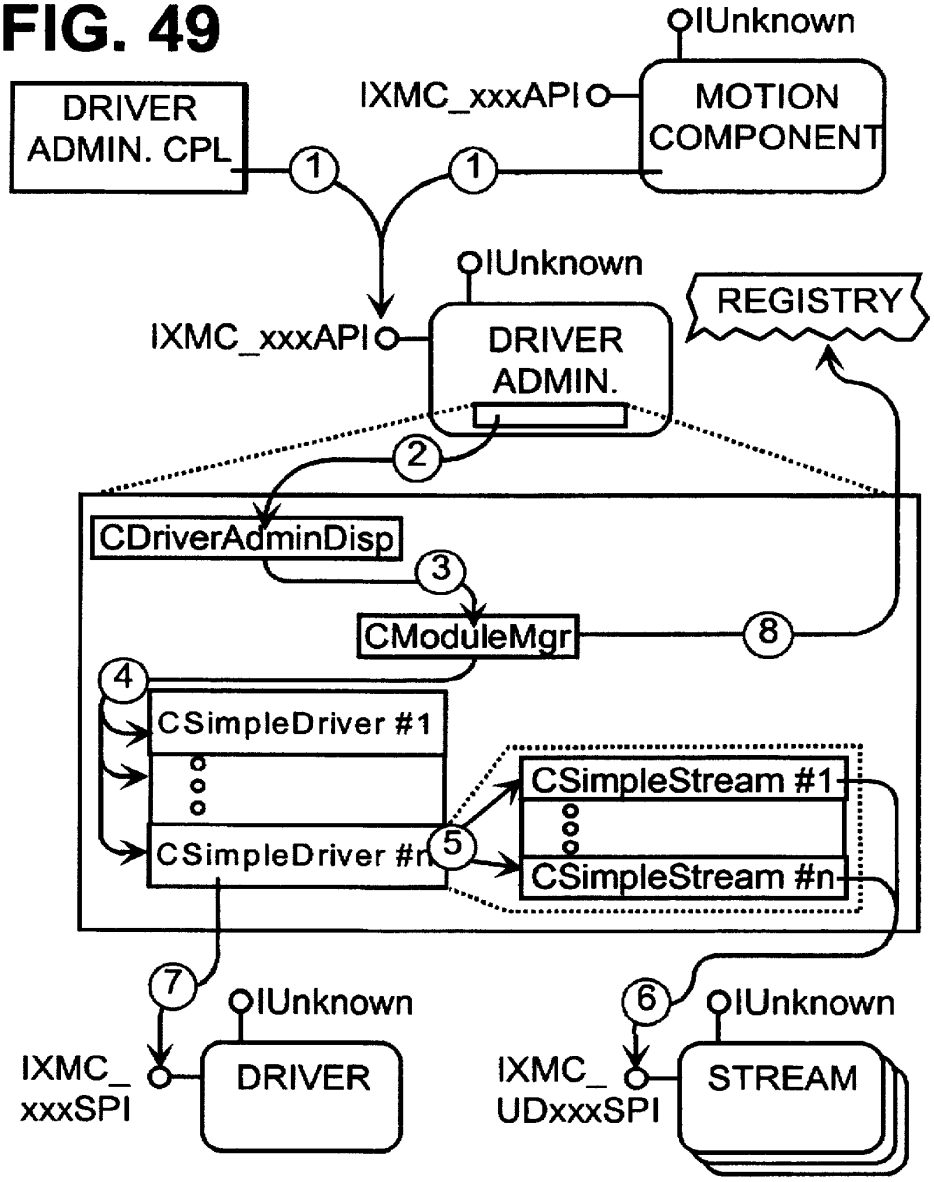

IUnknown
DRIVER ADMIN. CPL
IXMC_xxxAPI
MOTION COMPONENT
1
1
IUnknown
REGISTRY
IXMC_xxxAPI
DRIVER ADMIN.
2
CDriverAdminDisp
3
CModuleMgr
8
4
CSimpleDriver #1
CSimpleStream #1
CSimpleDriver #n
5
CSimpleStream #n
7
6
IUnknown
IUnknown
IXMC_ xxxSPI
DRIVER
IXMC_ UDxxxSPI
STREAM

FIG. 50

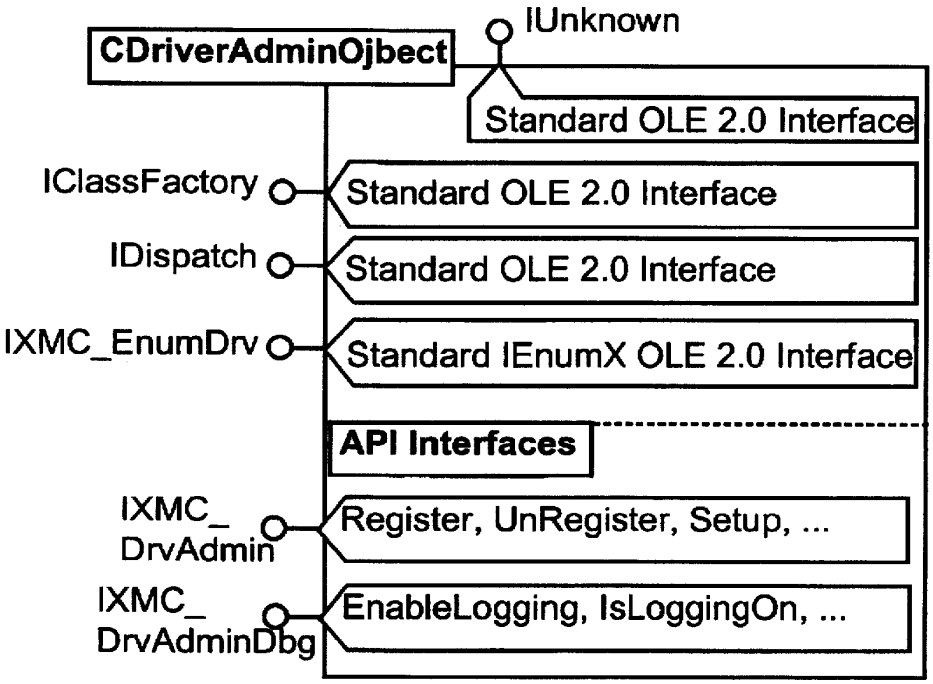

IUnknown
CDriverAdminOjbect
Standard OLE 2.0 Interface
IClassFactory
Standard OLE 2.0 Interface
IDispatch
Standard OLE 2.0 Interface
IXMC_EnumDrv
Standard IEnumX OLE 2.0 Interface
API Interfaces
IXMC_ DrvAdmin
Register, UnRegister, Setup, ...
IXMC_ DrvAdminDbg
EnableLogging, IsLoggingOn, ...

FIG. 51

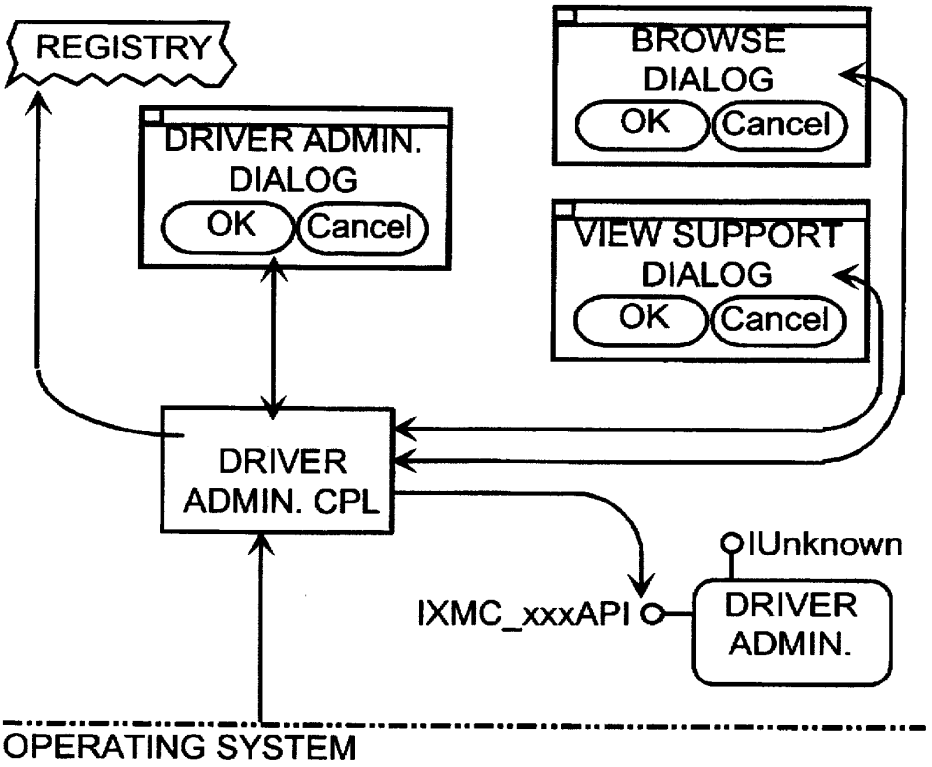

REGISTRY
DRIVER ADMIN. DIALOG
OK
Cancel
BROWSE DIALOG
OK
Cancel
VIEW SUPPORT DIALOG
OK
Cancel
DRIVER ADMIN. CPL
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
OPERATING SYSTEM

FIG. 52

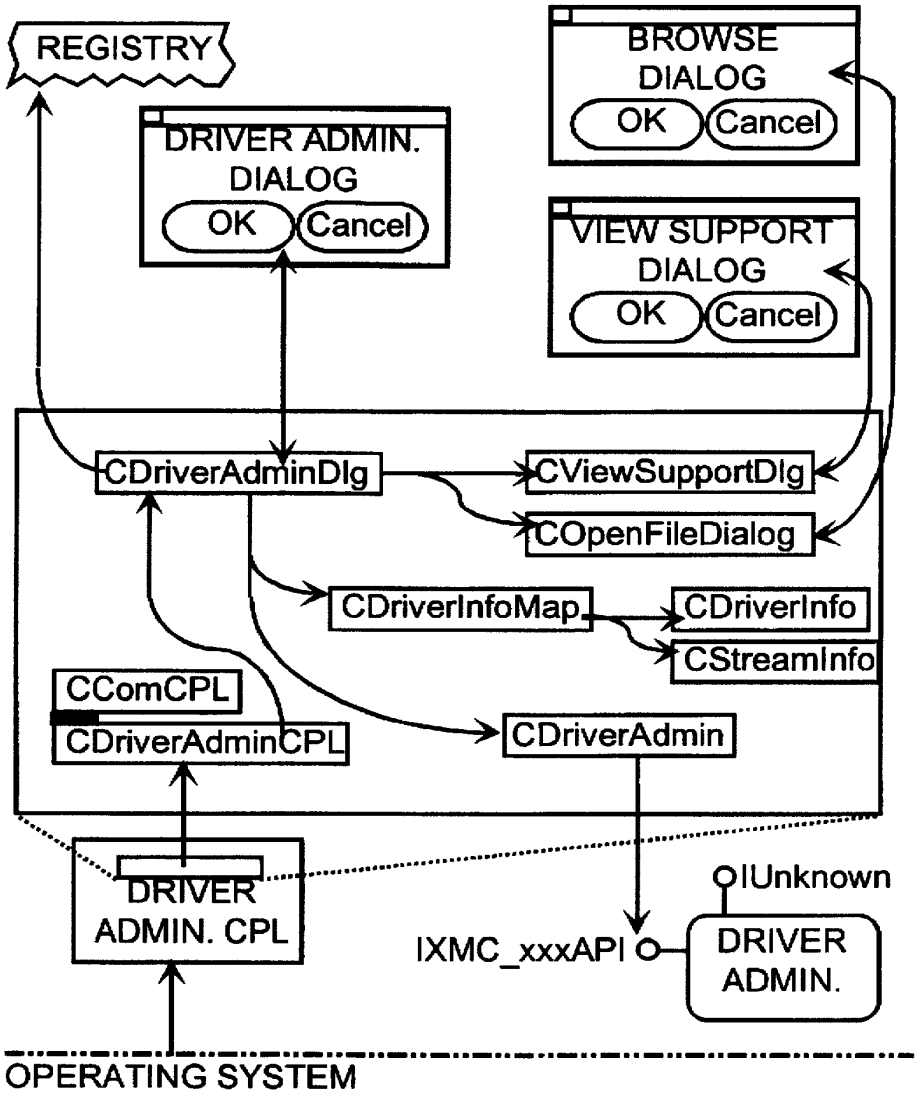

REGISTRY
BROWSE DIALOG
OK
Cancel
DRIVER ADMIN. DIALOG
OK
Cancel
VIEW SUPPORT DIALOG
OK
Cancel
CDriverAdminDlg
CViewSupportDlg
COpenFileDialog
CDriverInfoMap
CDriverInfo
CStreamInfo
CComCPL
CDriverAdminCPL
CDriverAdmin
DRIVER ADMIN. CPL
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.
OPERATING SYSTEM

FIG. 53

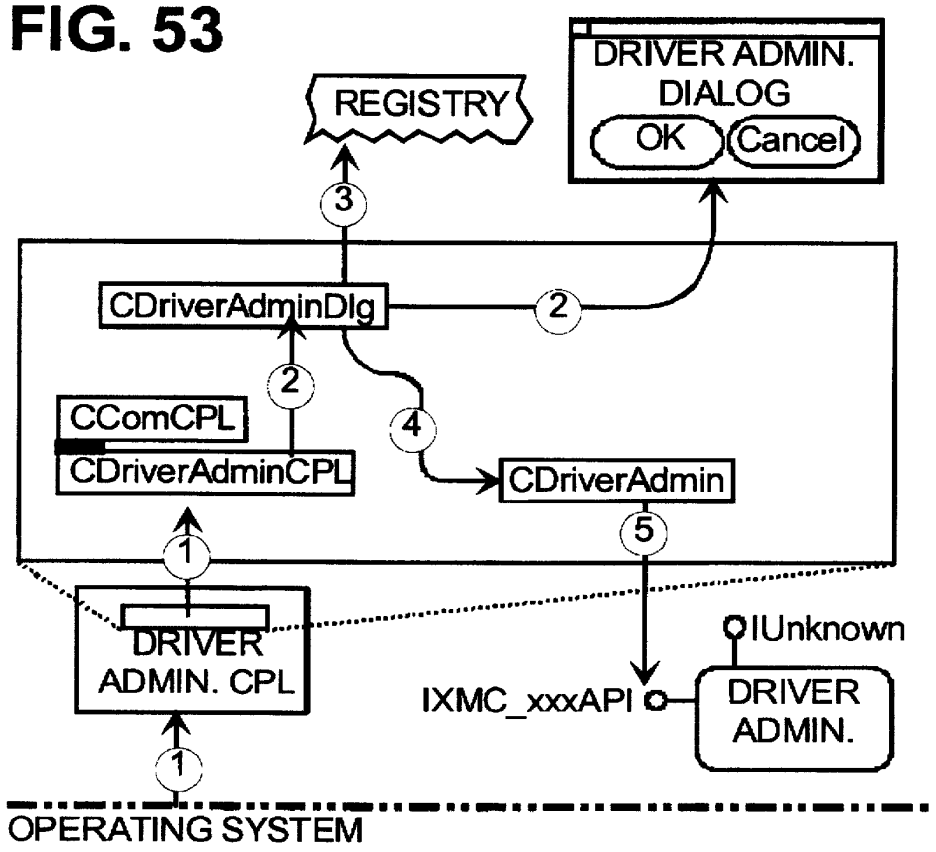

DRIVER ADMIN.
DIALOG
OK
Cancel
REGISTRY
3
CDriverAdminDlg
2
2
CComCPL
CDriverAdminCPL
4
CDriverAdmin
5
1
DRIVER
ADMIN. CPL
IUnknown
IXMC_xxxAPI
DRIVER
ADMIN.
1
OPERATING SYSTEM

FIG. 54

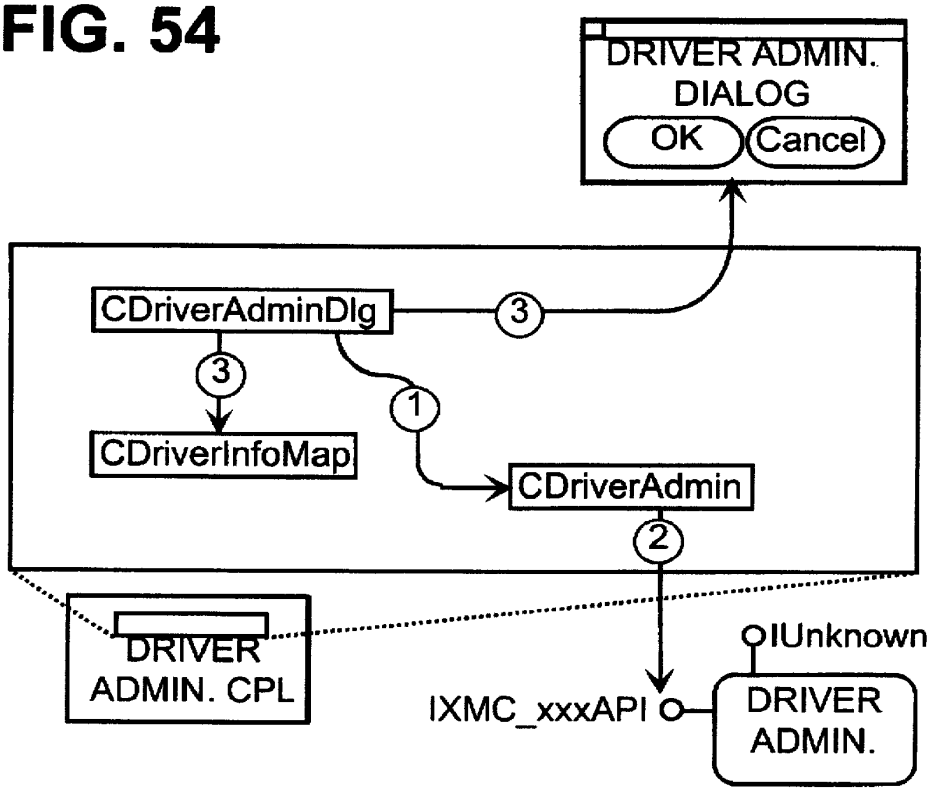

DRIVER ADMIN.
DIALOG
OK
Cancel
CDriverAdminDlg
3
3
1
CDriverInfoMap
CDriverAdmin
2
DRIVER
ADMIN. CPL
IUnknown
IXMC_xxxAPI
DRIVER
ADMIN.

FIG. 55

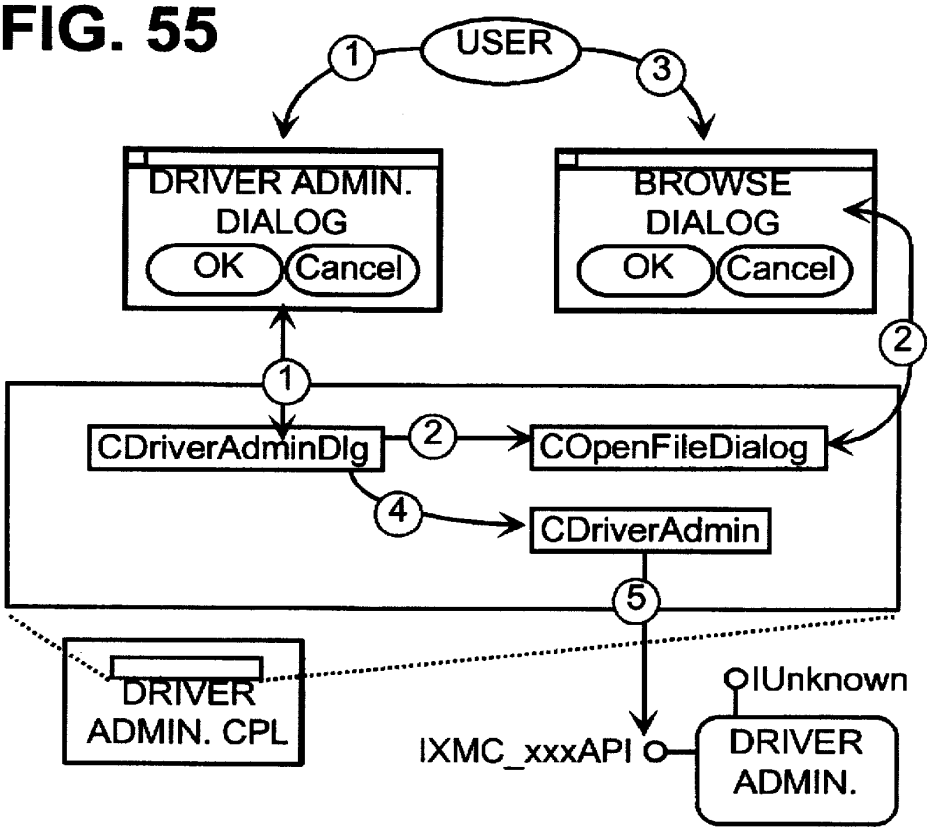

USER
1
3
DRIVER ADMIN. DIALOG
OK
Cancel
BROWSE DIALOG
OK
Cancel
2
1
CDriverAdminDlg
2
COpenFileDialog
4
CDriverAdmin
5
DRIVER ADMIN. CPL
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.

FIG. 56

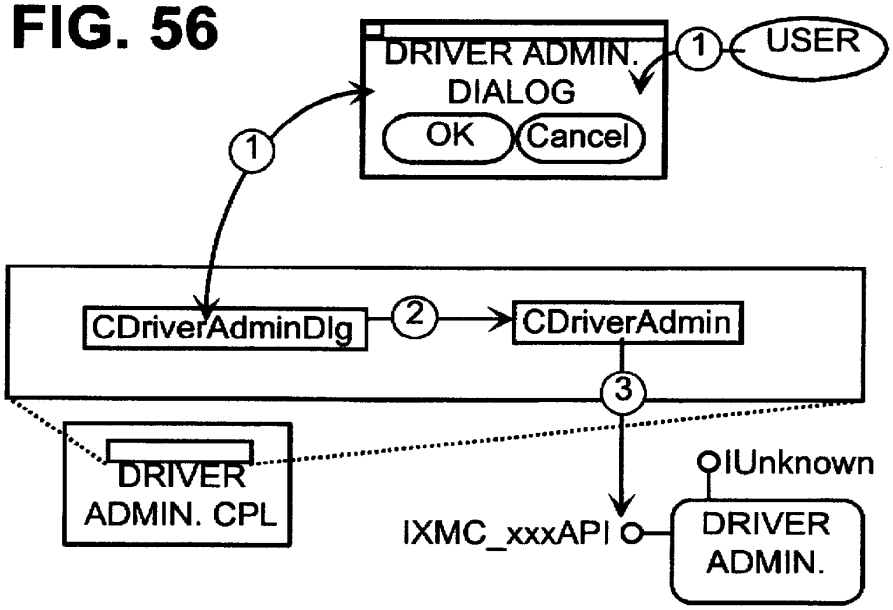
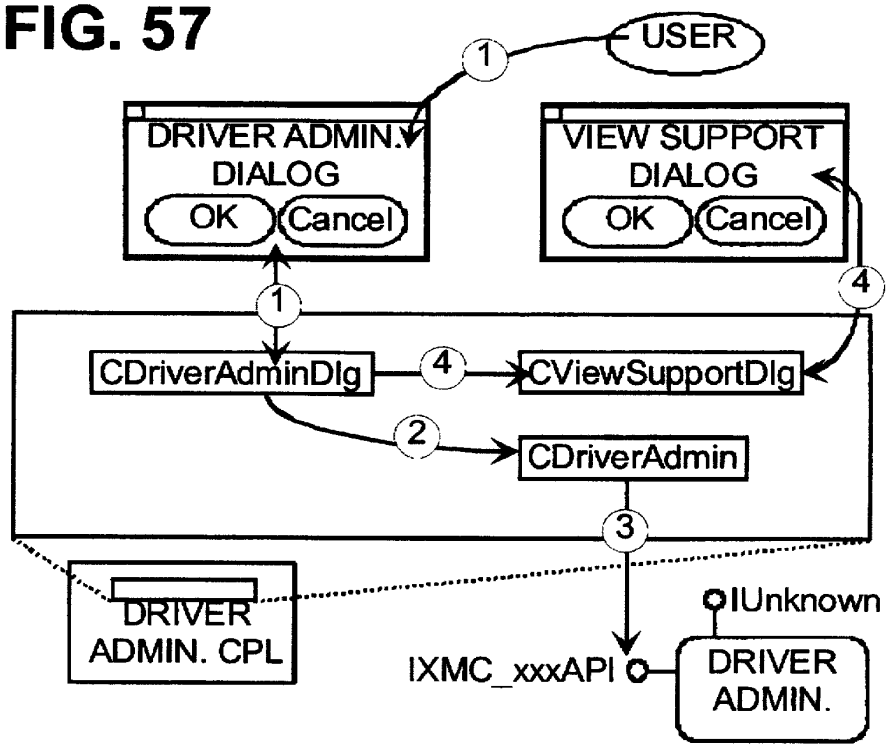

DRIVER ADMIN. DIALOG
OK
Cancel
USER
1
CDriverAdminDlg
2
CDriverAdmin
3
DRIVER ADMIN. CPL
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.

FIG. 57


USER
1
DRIVER ADMIN. DIALOG
OK
Cancel
VIEW SUPPORT DIALOG
OK
Cancel
4
CDriverAdminDlg
CViewSupportDlg
2
CDriverAdmin
3
DRIVER ADMIN. CPL
IUnknown
IXMC_xxxAPI
DRIVER ADMIN.

MOTION CONTROL SYSTEMS

TECHNICAL FIELD

The present invention relates to motion control systems and, more particularly, to interface software that facilitates the creation of hardware independent motion control software.

BACKGROUND OF THE INVENTION

The purpose of a motion control device is to move an object in a desired manner. The basic components of a motion control device are a controller and a mechanical system. The mechanical system translates signals generated by the controller into movement of an object.

While the mechanical system commonly comprises a drive and an electrical motor, a number of other systems, such as hydraulic or vibrational systems, can be used to cause movement of an object based on a control signal. Additionally, it is possible for a motion control device to comprise a plurality of drives and motors to allow multi-axis control of the movement of the object.

The present invention is of particular importance in the context of a mechanical system including at least one drive and electrical motor having a rotating shaft connected in some way to the object to be moved, and that application will be described in detail herein. But the principles of the present invention are generally applicable to any mechanical system that generates movement based on a control signal. The scope of the present invention should thus be determined based on the claims appended hereto and not the following detailed description.

In a mechanical system comprising a controller, a drive, and an electrical motor, the motor is physically connected to the object to be moved such that rotation of the motor shaft is translated into movement of the object. The drive is an electronic power amplifier adapted to provide power to a motor to rotate the motor shaft in a controlled manner. Based on control commands, the controller controls the drive in a predictable manner such that the object is moved in the desired manner.

These basic components are normally placed into a larger system to accomplish a specific task. For example, one controller may operate in conjunction with several drives and motors in a multi-axis system for moving a tool along a predetermined path relative to a workpiece.

Additionally, the basic components described above are often used in conjunction with a host computer or programmable logic controller (PLC). The host computer or PLC allows the use of a high-level programming language to generate control commands that are passed to the controller. Software running on the host computer is thus designed to simplify the task of programming the controller.

Companies that manufacture motion control devices are, traditionally, hardware oriented companies that manufacture software dedicated to the hardware that they manufacture. These software products may be referred to as low level programs. Low level programs usually work directly with the motion control command language specific to a given motion control device. While such low level programs offer the programmer substantially complete control over the hardware, these programs are highly hardware dependent.

In contrast to low-level programs, high-level software programs, referred to sometimes as factory automation applications, allow a factory system designer to develop application programs that combine large numbers of input/output (I/O) devices, including motion control devices, into a complex system used to automate a factory floor environment. These factory automation applications allow any number of i/O devices to be used in a given system, as long as these devices are supported by the high-level program. Custom applications, developed by other software developers, cannot be developed to take advantage of the simple motion control functionality offered by the factory automation program.

Additionally, these programs do not allow the programmer a great degree of control over the each motion control device in the system. Each program developed with a factory automation application must run within the context of that application.

PRIOR ART

In the following discussions, a number of documents are cited that are publicly available as of the filing date of the present invention. With many of these documents, the Applicant is not aware of exact publishing dates. The citation of these documents should thus not be considered an admission that they are prior art; the Applicant will take the steps necessary to establish whether these documents are prior art if necessary.

As mentioned above, a number of software programs currently exist for programming individual motion control devices or for aiding in the development of systems containing a number of motion control devices.

The following is a list of documents disclosing presently commercially available high-level software programs: (a) Software Products For Industrial Automation, iconics 1993; (b) The complete, computer-based automation tool (IGSS), Seven Technologies A/S; (c) OpenBatch Product Brief, PID, Inc.; (d) FIX Product Brochure, Intellution (1994); (e) Paragon TNT Product Brochure, Intec Controls Corp.; (f) WEB 3.0 Product Brochure, Trihedral Engineering Ltd. (1994); and (g) AIMAX-WIN Product Brochure, TA Engineering Co., Inc. The following documents disclose simulation software: (a) ExperTune PID Tuning Software, Gerry Engineering Software; and (b) XANALOG Model NL-SIM Product Brochure, XANALOG.

The following list identifies documents related to low-level programs: (a) Compumotor Digiplan 1993–94 catalog, pages 10–11; (b) Aerotech Motion Control Product Guide, pages 233–34; (c) PMAC Product Catalog, page 43; (d) PC/DS-Series Motion Controller C Programming Guide, pages 1–3; (e) Oregon Micro Systems Product Guide, page 17; (f) Precision Microcontrol Product Guide.

The Applicants are also aware of a software model referred to as WOSA that has been defined by Microsoft for use in the Windows programming environment. The WOSA model is discussed in the book Inside Windows 95, on pages 348–351. WOSA is also discussed in the paper entitled WOSA Backgrounder: Delivering Enterprise Services to the Windows-based Desktop. The WOSA model isolates application programmers from the complexities of programming to different service providers by providing an API layer that is independent of an underlying hardware or service and an SPI layer that is hardware independent but service dependent. The WOSA model has no relation to motion control devices.

The Applicants are also aware of the common programming practice in which drivers are provided for hardware such as printers or the like; an application program such as a word processor allows a user to select a driver associated with a given printer to allow the application program to print on that given printer.

While this approach does isolates the application programmer from the complexities of programming to each hardware configuration in existence, this approach does not provide the application programmer with the ability to control the hardware in base incremental steps. In the printer example, an application programmer will not be able to control each stepper motor in the printer using the provided printer driver; instead, the printer driver will control a number of stepper motors in the printer in a predetermined sequence as necessary to implement a group of high level commands.

The software driver model currently used for printers and the like is thus not applicable to the development of a sequence of control commands for motion control devices.

OBJECTS OF THE INVENTION

From the foregoing, it should be clear that one primary object of the invention is to provide improved methods and devices for moving objects.

Another more specific object of the present invention is to obtain methods and apparatus for designing and deploying motion control devices in which these methods and apparatus exhibit a favorable mix of the following characteristics:

(a) allow the creation of high-level motion control programs that are hardware independent, but offer programmability of base motion operations;

(b) hide the complexities of programming for multiple hardware configurations from the high-level programmer;

(c) can easily be extended to support additional hardware configurations; and

(c) transparently supports industry standard high-level programming environments.

SUMMARY OF THE INVENTION

The present invention is, in one form, a method of moving an object comprising the steps of developing a high-level motion control application program comprising a sequence of component functions that describe a desired object path, correlating these component functions with driver functions, selecting a software driver for the specific hardware configuration being controlled, generating control commands from the driver functions and the software driver associated with the hardware configuration being controlled, and controlling a motion control device based on the control data to move the object along the desired object path.

In another form, the present invention is a method of generating a sequence of control commands for controlling a motion control devices to move an object along a desired path. An application program comprising a series of component functions defines a sequence of motion steps that must be performed by the motion control device to move the object along the desired path. The component functions contain code that relates the component functions to driver functions. The driver functions are associated with or contain software drivers containing driver code for implementing the motion steps on a given motion control device. The control commands are generated based on the application program and the driver code associated with a given motion control device.

The use of component functions that are separate from driver functions isolates the programmer from the complexities of programming to a specific motion control device. This arrangement also allows a given application program to be used without modification for any motion control device having a software driver associated therewith.

The driver functions are grouped into core driver functions and extended driver functions. All software drivers must support the core driver functions; the software drivers may also support one or more of the extended driver functions, although this is not required.

Where the software drivers do not support the extended driver functions, the functionality associated with the extended driver functions can normally be simulated using some combination of core driver functions. In this case, the method of the present invention comprises the steps of determining which of the extended driver functions are not supported by the software driver and, where possible, substituting a combination of core driver functions. In some cases, the functionality of an extended driver function cannot be emulated using core driver functions, and this functionality is simply unavailable to the programmer.

The use of core driver functions to emulate extended driver functions provides functionality where none would otherwise exist, but the preferred approach is to provide a software driver that supports each of the extended driver functions. When an extended driver function is supported and not emulated, the task being performed will normally be accomplished more quickly and accurately.

Additionally, to simplify the use of emulated extended driver functions, the method of the present invention further comprises the steps of determining which, if any, extended driver functions are not supported by the software driver for a given hardware configuration, developing a function pointer table of both unsupported extended driver functions and supported extended driver functions, and consulting the table each time an extended driver function is called to determine whether that extended driver function must be emulated. In this manner, the process of calling the sequence of core driver functions employed to emulate the unsupported extended driver functions is optimized.

As the control commands are generated as described above, they may be used to control a motion control device in real time or they may be stored in a file for later use. Preferably, the method of the present invention comprises the step of providing a number of streams containing stream code. Each stream is associated with a destination of control commands, and the stream code of a given stream dictates how the control commands are to be transferred to the destination associated with that given stream. The user is thus provided the opportunity to select one or more streams that dictate the destination of the control commands.

To help isolate the programmer from hardware specific complexities, the method of the present invention may comprise the additional administrative steps such as selecting a driver associated with a particular motion control device and/or translating units required to define the motion control system into the particular system of units employed by a given motion control device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a system interaction map of a motion control system constructed in accordance with, and embodying, the principles of the present invention;

FIG. **2** is a module interaction map of a motion control component of the system shown in FIG. **1**;

FIG. **3** is an object interaction map of the component shown in FIG. **2**;

FIGS. **4**, **5**, **6**, **7** and **8** are scenario maps of the component shown in FIG. **2**;

FIG. **9** is an interface map of the component shown in FIG. **2**;

FIG. **10** is a data map showing one exemplary method of accessing the data necessary to emulate extended driver functions using core driver functions;

FIG. **11** is a module interaction map of the driver portion of the system shown in FIG. **1**;

FIG. **12** is an object interaction map of the driver portion shown in FIG. **11**;

FIGS. **13**, **14**, **15**, **16**, **17**, **18**, **19** and **20** are scenario maps related to the driver shown in FIG. **11**;

FIG. **21** is an interface map for the driver shown in FIG. **11**;

FIG. **22** is a module interaction map of the streams used by the system shown in FIG. **1**;

FIG. **23** is an object interaction map of the streams shown in FIG. **22**;

FIGS. **24**, **25**, **26**, **27**, **28**, **29**, **30**, **31** and **32** are scenario maps of the streams shown in FIG. **22**;

FIG. **33** is an interface map of the objects comprising the stream shown in FIG. **22**;

FIG. **34** is a module interaction map of the driver stub portion of the system shown in FIG. **1**;

FIG. **35** is an object interaction map of the driver stub shown in FIG. **34**;

FIGS. **36**, **37** and **38** are scenario maps of the driver stub shown in FIG. **34**;

FIG. **39** is an interface map of the driver stub portion shown in FIG.

FIG. **40** is a module interaction map of the driver administrator portion of the system shown in FIG. **1**;

FIG. **41** is an object interaction map of the driver administrator shown in FIG.

FIGS. **42**, **43**, **44**, **45**, **46**, **47**, **48** and **49** are scenario maps relating to the driver administrator shown in FIG. **40**;

FIG. **50** is an interface map of the objects that comprise the driver administrator shown in FIG. **40**;

FIG. **51** is a module interaction map of the driver administrator CPL applet portion of the system shown in FIG. **1**;

FIG. **52** is an object interaction map of the driver administrator CPL applet shown in FIG. **51**;

FIGS. **53**, **54**, **55**, **56** and **57** are scenario maps related to the driver administrator CPL applet shown in FIG. **51**.

DETAILED DESCRIPTION OF THE INVENTION

Referring now to the drawing, depicted therein at **10** in FIG. **1** is a motion control system constructed in accordance with, and embodying, the principles of the present invention. This system **10** comprises a personal computer portion **12** having a hardware bus **14**, a plurality of motion control hardware controllers **16***a*, **16***b*, and **16***c*, and mechanical systems **18***a*, **18***b*, and **18***c* that interact with one or more objects (not shown) to be moved.

The personal computer portion **12** of the system **10** can be any system capable of being programmed as described herein, but, in the preferred embodiment, is a system capable of running the Microsoft Windows environment. Such a system will normally comprise a serial port in addition to the hardware bus **14** shown in FIG. **1**.

The hardware bus **14** provides the physical connections necessary for the computer **12** to communicate with the hardware controllers **16**. The hardware controllers **16** control the mechanical system **18** to move in a predictable manner. The mechanical system **18** comprises a motor or the like the output shaft of which is coupled to the object to be moved. The combination of the hardware controllers **16***a*, **16***b*, and **16***c* and the mechanical systems **18***a*, **18***b*, and **18***c* forms motion control devices **20***a*, **20***b*, and **20***c*, respectively.

The hardware bus **14**, hardware controllers **16**, and mechanical systems **18** are all well-known in the art and are discussed herein only to the extent necessary to provide a complete understanding of the present invention.

The personal computer portion **12** contains a software system **22** that allows an application user **24** to create software applications **26** that control the motion control devices **20**.

More particularly, based on data input by the user **24** and the contents of the application program **26**, the software system **22** generates control commands that are transmitted by one or more streams such as those indicated at **28***a*, **28***b*, **28***c*, and **28***d*. The streams **28** transmit control commands incorporating the hardware specific command language necessary to control a given motion control device to perform in a desired manner. As will be discussed in more detail below, the streams **28** implement the communication protocol that allows the control commands to reach the appropriate motion control device **28** via an appropriate channel (i.e., PC bus, serial port).

Using the system **22**, the application program **26** is developed such that it contains no code that is specific to any one of the exemplary hardware controllers **16**. In the normal case, the application program **26**, and thus the user **24** that created the program **26**, is completely isolated from the motion control devices **20**. The user **24** thus need know nothing about the hardware specific command language or communication protocol associated with each of these devices **20**; it may even be possible that the command language of one or more of the hardware controllers **16** was not defined at the time the application program **26** was created.

The software system **22** comprises a combination of elements that allow the application program **26** to be completely isolated from the hardware controllers **16**. In the following discussion, the framework of the software system **22** will be described in terms of a method of moving an object and/or a method of generating control commands. After this general discussion, each component of the system **22** will be described in detail in a specific operating environment.

I. Method of Generating Control Commands for Controlling a Motion Control Device to Move an Object

Initially, it should be noted that, in most situations, the method described in this section will normally but not necessarily involve the labors of at least two and perhaps three separate software programmers: a software system designer; a hardware designer familiar with the intricacies of the motion control device; and a motion control system designer. The application user **24** discussed above will normally be the motion control system designer, and the roles of the software system designer and hardware designer will become apparent from the following discussion.

The software system designer develops the software system **22**. The software system designer initially defines a set of motion control operations that are used to perform motion control. The motion control operations are not specifically related to any particular motion control device hardware configuration, but are instead abstract operations that all

motion control device hardware configurations must perform in order to function.

Motion control operations may either be primitive operations or non-primitive operations. Primitive operations are operations that are necessary for motion control and cannot be simulated using a combination of other motion control operations. Examples of primitive operations include GET POSITION and MOVE RELATIVE, which are necessary for motion control and cannot be emulated using other motion control operations. Non-primitive operations are motion control operations that do not meet the definition of a primitive operations. Examples of non-primitive operations include CONTOUR MOVE, which may be emulated using a combination of primitive motion control operations.

Given the set of motion control operations as defined above, the software system designer next defines a service provider interface (SPI) comprising a number of driver functions. Driver functions may be either core driver functions or extended driver functions. Core driver functions are associated with primitive operations, while extended driver functions are associated with non-primitive operations. As with motion control operations, driver functions are not related to a specific hardware configuration; basically, the driver functions define parameters necessary to implement motion control operations in a generic sense, but do not attach specific values or the like to these parameters. The SPI for the exemplary software system **22** is attached hereto as Appendix A.

The software system designer next defines an application programming interface (API) comprising a set of component functions. For these component functions, the software system designer writes component code that associates at least some of the component functions with at least some of the driver functions. The relationship between component functions and driver functions need not be one to one: for example, certain component functions are provided for administrative purposes and do not have a corresponding driver function. However, most component functions will have an associated driver function. The API for the exemplary software system **22** is attached hereto as Appendix B.

The overall software model implemented by the software program **22** thus contains an API comprising component functions and an SPI comprising driver functions, with the API being related to the SPI by component code associated with the component functions.

In order for the system **22** to generate the control commands, at least two more components are needed: the application program **26** and at least one software driver such as the drivers indicated at **30***a*, **30***b*, and **30***c* in FIG. **1**.

The software drivers **30** are normally developed by a hardware designer and are each associated with a single motion control device. The hardware designer writes driver code that dictates how to generate control commands for controlling the motion control device associated therewith to perform the motion control operations associated with at least some of the driver functions.

In the exemplary software system **22**, the software drivers **30***a*, **30***b*, and **30***c* are associated with the motion control devices **20***a*, **20***b*, and **20***c*, respectively. As a software driver exists for each of the motion control devices **20***a*, **20***b*, and **20***c*, these devices **20***a*, **20***b*, and **20***c* form a group of supported motion control devices.

A careful review of the framework of the software system **22** as described above will illustrate that, of all the components of this system **22**, only the software drivers **30** are hardware dependent.

The motion control system designer, normally also the user **24**, develops the application program **26**. The application program **26** comprises a sequence of component functions arranged to define the motion control operations necessary to control a motion control device to move an object in a desired manner. The application program **26** is any application that uses the system **22** by programming the motion control component **34**. Applications may program the system **22** either through OLE Automation or by using any of the custom OLE interfaces making up the API.

As mentioned above, the component code associates many of the component functions with the driver functions, and the driver functions define the parameters necessary to carry out the motion control operations. Thus, with appropriately ordered component functions, the application program **26** contains the logic necessary to move the object in the desired manner.

Once the application program **26** has been written and the software drivers **30** have been provided, the user **24** selects at least one motion control device from the group of supported motion control devices **20***a*, **20***b*, and **20***c*. Using a driver administrator module **32**, the user **24** then selects the software driver associated with the selected motion control device. This driver administrator module **32** is used to install, uninstall, register, and setup each stream.

As currently implemented, the driver administrator **32** allows only one software driver to be selected. In future versions of the software system **22**, the driver administrator will allow the user to select one or more software drivers.

The software system **22** thus generates control commands based on the component functions contained in the application program **26**, the component code associated with the component **34** functions, and the driver code associated with the selected software driver **28**.

As the control commands are being generated as described above, they may be directly transmitted to a motion control device to control this device in real time or stored in an output file for later use. The software system **22** employs the streams **28** to handle the transmission of the control commands to a desired destination thereof.

In the exemplary system **22**, the destinations of the control commands may be one or more of an output file **34** and/or the controllers **16**. Other possible destinations include a debug monitor or window or other custom output mechanism defined for a specific situation. The software system designer, or in some cases the hardware system designer, will write transmit stream code for each stream **28** that determines how the control commands are to be transferred to a given one of the control command destinations **16** and **34**. Using the driver administrator **32**, the user **24** selects one or more of the control command destinations **16** and **34**, and, later when run, the system **22** transfers the control commands to the selected control command destination **16** and/or **34** based on the transmit stream code in the stream **28** associated with the selected control command destination **16** and/or **34**.

Many control command destinations such as **16** and **34** are capable of transmitting data back to the system **22**. Data transmitted from a control command destination back to the system **22** will be referred to as response data. The software system designer thus further writes data response stream code for each of the streams **28***a*, **28***b*, and **28***c* that determines how response data is transmitted from the controllers **16** to the system **22**. The system **22** thus processes the response data sent by the controllers **16** based on the data response stream code contained in the streams **28**.

Referring again to FIG. 1, this Figure shows that the system **22** further comprises a motion control component **35** and a driver stub module **36**. The motion control component module **35** is the portion of the software system **22** that relates the component functions to the driver functions. The motion control component module **35** thus contains the component code that makes the association between the component functions contained in the application program **26** and the driver functions.

The driver stub module **36** is not required to implement the basic software model implemented by the system **22**, but provides the system **22** with significantly greater flexibility to accommodate diverse motion control hardware configurations with minimal effort.

More particularly, when the driver stub module **36** is employed, the hardware designer need not develop driver code to implement all of the driver functions; to the contrary, the hardware designer must write driver code for implementing the core driver functions but need not write driver code to implement the extended driver functions. The software system designer provides the motion control driver stub **36** with stub code that identifies the combinations of core driver functions that are employed to emulate the functionality of extended driver functions.

The motion control component **24** will determine for the selected software driver **30** which extended functions, if any, the selected driver **30** supports. For extended functions that are not supported, referred to herein as non-supported extended driver functions, the motion control component **35** refers to the driver stub module **36** to determine the appropriate combination of core driver functions to emulate the functionality of the non-supported extended driver functions. The system **22** thus generates the control commands necessary to implement the non-supported extended driver functions using the appropriate combination of core driver functions.

The process of determining when extended driver functions need to be emulated can be optimized by providing the motion control component **35** with a function pointer table that contains a pointer to each of extended functions. When building the function pointer table, the motion control component **35** checks the selected driver module **30** to see if it supports each extended function. If the selected driver module **30** supports the extended function, the motion control component module **35** stores a pointer to the function, implemented by the selected driver module **30**, in the table location corresponding to the extended function. In the event that the selected driver module **30** does not support the extended function, the motion control component module **35** stores a pointer to the extended function implementation located in the driver stub module **36**. The driver stub module **36** implementation of the extended function contains calls to a plurality of core functions implemented by the selected driver module **30**.

Therefore, the driver stub module **36** allows the motion control system designer to use, with minimal time and effort by the hardware designer, a working software driver **28** that contains driver code to implement only the core functions. The software driver **28** developed to implement the core driver functions can then be improved by developing driver code to implement extended driver functions as desired.

The use of driver code specifically designed to implement extended driver functions is, in general, preferable to relying on the driver stub module **36** to emulate the extended driver functions; driver code specifically written to implement an extended driver function will almost always obtain a more optimized implementation of the driver function than the emulation of that driver function with a combination of core driver functions.

Referring again for a moment to FIG. 1, this Figure illustrates that the system **22** additionally comprises a driver administrator CPL applet **38** and a DDE server **40**. The driver administration CPL applet **38** generates the user interface through which the user **24** communicates with the driver administrator module **32**. The DDE server **40** provides the software interface through which the application program **26** communicates with the motion control component module **35**.

II. MOTION CONTROL COMPONENT

The motion control component **35** will now be described in further detail with reference to FIGS. **2–10**. The motion control component **35** is used by every application programming the system **22** to perform motion control operations. The major set of the API is implemented by this component. When operating, the motion control component **35** interacts with the driver administrator **32**, to get the current driver, and the driver **30** and driver stub **36**, to carry out motion control operations. Applications, using system **22**, only interact with the motion control component **35**.

This section describes the design of the motion control component **35** in three main parts. First, all binary modules that affect the component **35** are described along with their interactions with the component **35**. Next, the module interaction-map is drawn in more detail to show the interactions between all C++ objects used to implement the motion control component **35**. Next, the object interaction-map is tested by displaying the specific interactions that take place during certain, key process that the component **34** is requested to perform.

The module interaction-map shown in FIG. **2** displays all binary modules and their interactions with the motion control component **35**. As can be seen from the module interaction-map, applications only communicate with the motion control component **35**. From this point, the component **35** coordinates all interactions between the driver administrator **32**, driver **30**, and driver stub **36** components.

Breaking the module interaction-map and adding the interactions taking place between all C++ objects used to implement the motion control component **35**, produces the object interaction-map shown in FIG. **3**.

Each object in the diagram is described as follows. The CCmpntDisp object is the dispatch object used to dispatch exposed interface methods. During the dispatch process, all raw data is converted into the appropriate C++ form. For example, collections of data passed between OLE components is usually packaged in a raw block of memory. The CCmpntDisp object takes care of packing outgoing data and unpacking incoming data. Data packing involves converting the data between a raw and native C++ format.

The CDriverAdmin object is used to communicate directly with the driver administrator component. All OLE related details are encapsulated within this class.

The CDriverMgr object is used to control all unit mapping taking place before calling the appropriate SPI function. The CUnitMapper object is used to do the actual mapping between units.

The CUnitMapper object is used to map units between the Part Coordinate System (PCS) and the Machine Coordinate System (MCS). Both directions of unit mapping are done by this object.

The CDriver object is used to build the SPI table containing both core and extended SPI functions. Depending on the level of driver support, the extended functions in the SPI table may point to functions implemented in either the driver stub **36** or the driver **30**.

The following discussion of FIGS describes all main scenarios, or operations, that occur on the motion control component **35**. Each scenario-map displays all objects involved, and the interactions that take place between them in the sequence that they occur.

As shown in FIG. **4**, before an application can use the motion control component **35**, it must create an instance of the object, using the CoCreateinstance OLE function, and then initialize the instance calling the exposed Initialize custom interface method implemented by the component **35**. FIG. **4** displays the sequence of events that take place when the Initialize method is called.

During initialization, the following steps occur. First the application must create an instance of the motion control component **34** by calling the standard OLE function CoCreateInstance. Once loaded, the application must call the component **35**'s exposed Initialize method. When first loaded, the component **35** loads any registration data previously stored. Next, the component **35** directs the CCmpntDisp to initialize the system. The CCmpntDisp directs the CDriverAdmin to get the current driver(s) to use. The CDriverAclmin, first, loads the driver administrator **32** using the standard OLE CoCreateinstance function. Next, it initializes the driver administrator. Then, it queries the driver administrator for the driver(s) to use and their SPI support information. Finally, the driver administrator returns the driver(s) and the support information to the component **35**, and releases all interfaces used from the driver administrator component **32**.

Once receiving the active driver(s) **30** and their support information, the motion control component **35** passes the driver(s) **30** to the CDriverMgr and directs it to initialize the system. During its initialization, the CDriverMgr initializes the CUnitMapper. Also while initializing, the CDriverMgr initializes a CDriver for each driver used. After initializing each CDriver, the support information is used to build each SPI table inside each CDriver object. When building the SPI table, all core and supported extended SPI interfaces are queried from the driver. Also, when building the SPI table, the CDriver queries all interfaces, not supported by the driver **30**, from the driver stub **36**.

Referring now to FIG. **5**, once the motion control component **35** is initialized, the application **26** may perform operations on it. There are two types of operations that may take place on the component **35**: Operations that use core SPI functions, and operations that use extended SPI functions. Even though the difference between the two is completely invisible to the application using the component **35**, the internal interactions are different between the two. The following discussion outline these differences.

The following interactions take place when the component **35** performs an operation that uses core SPI functions only. First the application must request the operation and pass all pertinent parameters to the component **35**. Next, the component **35** directs the CCmpntDisp to carry out the operation. The CCmpntDisp then directs the CDriverMgr to perform the operation and passes all pertinent parameters to it. Before carrying out the operation, the CDriverMgr uses the CUnitMapper to convert all units to the Machine Coordinate System (MCS). Next, the CDriverMgr directs the CDriver object to carry out the operation and passes the newly mapped parameters to it. The CDriver object uses its internal SPI table to communicate directly with the core SPI function implemented by the driver component.

FIG. **6** shows the sequence of events that occurs when the component **35** is directed to carry out an operation than happens to use extended SPI not supported by the driver **30**. The following steps occur when the operation is requested.

First the application must request the operation and pass all pertinent parameters to the component **35**. Next, the component **35** directs the CCmpntDisp to carry out the operation. The CCmpntDisp then directs the CDriverMgr to perform the operation and passes all pertinent parameters to it. Before carrying out the operation, the CDriverMgr uses the CUnitMapper to convert all units to the Machine Coordinate System (MCS). Next, the CDriverMgr directs the CDriver object to carry out the operation and passes the newly mapped parameters to it. The CDriver object uses its internal SPI table to communicate directly with the core SPI function implemented by the driver component.

As briefly discussed above, when using the system **22**, there are several types of units and two different coordinate systems used. The process of unit mapping involves converting measurements between the Part and Machine coordinate systems. FIG. **7** illustrates this process, and the following steps occur when the operation is requested.

First the application must request the operation and pass all parameters to the component **35**. Note, all parameters are in the PCS. Next, the component **35** directs the CCmpntDisp to carry out the operation. The CCmpntDisp directs the CDriverMgr to carry out the operation and passes the PCS parameters to it. The CDriverMgr takes all measurements and uses the CUnitMapper to convert them to the MCS. The newly mapped parameters are then passed to the Cdriver. The CDriver directs either the driver or the driver stub component to carry out the operation.

When the application is finished using the motion control component **35** it directs the component **35** to free all of its resources by calling its exposed Release method. This process is depicted in FIG. **8**. During the clean-up process, the following steps occur.

First the application must direct the component **35** to release all of its resources by calling its Release method. When invoked, the component **35** passes the call on to the CCmpntDisp object. The CCmpntDisp object directs the CDriverMgr to release any resources it is using. The CDriverMgr directs each CDriver object to release any of its resources, then deletes the CDriver objects. First, the CDriver object releases any interfaces it is using from the driver component. Then, the CDriver object releases any interfaces it is using from the driver stub component.

FIG. **9** is an interface map related to the motion control component **35**. FIG. **10** is a data map showing how data relating to the whether extended driver functions need to be emulated is stored. Attached hereto as Appendix C is a document that describes the actual OLE Interfaces exposed, the definitions of the data structures used when passing data around, and the definitions of each class used internally by the motion control component **35**.

III. SOFTWARE DRIVERS

The driver **30** is used by both the driver administrator **32** and the component **35**. Its main purpose is to implement functionality that generates motion control commands for the specific hardware supported. For example, the AT6400 driver, used to control the Compumotor AT6400 motion control hardware, generates AT**6400** command codes. Dur-

ing the initialization phase of the system **22**, the driver administrator **32** communicates with each driver **30**, allowing the user to add, remove, or change the configuration of the driver. When an application, using the system **22**, is run, the component **35** communicates with the driver **30** directing it to carry out the appropriate motion control operations.

This section describes the complete design of a generic driver **30**. All drivers are designed from the base design described herein. This section is divided into three parts. First, a module interaction-map that describes all binary modules that interact with the driver **30** is discussed. Next, the module interaction-map is drawn as an object interaction-map, where all the internals of the driver are exposed. In this map, all C++ objects, making up the driver, and their interactions are shown. Next, several scenario-maps are drawn. Each scenario-map displays the interactions taking place between the C++ objects involved during a certain process. Finally, this section describes the interfaces exposed by the driver component, all data structures used, and the definitions of each C++ class used.

Referring now to FIG. **11**, the module interaction-map displays all binary modules and their interactions with the driver **30**. There are two modules that interact directly with the driver: the motion control component **35**, and the driver administrator **32**. The driver administrator **32** queries and changes the driver settings and the component **35** directs the driver to carry out motion control operations, such as moving to a certain location in the system. Shown at **42** in FIG. **11** is the standard Windows registration database, referred to herein as the registry.

Breaking the module interaction-map down into more detail by including the interactions taking place between all C++ objects used to implement the driver, produces the object interaction-map. The object interaction-map for the driver **30** is shown in FIG. **12**.

Each object in the diagram is described as follows.

CDriverDisp is the dispatch object used to dispatch exposed interface methods. During the dispatch process, all raw data is converted into the appropriate C++ form. For example, collections of data passed between OLE components is usually packaged in a raw block of memory. The CDriverDisp object takes care of packing outgoing data and unpacking incoming data. Data packing involves converting the data between a raw and native C++ format.

The CStreamMgr object is responsible for managing the set of streams registered with the driver. Streams, may be added, removed, and enabled. Only enabled streams are sent data. The CLSID and enabled status of each stream registered, is stored in the registration database. When communicating to streams, the CStreamMgr is used to send the command string to all enabled streams.

The CCommandMgr object is used to build commands sent to the stream, and extracting responses received from the stream. The CCommandMgr is the controlling object that manages the CResponse, CCommandList, and CStream objects.

The CCommandList object stores the complete list of commands making up the motion control command language. Such commands may be stored as text resources or in a text file.

The CCommand object builds command strings that are then sent to the CStream. Each command built is a complete motion control command string.

The CResponseList object builds CResponse objects that are initialized with the parsing format for the expected response.

The CResponse object converts raw response strings, returned by the CStream, and converts them into C++ data types. For example, a response string containing position data may be converted into a set of double values.

The CStream object is used to communicate directly with the underlying stream component.

FIGS. **14–20** contain scenario maps that describe all main scenarios, or operations, that occur on the driver **30**. Each scenario-map displays all objects involved, and the interactions that take place between them in the sequence that they occur.

There are two types of operations that occur on the driver **30**. First, the driver administrator **32** may initiate operations, such as adding streams or configuring the driver. Next, the motion control component **35** may initiate operations on the driver when an application is actually running. The following discussion describes each perspective, starting with the operations directed by the Driver Administrator; all operations made on the driver by the driver administrator are discussed in the order that they may occur when using the driver.

Before a driver may be used, it must be registered in the OLE system. In order to register a driver the driver administrator first verifies that the module being registered is actually an driver **30**, then it calls the DLLRegisterServer exported function to register the driver. Each module of the system **22** exports a function called DLLGetModuleType. This function is used to verify that the module is an driver **30** component. FIG. **13** displays the interactions that take place when registering a driver.

During the registration process shown in FIG. **13**, the following steps occur. First, the driver administrator must load the DLL, containing the stream component, verify that the module is an driver **30**. To do so, the driver administrator calls the DLLGetModuleType function, exported by the driver. If the function returns a value that contains the value XMC_DRIVER_MT in the high byte, then the driver administrator proceeds and registers the driver by calling its exported function, DLLRegisterServer. When called, the implementation of the DLLRegisterServer writes all OLE 2.0 registration information to the Windows registration database.

Referring now to FIG. **14**, after the driver is registered, the driver administrator can load the component **35** using the OLE CoCreateInstance function. During the initialization process, the driver loads all registration data and initializes both the CDriverDisp and CStreamMgr C++ objects.

During initialization, the following steps occur.

Before loading the driver component, the driver administrator must query the driver module for its CLSID. Calling the driver's exported function, DLLGetCLSID, returns the CLSID. Once it has the CLSID, the driver administrator may create an instance of the driver by calling the standard OLE function CoCreateinstance. When first loaded, the driver loads any registration data previously stored. Next, the driver directs the CDriverDisp object to initialize the system. When notified, the CDriverDisp object initializes itself and then directs the CStreamMgr to initialize itself. During its initialization, the CStreamMgr loads all stream settings from the registration database. For example, the CLSID and enabled state of all streams previously registered with the driver, are loaded.

After initializing the driver, the driver administrator may perform operations on it. For example, the driver administrator may request the driver to add or remove a stream. FIG. **15** displays the sequence of events occurring when the driver

is requested to add a new stream. When adding a stream, the following steps occur.

First the driver administrator directs the stream to add a new stream and passes CLSID of the stream, to be added, to the driver. The driver then passes the CLSID to the CDriverDisp object and directs it to add the stream. The CDriverDisp object passes the information on to the CStreamMgr and directs it to add the stream. In the final step, the CStreamMgr assumes that the module is a valid stream component **28** and adds the CLSID to the drivers set of information in the registration database.

Another operation requested of the driver, after initialization, is that of querying it for its current settings. Before displaying information about the driver, like the name of the hardware it supports, the driver administrator must query the driver for the information. For example, FIG. **16** displays the process of querying the driver for an enumeration of the streams registered with it. When querying the driver for information, the following steps occur.

First the driver administrator, calls the interface method used to query the driver's stream enumeration. Next, the driver directs the CDriverDisp to create the stream enumeration. The CDriverDisp object then directs the CStreamMgr to prepare the stream enumeration. The CStreamMgr checks the registration database and makes sure its internal state is in sync with the data stored in the registry. Next, it sets a lock that will cause all stream management operations, such as adding or removing streams, to fail. The CStreamMgr prepares the list of streams and loads them into memory using the CStream object. The CStream object loads the stream component using the OLE CoCreateInstance API.

After the driver administrator is done using the driver, it must release the driver by calling its exposed Release method. Calling this method, directs the driver to release all resources used. FIG. **17** displays the process of releasing the driver component. During the clean-up process, the following steps occur.

First the driver administrator must direct the driver component to clean itself up by calling its Release method. When invoked, the driver component passes the call on to the CDriverDisp object. The CDriverDisp object then directs the CStreamMgr to save all data. The CStreamMgr saves all data, including the state of each stream, in the registration database. Finally, the driver saves all internal data in the registration database.

After a driver is successfully installed into the system **22** and configured using the driver administrator, it is ready for use by the motion control component **35**. The component **35** uses the driver **30** when performing motion control operations requested from the application using the component **35**. The following discussion describes the component **35** directed operations that can take place on the driver.

Before using the driver, it must be initialized by the component **35**. This operation is different from the driver initialization taking place on the driver when used by the driver administrator because the system must be prepared for sending and receiving commands. In order to prepare for the data communication, the stream must be initialized and then opened. FIG. **18** describes the initialization process. The following steps occur during the initialization process.

First the component **35** must direct the driver to initialize itself. This is usually a two step process. In the first step, the component **35** creates and instance of the driver using the standard OLE CoCreateInstance function. Next, the Initialize method, exposed by the driver, is called to prepare the driver for data transmissions. When the Initialize method is called, the driver first loads any internal data stored in the registration database **42**. Next, the driver directs the CDriverDisp to initialize the internal system. The CDriverDisp then directs the CStreamMgr to initialize the streams. Next, the CStreamMgr loads all data from the registration database, including the set of all CLSID's and enabled status' for all streams registered with the driver. Then the CStreamMgr loads each enabled stream by creating a new CStream object for each enabled stream. When creating each CStream object, the CLSID for the underlying stream is passed to the CStream object. When each CStream object is created and attached to a stream component it loads the component **35** by calling the standard OLE CoCreateInstance function. Once the CStreamMgr is done, the CDriverDisp directs the CCommandMgr to initialize itself. During its initialization process, the CCommandMgr initializes and loads the CCommandList. Also, when the CCommandMgr is initializing, it loads the CResponseList corresponding to the CCommandList.

Once the system is initialized, the motion control component **35** can direct the driver to carry out certain command operations. Command operations are standard motion control operations such as moving to a specific location in the system, or querying the system for the current position. FIG. **19** describes the process of commanding the driver to carry out a certain operation. When commanding the driver to perform a certain operation the following steps occur.

First, the component **35** directs the driver to perform the operation, such as moving to a position or querying the system for the current position. Next, the driver directs the CDriverDisp object to perform the operation. The CDriverDisp object then directs the CCommandMgr to build the appropriate command. Any parameters related to the command are passed to the CCommandMgr. For example, when directing the driver to move to a certain position, the position information is passed to the CCommandMgr. Next, the CCommandMgr requests the CResponseList to create a CResponse object. The CResponseList looks up the response format and uses it to create a new CResponse object that is returned to the CCommandMgr. Then, the CCommandMgr directs the CCommandList to create the command. Any parameters related to the command are passed to the CCommandList. The CCommandList creates a new CCommand object, looks up the raw command string, and passes it and the command parameters to the CCommand object who then builds the command string.

The CCommandMgr, then passes the CCommand object, returned by the CCommandList, and the previously created CResponse object to the CStreamMgr object. The CStreamMgr object is directed to process the objects. The CStreamMgr passes the CCommand and CResponse objects to all enabled CStream objects. The CStream object queries the CCommand object for the full command string in raw text form. The raw text command is passed to the stream component. Next, the CStream object waits for the response, then reads the raw text response into a buffer. The raw text response is then passed to the CResponse object. Next the CRETONNE object is returned to the CStreamMgr, who returns it to the CCommandMgr, who returns it to the CDriverDisp object. Eventually the CResponse returns to the CDriverDisp object, who then directs the CResponse to convert the response into a generic C++ type. The generic type is returned no the motion control component **35**.

Once the component **35** is finished using the driver, the driver must be released by calling its Release method. Releasing the driver frees all resources used by the driver. FIG. **20** describes the process of releasing the driver. The

following steps occur when cleaning up and freeing all resources used by the driver.

First, the component **35** must call the driver's Release method. When called, the driver directs the CDriverDisp object to release any resources used. The CDriverDisp then directs the CStreamMgr to free any resources used. The CStreamMgr then frees all active CStream objects. Each CStream object releases all stream component interfaces used. Next the CDriverDisp directs the CCommandMgr to free all of its resources. During its clean-up, the CCommandMgr frees the CCommandnist object. To complete its clean-up, the CCommandMgr frees the CResponseList object.

Attached hereto as Appendix D is a document that describes the actual OLE Interfaces exposed, the definitions of the data structures used when passing data around, and the definitions of each class used internally by the driver.

IV. STREAMS

This section describes the stream component **28** used as the data transport layer between the driver **30** component and the destination output location such as the motion control device **20** and/or the output file **34**. For example, when using motion control hardware that is connected to the PC Bus, the driver **30** Component will communicate with the PC Bus stream component **28**.

The design of a stream component **28** will be discussed in three parts. First, a Module Interaction-Map describes the modules that are involved, with respect to the stream, and how they interact with one another. Next, the Object Interaction-Map breaks the Module Interaction-Map down into a more detailed view that not only displays the interactions occurring between modules, but also the interactions taking place between the C++ objects within the stream component **28**. Then, the Object Interaction-Map is "tested" by running it through several Scenario-Maps. Each Scenario-Map displays the object interactions taking place during a certain operation.

The Module Interaction-Map shown in FIG. **22** displays all modules that interact with the stream component **28**. Interactions begin from two different perspectives. First, the driver administrator **32** interacts with the stream component **28** when installing, removing, and configuring the stream. Next, when used, each driver **30** interacts with the stream while sending and retrieving data to and from the destination. For example, when a driver writes data to a text file stream, the stream takes care of writing the data out to the file. Or, if the driver reads data from a PC Bus stream, the stream does the actual read from the hardware and passes the data back to the driver.

Drivers only communicate with streams that have been specifically connected to the driver. Once connected, the stream is used to communicate with the destination object, like the PC Bus, serial I/O connection, text file, or debug monitor.

The stream component **28** shown in FIG. **22** is the object that operates as the data transport layer for each driver. Each stream has a different target that defines the type of the stream. The following are the current stream targets.

PC Bus/WinNT—This Windows NT stream uses a Windows NT .SYS device driver to communicate directly with the motion control hardware connected to the PC Bus.

PC Bus/Win95—This Windows 95 stream uses a Windows 95 V×D to communicate directly with the motion control hardware connected to the PC Bus.

PC Bus/Win 3.1—This Windows 3.1 stream communicates directly with the motion control hardware connected to the PC Bus.

Serial—This stream uses the COMMAPI to communicate with the motion control hardware connected to the serial port.

Text File—This stream is write-only and sends all data to a text file.

Debug Monitor—This stream is write only and sends all data to the debug monitor.

Custom—This is a custom stream that sends data to an unknown location.

Similar to the Module Interaction-Map, the Object Interaction-Map displays interactions between modules. In addition, this map, shows all interactions taking place between each C++ object within the stream component **28**. FIG. **23** is the Object Interaction-Map for the stream component **28**.

Each object in the diagram is described as follows. The CStreamDisp object is the dispatch object used to dispatch exposed interface methods. During the dispatch process, all raw data is converted into the appropriate C++ form. For example, collections of data passed between OLE components are usually packaged in a raw block of memory. The CStreamDisp object takes care of packing outgoing data and unpacking incoming data. Data packing involves converting the data between a raw and native C++ format.

The CRegistryMgr object takes care of managing all data stored in the registration database. Since many streams of the same type may exist at the same time, each stream is assigned a handle. The handle assigned, is used by the stream to look up the location it uses to load and store data in the registration database, much as an library index is used to locate a library book.

All input and output is funnelled through the CIOMgr manager. Management of input and output operations consists of buffering data and controlling primitives used to transport data to and from the target location.

The CIOHAL object is the input/output hardware abstraction layer. With in this object lay all hardware dependent code such as calls to inp and outp. Each different type of stream contains a different implementation of this object.

Scenario-Maps are specialized Object Interaction-Maps that display how each module and the objects inside the stream component interact with one another during the operation described by the map. The Scenario-Maps in FIGS. **24-32** are broken into two different categories; those that are initiated by the driver administrator **32**, and those that are initiated by the driver **30**.

Operations directed by the driver administrator are usually related to initializing, uninitializing, and configuring the stream. The following sections describe all operations, directed by the driver administrator, that take place on the stream.

Before a stream component can be used by anyone, it must be registered in the Windows registration database. Registration is a standard OLE **2.0** operation required in order to use any OLE **2.0** component, such as the stream component. FIG. **24** describes this process. During the registration process, the following steps occur.

First, the driver administrator must load the DLL, containing the stream component, verify that the module is an stream component **28**. To do so, the driver administrator calls the DLLGetModuleType function, exported by the stream. If the high byte in the return value contains the value XMC_STREAM_MT, then the driver administrator proceeds and registers the stream by calling its exported

function, DLLRegisterServer. When called, the implementation of the DLLRegisterServer writes all OLE 2.0 registration information to the Windows registration database.

After the stream component is successfully registered, it is ready for initialization. During initialization, the stream component not only initializes itself, but also initializes any device drivers used by registering the driver with the operating system. For example, the Windows NT stream component registers the Windows NT .SYS driver with Windows NT and starts the service. FIG. 25 describes this process. During initialization, the following steps occur.

First the driver administrator must direct the stream to initialize itself. When making this call, the name and location of the driver used, and the handle of the stream are passed into the method as arguments. Once directed to initialize itself, the stream component calls the CStreamDisp and directs it to initialize the system. The CStreamDisp object then directs the CRegistryMgr to load all pertinent data for the stream using the handle passed to it. The CRegistryMgr loads all data from the registration database. After all information is loaded from the registry, the CStreamDisp directs the CIOMgr to register the appropriate driver with the operating system. The CIOMgr directs the CIOHAL to register the driver, if appropriate. If running in Windows NT, the CIOHAL registers the .SYS driver with the Windows NT operating system and starts the driver. If running in Windows 95, the V×D integrity is verified with a quick, dynamic, load and unload.

After initializing the stream component, it may be queried for its current settings or directed to set new settings. Since both operations are very similar, only changing settings will be described. Stream settings include data such as: port addresses, IRQ levels, file names, etc. Any data needed to communicate with the output/input target are included in the stream settings. FIG. 26 describes the process of changing the streams settings. During the setup process, the following steps occur.

First the driver administrator directs the stream to use the data passed to change its internal data. Once directed, the stream component passes the interface method invocation to the CStreamDisp object. The CStreamDisp object then directs the CRegistryMgr to store the new settings. The CRegistryMgr stores the new values in the registration database.

When the driver administrator is done using a stream component, it must clean up the resources used. FIG. 27 describes this process. During the clean-up process, the following steps occur. First the driver administrator must direct the stream component to clean itself up by calling its Release method. When invoked, the stream component passes the call on to the CStreamDisp object. The CStreamDisp object then directs the CRegistryMgr to save all data. All persistent data is saved to the registration database by the CRegistryMgr.

Driver directed operations occur when each driver 30 uses the stream component 28 connected to it. Remember, each stream component is used as the data transport layer. Each driver uses the stream to transfer the motion control command data, it generates, to the output target. Streams are also used to transfer data back to the driver when read operations occur. Only certain streams are readable.

Before the driver can perform operations on the stream, the stream must be initialized. Initialization occurs in two steps. First the OLE stream component must be loaded, and then once it is, the stream must be explicitly initialized. FIG. 28 describes the second portion of the initialization process. The following steps occur during the initialization process.

First the driver must invoke the Initialize methods exported by one of the stream interfaces. When calling Initialize, the driver passes to the stream, the stream handle. Next, the stream passes the directive on to the CStreamDisp object for dispatching. The CStreamDisp object first directs the CRegistryMgr to load all settings stored in the location defined by the stream handle. The CRegistryMgr reads in the data stored in the registry at the handle. After the data is loaded, the CStreamDisp, directs the CIOMgr to initialize itself. As part of its initialization, the CIOMgr initializes the CIOHAL object that it is using.

Once a stream has been initialized, it must be opened. Opening a stream places the stream in a state where it can pass data between the driver and the target. FIG. 29 describes the process of opening a stream. When opening a stream, the following steps occur.

First the driver directs the stream to open itself, by calling the Open exposed interface method. Once directed, the stream passes the call on to the CStreamDisp object. Next, the CStreamDisp object directs the CIOMgr to open the stream. At this time, the CIOMgr prepares any buffers that will later be used when transferring data through the stream. After the buffers are ready, the CIOMgr directs the CIOHAL object to interact with the target and open it. CIOHAL directly communicates with the target or with a device driver and opens the stream. When operating with hardware streams, the device driver, or Serial IO directly communicates with the hardware and prepares it for operation.

After opening a stream, it is ready to perform data transport operations. There are two main data transport operations available: Reading data, and writing data. FIG. 30 describes the process of writing data to the stream. When writing to the stream, the following steps occur. First the driver directs the stream to write data to the target and passes the data to the stream. Next, the stream passes the data to the CStreamDisp object. The CStreamDisp object passes the block of data to the CIOMgr and directs it to write it to the target. The CIOMgr object either passes the complete block of data to the CIOHAL object, or stores the block in an internal buffer and then passes pieces of the buffer to the CIOHAL object until the complete buffer is sent. The CIOHAL object takes the data passed to it and either sends it directly to the target, passes it to a device driver, or calls COMMAPI to send the data to the Serial IO port. The device driver or COMM API sends the data directly to the hardware controlled.

Certain streams, like the PC Bus and Serial IO streams, return data after write operations occur on them. The data returned may be specific to a previous request for data, or status describing the success or failure of the previous write operation. FIG. 31 describes the process of reading data from the stream. It should be noted that not all streams are readable. Currently, the only readable streams are the PC Bus and Serial streams. During the operation of reading data from the target, the following steps occur.

First the driver directs the stream to read data from the target. The stream passes the call on to the CStreamDisp object. The CStreamDisp object directs the CIOMgr to perform the read. Depending on how the stream is implemented, the CIOMgr may either make one call or multiple calls to the CIOHAL object. If multiple calls are made, all data read is stored in CIOMgr internal buffers. The CIOHAL object either directly communicates to the hardware, uses the COMMAPI, or a device driver to read the data. If a device driver or the COMMAPI are used, they directly communicate with the hardware to read the data.

Once the driver is done using the stream, it must direct the stream to clean-up all resources used. To do so, the driver

calls the standard Release method. FIG. **32** displays the sequence of events taking place after the Release method is called. The following steps occur when cleaning up and freeing all resources used by the stream.

First the driver must call the stream's Release method. Next, the stream directs the CStreamDisp object to release all resources used. The CStreamDisp object then directs the CIOMgr to free any resources used in buffers, etc. Next, the CIOMgr directs the CIOHAL to free any resources used. During its clean-up and depending on the type of stream, the CIOHAL will delete text files used, close the debug monitor, shut-down the hardware, or direct any device drivers to shutdown the hardware. If device drivers or the COMM API are used, they direct the hardware to shut-down.

FIG. **33** depicts an interface map for the stream **28**. Attached hereto in Appendix E is a document that describes the actual OLE Interfaces exposed, the definitions of the data structures used when passing data around, and the definitions of each class used internally by the stream.

V. DRIVER STUB MODULE

The driver stub module **36** is used to fill in the extended SPI functions that the driver **30** is unable to support or implement. By simulating the extended SPI, applications are able to use a larger set of motion control functionality than would be available if the application directly programmed the motion control hardware. In order to implement the extended SPI, the driver stub uses software algorithms that call core SPI interface methods implemented by the driver **30**. During the initialization of the driver stub, the driver **30** to use is registered with the driver stub.

This section describes all aspects of the driver stub **36** in three basic parts. The first part of this section describes all binary modules affecting the driver stub. Next, a more detailed view, that includes all C++ objects used inside the driver stub, is described. Then several processes that take place on the driver stub are described.

The module interaction-map displays all binary modules and their interactions with the driver stub **36**. As can be seen from FIG. **34**, the driver stub is used by the component **35**. More or less, the driver stub acts as a helper to the component **35** by filling in all extended SPI functionality possible.

By taking the module interaction-map in FIG. **34** and displaying all interactions taking place with all C++ objects implementing the driver stub, we produce what is called the object interaction-map. FIG. **35** is the object interaction-map for the driver stub **36** component.

Each object in the diagram is described as follows.

The CDriverStubDisp object is the dispatch object used to dispatch exposed interface methods. During the dispatch process, all raw data is converted into the appropriate C++ form. For example, collections of data passed between OLE components is usually packaged in a raw block of memory. The CDriverStubDisp object takes care of packing outgoing data and unpacking incoming data. Data packing involves converting the data between a raw and native C++ format.

The CSPIMgr object is responsible for managing all SPI issues such as managing the CSimpleDriver by directing it to connect to the appropriate SPI core interfaces exposed by the driver.

The CSimpleDriver object is used to directly communicate with the driver implementing the SPI core interfaces. The CSimpleDriver only communicates with the core SPI interfaces implemented by the driver.

The following discussion describes all main scenarios, or operations, that occur on the driver stub **36**. Each scenario-map displays all objects involved, and the interactions that take place between them in the sequence that they occur. All operations on the driver stub originate from the motion control component **35**. In addition to the motion control component **35**, the XMC Setup Component interacts with the driver stub when installing the system **22**. It should be noted that all scenarios below assume that the driver stub **36** has already been registered in the OLE system. Registering this component is the responsibility of the setup application and setup component.

This discussion describes all operations made on the driver stub by the motion control component **35**. Each section is discussed in the order that they may occur when using the driver.

As shown in FIG. **36**, before using the driver stub **36**, the motion control component **35** must initialize it by creating an instance of the driver stub, and then initializing the instance created. Calling the standard OLE function CoCreateInstance completes the first step. After an instance is created, the component **35** must call the driver stub exposed Initialize interface method. During initialization, the following steps occur.

The component **35** creates an instance of the driver stub by calling the standard OLE function CoCreateInstance. Once loaded, the CLSID of the driver to use is passed to the driver stub when calling its Initialize exposed interface method. When first loaded, the driver loads any registration data previously stored. Next, the component **35** passes the CLSID, of the driver to use, to the CDriverStubDisp object and directs it to initialize the system. The CDriverStubDisp object then directs the CSPIMgr to initialize itself and passes the driver CLSID to it. The CSPIMgr passes the CLSID to the CSimpleDriver and directs it to only query the core SPI interfaces exposed by the driver. The CSimpleDriver loads an instance of the driver then queries all core interfaces exposed by the driver.

Once the driver stub is initialized, it is ready to perform operations such as performing extended SPI functions. FIG. **37** describes the steps that occur when the component **35** directs the driver stub to perform an extended SPI operation. The following steps occur when the operation is requested.

First the component **35** must request the operation and pass all pertinent parameters to the driver stub. Next, the driver stub directs the CDriverStubDisp to handle the operation. The CDriverStubDisp then directs the CSPIMgr to perform the SPI extended function and passes the appropriate XMC_EXT_SPI identifier as a parameter. The CSPIMgr calls the appropriate function corresponding to the XMC_EXT_SPI identifier. The function simulates the extended SPI function and calls the CSimpleDriver for core operations. When directed, the CSimpleDriver performs SPI core functions by directly calling the exposed interfaces implemented by the driver.

When the motion control component **35** is finished using the driver stub **36**, it must release it by calling the exposed Release method. Calling the Release method causes the driver stub to free all the resources it uses. FIG. **38** displays this sequence of events. During the clean-up process, the following steps occur.

First the component **35** must direct the driver stub to release all of its resources by calling its Release method. When invoked, the driver component passes the call on to the CDriverStubDisp object. The CDriverStubDisp object then directs the CSPIMgr to release any resources that it was using. The CSPIMgr releases all resources including the CSimpleDriver object used. When freed, the CSimpleDriver releases any interfaces used from the driver.

FIG. 39 is an interface map of the driver stub module 36. Attached hereto as Appendix F is a document that describes the actual OLE Interfaces exposed, the definitions of the data structures used when passing data around, and the definitions of each class used internally by the driver.

VI. DRIVER ADMINISTRATOR MODULE

The driver administrator 32 is used from two different perspectives. When the driver administrator Control Panel Applet 38 is used to configure the system, the applet directs the driver administrator 32 to carry out the operations. The applet 38 simply provides the user-interface, and the component 35 does the real work of managing drivers and streams used with the system 22. Using the driver administrator component with the control panel applet is the first perspective on using the component 35.

In the second perspective, the motion control component 35 uses the driver administrator component to query for the current set of enabled the driver 30. It should be noted that, currently, only single driver operation is allowed. Clearly, the system 22 may support multiple drivers that are virtualized. For example, if two, four axis, drivers are installed, applications using the system could act as though they were using an eight axis system.

This section describes the driver administrator 32 in three main parts. First, all modules interacting with the driver administrator component are described along with their interactions. Next, the module interaction-map is expanded to display all interactions taking place between the C++ objects used to implement the driver administrator 32 Component. This description is called the object interaction-map. Then, the object interaction-map is tested by running it through several scenarios, or scenario-maps. Each scenario-map displays the events and the order in which they occur in a certain process taking place on the driver administrator component.

The module interaction-map shown in FIG. 40 displays all binary modules and their interactions with the driver administrator 32 Component. Both the driver administrator CPL 38 and the motion control component 35 are the main modules that interact with the driver administrator 32 Component.

The driver administrator CPL module 38 provides the user-interface that allows the user to add, configure, and remove drivers and streams in the system 22. The driver administrator 32 handles all driver and stream management. Even though the control panel applet provides the user-interface, this module 32 does the actual management work.

In addition, the driver administrator is used by the component 35 to access the current driver(s) to use when carrying out motion control operations. For example, if the AT6400 driver is selected as the current driver when the component 35 queries the driver administrator, the driver administrator returns the CLSID of the AT6400 driver.

Taking the driver administrator 32, displayed in the module interaction-map, and displaying all interactions occurring between the C++ objects used to implement the administrator 34, produces the object interaction-map therefor. The object interaction-map for the driver administrator 32 is shown in FIG. 41.

Each object in the diagram is described as follows.

The CDriverAdminDisp object is the dispatch object used to dispatch exposed interface methods. During the dispatch process, all raw data is converted into the appropriate C++ form. For example, collections of data passed between OLE components is usually packaged in a raw block of memory. The CDriverAdminDisp object takes care of packing outgoing data and unpacking incoming data. Data packing involves converting the data between a raw and native C++ format.

The CDriverInfoMap object is used to build the information used by the driver administrator CPL 38 when displaying information about each driver or stream.

The CModuleMgr object is responsible for managing all stream and driver modules in the system. A list of all drivers registered are stored persistently in the registration database by the CModuleMgr. Each time a driver or stream is accessed the CModuleMgr is used to get the module.

The CSimpleDriver object is used to directly communicate with the driver component. All OLE specific details are encapsulated within this object.

The CSimpleStream object is used to directly communicate with the stream component. All OLE specific details are encapsulated within this object.

FIGS. 42–49 describe all main scenarios, or operations, that occur on the driver administrator 32. Each scenario-map displays all objects involved, and the interactions that take place between them in the sequence that they occur.

Referring now to FIG. 42, before using the driver administrator component, it must be initialized. FIG. 42 describes the process of initializing the driver administrator component from either the driver administrator control panel applet or the motion control component. During initialization, the following steps occur.

First, either the control panel applet or the motion control component must create an instance of the driver administrator component by calling the standard OLE function CoCreateInstance. Next, the exposed Initialize interface method must be called. When the Initialize method is called, the driver administrator component directs the CDriverAdminDisp to initialize the system. Next, the CDriverAdminDisp directs the CModuleMgr to initialize itself and any modules that it is managing. The CModuleMgr, first, loads all information from the registration database. Then for each driver registered, the CModuleMgr creates an instance of the driver by calling the standard OLE function CoCreateInstance. Next, the CModuleMgr calls each drivers Initialize method, passing to the method the CLSID of the driver component to attach. The CSimpleDriver attaches to the driver component by calling the standard OLE function CoCreateInstance.

The driver administrator 32 can register both drivers and streams. Registering drivers is very direct, since the driver administrator manages the drivers registered in the system. Registering streams, on the other hand, is more complex, since each stream must be registered with a driver and the driver manages the streams registered with it, not the driver administrator. The following discussion describes the process of registering both drivers and streams.

Registering a driver entails verifying that the module is actually a driver, verifying that the driver can be loaded, and storing the driver information in a persistent location. FIG. 43 describes this process. When registering a driver, the following steps occur.

First, the driver administrator CPL passes the name of the driver and directs the driver administrator component to register it. Next, the driver administrator component passes the driver name to the CDriverAdminDisp and directs it to register the module. The CDriverAdminDisp directs the CModuleMgr to register the new driver. The CModuleMgr

creates a new CSimpleDriver and requests it to register the driver. First the CSimpleDriver verifies that the driver is valid by calling its DLLGetModuleType exported function. If the function returns XMC_DRIVER_MT the CSimpleDriver then calls the driver's exported function DLLRegisterServer to register the module in the OLE system. Next the CLSID is queried from the module by calling its exported DLLGetCLSID function. The CLSID returned is then used to load the driver by calling the standard OLE function CoCreateInstance. If the CSimpleDriver is successful, the CModuleMgr stores the driver CLSID in the registration database.

Registering a stream is similar to registering a driver, but a little more complex, since each stream must be registered with a specific driver. FIG. **44** displays the process of registering a stream. When registering a stream, the following steps occur.

First, the driver administrator CPL passes the CLSID of the driver and the filename of the stream to register with the driver, to the driver administrator component. The driver administrator component directs the CDriverAdminDisp to register the stream. The CDriverAdminDisp object directs the CModuleMgr to register the stream and passes the CLSID of the driver and the name of the stream along to it. First, the CModuleMgr verifies that the CLSID of the driver one of the registered drivers. If it is not, the driver is registered as discussed above.

Next, the CModuleMgr creates a new CSimpleStream object and directs it to verify and load the stream component. The CSimpleStream first verifies that the module is actually an stream component **28** by calling its exported DLLGetModuleType function. If the function returns XMC_*STREAM*_MT, the CSimpleStream continues and registers the stream component by calling its DLLRegisterServer exported function. Finally, the CSimpleStream object queries the new module for its CLSID by calling the module's exported DLLGetCLSID function. The new CLSID is used, by the CSimpleStream, to load the stream component using the standard OLE function CoCreateInstance. If the CSimpleStream succeeds, the CLSID of the stream is passed along to the CSimpleDriver who is directed to register the stream. The CSimpleDriver passes the CLSID to the driver component and directs it to register the stream.

The following discussion describes setting information in either a driver or stream. When the user edits information in the driver administrator control panel applet **38**, the applet **38** directs the driver administrator **32** to edit the settings for the stream or driver being edited. The following discussion describes how this configuration process works.

Editing the settings of a driver takes place when the user changes the driver settings displayed in the driver administrator CPL. Changing these settings causes the process described in FIG. **45** to occur within the driver administrator component. The following steps occur when setting the driver configuration.

When driver settings are changed in the CPL **38**, the driver administrator CPL directs the driver administrator component to make the appropriate changes to the driver corresponding to the driver handle. A XMC_DRIVER_INFO structure is passed to the component **35**, describing the new values for the driver. The driver administrator Component takes the XMC_DRIVER_INFO structure and the handle to the driver and passes the information to the CDriverAdminDisp object, directing it to change the settings in the driver. The CDriverAdminDisp object directs the CModuleMgr to edit the driver corresponding to the driver handle. The CModuleMgr locates the CSimpleDriver with the handle and directs it to change its settings to those stored in the XMC_DRIVER_INFO structure. The CSimpleDriver passes the XMC_DRIVER_INFO structure to the driver component and directs it to change its settings.

As shown in FIG. **46**, when the user edits stream settings in the driver administrator CPL **38**, the following steps occur.

After the user changes settings for the stream in the CPL, the driver administrator CPL directs the driver administrator component to change the stream's settings and passes a handle to the driver containing the stream, a handle to the stream, and a XMC_STREAM_INFO structure describing the new values. The driver administrator component directs the CDriverAdminDisp object to change the streets settings. The CDriverAdminDisp object directs the CModuleMgr to change the settings of the stream corresponding to the handle.

First, the CModuleMgr locates the driver corresponding to the driver handle. Next, it requests the CSimpleDriver to change the settings for the stream corresponding to the stream handle. The CSimpleDriver searches for the stream corresponding to the stream handle and directs it to change its settings to those stored in the XMC_STREAM_INFO structure. The CSimpleStream directly communicates with the stream component and directs it to change its settings to those in the XMC_STREAM_INFO structure.

There are two different types of information that may be queried from the driver administrator **32**: the enumeration of all drivers registered, and the driver information map. The motion control component **35** uses the driver enumeration when selecting the set of drivers to use and control during motion control operations. The driver information map, on the other hand, is used by the driver administrator CPL **38** to update the user-interface display describing all drivers and streams registered in the system. The following discussion describes the process of querying for both the driver enumeration and the driver information map. Querying for the driver enumeration occurs during the initialization of the motion control component **35**. When initializing, the component **35** must know what drivers to use when performing motion control operations. The driver administrator **32** Component is used for that very purpose. Querying the driver enumeration just returns a pointer to the IXMC_EnumDriver interface exposed by the driver administrator **32** Component. FIG. **47** displays the events that occur when using the interface to get each driver in the enumeration. Using the interface causes, the following steps occur.

First, the motion control component **35** queries the driver administrator **32** Component for the next driver. Next, the driver administrator **32** Component directs the CDriverAdminDisp to get the next driver supported. The CDriverAdminDisp directs the CModuleMgr to get the next driver. The CModuleMgr then directs the CSimpleDriver to either return the CLSID or a pointer to the IUnknown interface for the driver, depending on the parameters of the enumeration. If the CSimpleDriver is requested to return a pointer to the IUnknown interface, the interface is queried from the driver component.

Another set of information that may be queried from the driver administrator **32** consists of the driver information map. This data is used by the driver administrator CPL **38** when displaying information describing the drivers and streams registered in the system. As shown in FIG. **48**, when querying the system for the driver interface map, the following steps occur.

First, the driver administrator CPL **38** queries the driver administrator **32** Component for the current driver information map. When queried, the driver administrator component directs the CDriverAdminDisp to create and load a CDriverInfoMap class. The CDriverAdminDisp creates the CDriverInfoMap. Next, the CDriverAdminDisp passes the CDriverInfoMap to the CModuleMgr and directs it to load the information map. The CModuleMgr queries each driver registered for its internal information. Each CSimpleDriver communicates directly with the driver component and queries it for all pertinent driver information. Next, the CModuleMgr queries each driver for a list of all streams registered with the driver. Using the stream enumeration, each CSimpleDriver creates an array of CSimpleStream objects and returns the array to the CModuleMgr. For each CSimpleStream object in each array, the CModuleMgr queries for all pertinent stream information. Each CSimpleStream communicates directly with the stream component and queries it for all information describing the stream.

After the driver administrator CPL **38** or the motion control component **35** are finished using the driver administrator **32**, they must release the component **35** to free any resources it was using. FIG. **49** describes this process. When cleaning up after a call to the Release method, the following steps occur.

First, either the driver administrator CPL **38** or the motion control component **35** must direct the driver administrator **32** to release itself by calling its Release method. Next, the driver administrator component directs the CDriverAdminDisp object to free all resources used in the system. The CDriverAdminDisp then directs the CModuleMgr to free any resources that it is using. First, the CModuleMgr traces through all CSimpleDriver objects, querying each for their CLSID and enabled state. Next, each CSimpleDriver is freed. Each CSimpleDriver object freed, frees all arrays of CSimpleStream objects registered with it. When freed, each CSimpleStream object releases all interfaces that it was using from the stream component. In its final clean-up, each CSimpleDriver releases all interfaces that it was using from the driver component. All CLSID and enabled state information is stored persistently in the registration database.

FIG. **50** depicts an interface map for the driver administrator **32**. Also, attached hereto as Appendix G is a document that describes the actual OLE Interfaces exposed, the definitions of the data structures used when passing data around, and the definitions of each class used internally by the driver administrator **32** component.

VII. DRIVER ADMINISTRATOR CPL APPLET

This document describes the design of the driver administrator control panel applet (CPL) **38** that is used by the user to add, configure, and remove both drivers **30** and stream components **28** later used by the component **35** when directed to carry out motion control operations. With regard to design, there are three main types of "views" used to look at how the control panel applet works.

First, a module interaction map shown in FIG. displays all main executable and user-interactable items, or modules, that the CPL uses and interacts with. For example, when a dialog is displayed by the CPL executable, both the dialog and the CPL modules are considered to interact with one another. Technically, the dialog is not a module since it is a figment displayed on the screen, but none the less, module interaction maps classify them as such since they are key destination points for user-input.

Second, an object interaction map shown in FIG. **52** displays all main objects making up the modules described in the module interaction map. Objects consist of the actual instances of C++classes defining each object. All interactions between the objects are drawn out in this interaction map.

Finally, FIGS. **53–57** display a set of scenario maps are drawn out using the object interaction map as a basis. Scenario interaction-maps describe the interactions taking place during a specific operation. Initialization, Adding a driver to the system, and Viewing the support offered by a driver, are all examples of a scenario interaction-map.

The design goals for the driver administrator **32** are the following:

1. User-Interface separation—Implement all user-interface elements used to control the driver administrator **32** Component.
2. Upgradable to OCX Client—Eventually each driver and stream may implement all UI elements with an OCX that then passes all input to the corresponding driver or stream. The driver administrator CPL **38** must be designed in a way that is easy to upgrade to become an OCX client.
3. Provide Stream Independence—drivers **30** should not be required to use streams **28** in order to operate. The design of the driver administrator **32** must make amends to ensure that it is not dependent on stream component **28** operations to operate.
4. Use Windows 95 UI—When ever possible, Windows 95 UI elements should be used. For example, TreeViews, ImageLists, Button Bars, Tab Dialogs and any other UI elements should be put to use to ensure a Windows 95 look-and-feel.

The following discussion describes the module interaction map for the control panel applet **38**. A module is defined as either an executable binary, an external data file, or a main user-interface element used when interacting with the user. FIG. **51** is a drawing of all modules that interact with each other when running the driver administrator control panel applet.

The driver administrator CPL **38** is a control panel applet. And, a control panel applet is a special DLL that exports several functions allowing the Windows Control Panel to communicate with the applet.

The Driver Administrator Dialog is the main dialog that appears when selecting the control panel applet icon from the Windows Control Panel.

The Browse Dialog is used to query the user for a filename. For example when adding a new stream or driver, the driver administrator uses this dialog to ask the user for the location of the new driver or stream to add.

The View Support Dialog displays the support provided by the selected driver **30**. Each driver may support a different set of extended functionality. This dialog shows the user exactly how much support is provided by each driver allowing them to determine which functions within their application may not operate when using the driver.

Unlike the Module Interaction-Map described above, the Object Interaction-Map shown in FIG. **52** describes how the actual instances of C++ objects interact with one another within each module.

Other than showing that each dialog is managed by the object, whose name is displayed in the dialog, the main difference from the module IA-map are both the CComCPL and CDriverAdmin C++ objects. Both objects are described below.

As the description of each dialog class is fairly straight forward and very similar to the dialog description above

they will not be described in this section. This section will describe all other C++ objects.

The CComCPL is a C++ object that is generated by the COMBuilder application from a template. It is used to handle all Windows messages sent from the Control Panel Application.

The CDriverAdmin object is used to drive, control, and manage the use of the driver administrator **32** Component. For example, all OLE 2.0 interface management and data translation is handled by this object. Data translation involves translating data from a standard C++ format to a raw format that is handled easily with the OLE 2.0 data transfer mechanisms.

Scenario Interaction-Maps are almost identical to object interaction-maps but they only display the objects and interactions taking part in a specific operation. Also, each interaction is numbered by the sequence in which they occur while the operation is running. The following discussion describes several key operations that occur while running the driver administrator CPL **38** Applet.

Initialization occurs when the user first runs the CPL Applet. During this process all other objects are initialized and several modules are loaded. There are two steps that take place during the initialization process: First the application is initialized, and second the dialog is initialized with values queried from the driver administrator **32**. The following sections describe each.

Initializing the application, which is shown in FIG. **53**, occurs when the application is first run and the main dialog has not yet been displayed. When initializing the application, the following steps occur.

Through a Windows message, Windows notifies the CComCPL object that the Control Panel Applet has just been loaded. CComCPL then loads the CDriverAdminDialog and tells it to do any dialog prepping before going modal. Next, CDriverAdminDialog loads any settings stored in the Registration Database. For example, the current window position and active tab may be stored in the database. CDriverAdminDialog then Loads the CDriverAdmin class and directs it to initialize itself. During initialization, CDriverAdminDialog creates an instance of the driver administrator **32** and queries all interfaces that will be used.

Once the application is initialized, the default settings to be displayed in the dialog must be set. These values are set when the dialog is initialized, just before displaying it. FIG. **54** describes this process. During the process of initializing the dialog, the following steps occur.

During the dialog preparation that occurs before the DoModal call, CDriverAdminDialog queries the CDriverAdmin object for the driver enumeration to be used when setting initial values to be displayed in the dialog box. CDriverAclmin uses the driver administrator **32** to query for the driver information map, which is then passed back to the CDriverAdminDialog. Once receiving the driver information map, the CDriverAdminDialog uses the information to update all user-interface items related to either drivers or streams.

Adding a driver to the system **22** can be broken down into two steps. First, the module name must be added to the system. Next, the driver administrator **32** main dialog must update itself to reflect the new driver just added.

Adding a driver occurs when the user presses the "Add . . ." button on the driver administrator **32**'s main dialog. FIG. **55** describes this process. When adding a new driver, the following steps occur.

When adding a driver, first the user must press the "Add . . ." button. After pressing the button, CDriverAdminDialog opens up the common open file dialog. The user must enter in the filename of the driver to add and close the dialog. CDriverAdminDialog then passes the filename to the CDriverAdmin object and calls the RegisterDriver method passing in the name of the module to register as a driver. CDriverAdmin then passes the driver filename to the driver administrator **32** and directs it to register the driver in the system **22**.

The process of updating the main dialog is identical to the process of initializing the dialog discussed above.

Similar to the process of adding a new driver, removing a driver involves both removing the driver from the system and then updating the main dialog. Pressing the "Remove" button removes a driver from the XMC software system. FIG. **56** describes this process. The following steps occur when removing a driver.

To remove a driver, the user must first select the "Remove" button. After pressing the button, the selected driver or parent driver to the selected stream will be removed. CDriverAdminDialog passes the XMC_HDRIVER of the driver to the CDriverAdmin and directs it to remove the driver by calling its UnRegister method. CDriverAdmin passes the XMC_HDRIVER to the driver administrator **32** and directs it to UnRegister the driver.

The process of updating the main dialog is identical to the process of initializing the dialog discussed above.

Viewing Support involves viewing the level of support implemented by the selected driver. FIG. **57** describes the process of providing this information to the user via the View Support Dialog. The following steps occur when viewing the support provided by the driver.

First the user muse select the "View Support" button on the driver administrator main dialog. When selected, CDriverAdminDialog queries CDriverAdmin for the driver support information. CDriverAdmin passes the query on to the driver administrator **32** component who actually fills out the information. Once the queried information is returned, the CDriverAdmirlDialog passes it on to CViewSupportDialog. CViewSupportDialog initializes itself using the driver support information.

Attached hereto as Appendix H is a document that describes the actual OLE Interfaces exposed, the definitions of the data structures used when passing data around, and the definitions of each class used internally by the driver administrator **32**.

VIII. DRIVER ADMINISTRATOR CPL APPLET

This section contains a description of the driver administrator control panel applet **38**. When using the driver administrator **32** to configure the motion control system, there are two main items that the user will work with: drivers and streams. Each driver **30** generates the hardware specific, control codes that are then sent to the selected stream component **28**. Streams facilitate the data transport layer between the driver and the control-code destination.

Depending on the current hardware setup, different streams may be used. For example, if the hardware is connected to the PC Bus, a PC Bus stream will be used to communicate to it. On the other hand, if the hardware is connected through a serial cable to a serial I/O Port, the serial stream will be used. Finally, all hardware configurations may use the file stream. When using the file stream, all control-codes are sent to the specified file that can be downloaded to the hardware at a later time.

This section describes both drivers and streams, and how each is configured. This section initially describes the driver items and all property pages used to edit them. This section

also contains a description of the streams and their property pages. Finally, this section describes the about box containing details on the Software.

The main purpose of each driver is to generate the hardware-specific control-codes directing the hardware to carry out specific motion control actions. For example, such actions may include querying the hardware for the current position or directing the hardware to move to a predetermined location in the system. The following discussion describes the property pages used to configure each driver.

There are two types of properties affecting each driver. First, a set of defaults may be set that are used by the motion control component **34** as recommended values. The scaling and units used are several example default values. In addition to setting default values, if the driver supports more advanced configuration, pressing the Advanced . . . button will display a dialog box used to set the driver configuration. For example, if a driver does not support streams, the advanced configuration dialog, provided by the driver, will allow the user to set the I/O Port and IRQ settings.

The properties affecting drivers **30** are as follows.

Scaling—Setting the scaling property affects the default scaling used on all axes within the motion control system. The range for scaling values is (0.0, 1.0]. Default setting may be overridden when programming XMC by using the IXMC_StaticState interface.

Units—Setting the units property affects all coordinates used when programming the system **22**.

The unit descriptions are as follows:

MM_ENGLISH—Inches are used as the base unit for all coordinates MM_METRIC—Millimeters are used as the base unit for all coordinates.

MM_NATIVE—The native coordinates defined by the hardware system are used. Coordinates used to program XMC are mapped 1:1 to the hardware coordinates.

Advanced . . . —Pressing this button will display a dialog used to edit any advanced properties for the driver that may be edited by the user.

In addition to allowing the user to set properties, each driver property page displays the full names of both the hardware supported and the hardware vendor who makes the hardware.

The buttons along the bottom of the windows work with the selected driver or stream. The following discussion describes each button and what it does.

Pressing the Make Default button selects the current driver to be the default. If a stream is selected, its parent driver becomes the default driver. The default driver is later used by the motion control component **35**.

Selecting the Add . . . button, displays the Add Module dialog. This dialog is used to add new drivers and streams to the system **22**. Once selected, the new driver or stream will be displayed in the Driver tree view. When adding a stream, the stream is added under the currently selected driver. To enable the stream, you must select the enable check box located in the streams property page.

Selecting the Remove button, removes the current driver or stream selected. If a driver is removed all of its streams are also removed.

Selecting the View Support . . . button displays a dialog used to view the level of XMC support implemented by the driver. For example, all API interfaces and subsequent methods are displayed. If a lack of implementation within the driver prohibits an API interface from operating, the driver stub **36** is used. If the lack of implementation within the driver **30** cannot be replaced by operations within the driver stub **36**, the interface or method is disabled.

The following are descriptions of each graphic found in the XMC Support View Dialog.

D—This graphic means that the interface or method is implemented by the driver **30**.

S—This graphic means that the interface or method is implemented within the driver stub **36**.

X—This graphic means that the interface or method is disabled because of a lack of implementation within the driver **30**.

Like the properties page, a debug page is also provided to set all debugging settings for the driver. Each driver may specify that all API calls used to control the driver are logged. The logging settings only affect the current driver selected. The Output field allows you to select the output stream where all debug information is sent. When Streams is enabled, debug information is sent to the specified text file. When Debug Monitor is enabled, debug information is sent to the debug monitor if it is running. Using Enable to enable a stream turns it on causing all debug information generated to be sent to the stream. More than one stream may be enabled at one time.

Stream Settings are available for each debug stream supported. Text File allows the name of the text file may be set. The Debug Monitor can only be enabled and disabled.

A stream is the transport layer used by the driver to pass data to the destination location. The destination location may be the actual motion control hardware or even a text file. Usually the control language used by a hardware vendor is supported by several different flavors of their motion control hardware. For example, some vendors have both PC Bus based and Serial I/O based motion control hardware that understand the same control language. In such a case, the same driver would be used for each hardware setup but it would communicate with different streams depending on the specific hardware setup. Graphically, each stream is listed below each driver that uses the stream.

This section describes the streams supported by the system **22** and how they are configured.

The PC Bus stream sends all data directly to a PC Bus based motion control hardware system by writing to the specified I/O Ports and IRQ's defined by the hardware. This section describes both the properties and debug settings available for the PC Bus Stream.

Stream properties only affect the currently selected stream. The user is required to select certain settings, such as the I/O Port and IRQ. Without setting these values, the PC Bus Stream will not be able to communicate with the hardware. The properties affecting PC Bus Streams are described below.

The I/O Port is the base port used to communicate with the motion control hardware that the stream is to send data

The IRQ is the interrupt request level used by the hardware.

Pressing the Advanced . . . button will display a dialog allowing the user to edit more advanced stream options. For example, if the stream supports a Port I/O map that the user can edit, the port map would be displayed in this dialog. This button is only enabled for streams supporting advanced features that the user may edit.

When debugging an application program it may be useful to see what codes are actually sent to the hardware. The Debug Settings page for streams allows the user to enable and disable both the Cmd and Bit Streams. The Cmd Stream is used to log all command-codes sent to the hardware. If this level of detail does not provide enough information, the Bit

Stream may be used. When enabled, the Bit Stream logs all values sent through each hardware port. All values read from and written to each port used by the hardware are logged. Note, when enabled, both streams may significantly slow down the application programming the motion control system.

Serial RS-232 Streams are used to send data from the driver to motion control hardware connected to the computer through the serial I/O port. Both property and debug settings only affect the selected Serial RS-232 Stream. The following discussion describes the available settings in each in detail.

All Serial RS-232 property settings must be set by the user for they let the stream know what I/O port and communication protocol to use when communicating with the hardware. The properties affecting Serial RS-232 Streams are as described below.

The Port is the serial port that the hardware is connected to. COM1–COM4 are valid ports that can be used.

The Baud Rate is the speed of data transmission supported by the hardware.

When Hardware is selected a more efficient, but less compatible, communication protocol is used to communicate to the hardware. If errors occur when this protocol is selected, use the XON/XOFF communication protocol.

When the XON/XOFF communication protocol is selected a simple and more compatible communication protocol is used.

Debug settings for the Serial RS-232 Stream are very similar to those supported by the PC Bus Stream. Serial RS-232 Streams only support command logging through the Cmd Stream and do not support bit logging.

The Text File Stream is used to build control-code programs for later use. Using this stream facilitates running the XMC software in code-generation-mode. No motion control actions take place when running in this mode. Instead, control-code programs may be built and stored to file. Later, after programs are built and saved, they may be downloaded to the motion control hardware and run. The following discussion describes the property and debug settings for the Text File Stream.

The main property set, when configuring a Text File Stream, is the actual name and location of the file to use. Once set, the stream is ready for use.

The following properties may be configured for the Text File Stream:

Filename is the filename and location of the file used to store all control-codes generated by the driver **30** selected. Pressing the Browse . . . button displays a dialog allowing you to graphically select the location and filename to use.

No debug settings are available for the Text File Stream.

It should be clear from the foregoing that the present invention may be embodied in other specific forms without departing from the essential characteristics thereof. The present embodiments are therefore to be considered in all respects as illustrative and not restrictive, the scope of the invention being indicated by the appended claims rather than by the foregoing description; all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced therein.

We claim:

1. A method of moving an object in a desired manner, comprising the steps of:

defining a set of motion control operations, where each motion control operation is either a primitive operation that is necessary to perform motion control and that cannot be simulated using other motion control operations or a non-primitive operation that does not meet the definition of a primitive operation;

defining a core set of core driver functions, where each core driver function identifies one of the primitive operations;

defining an extended set of extended driver functions, where each extended driver function identifies one of the non-primitive operations;

defining a set of component functions;

providing component code for each of the component functions, where the component code cross-references at least some of the component functions with at least some of the driver functions;

developing a set of software drivers, where (i) each software driver is developed for a motion control device in a supported group of motion control devices and (ii) each software driver comprises driver code for implementing the motion control operations associated with at least some of the driver functions;

providing an application program comprising a series of component functions, where the application program defines the motion steps necessary to move the object in the desired manner;

selecting one motion control device from the group of supported motion control devices;

attaching the object to the selected motion control device;

selecting from the set of software drivers the software driver developed for the selected motion control device;

generating control commands based on the component functions of the application program, the component code of the component functions, and the driver code of the selected software driver; and

operating the selected motion control device based on the control commands to move the object in the desired manner.

2. A method as recited in claim **1**, in which the step of providing the set of software drivers comprises the step of developing driver code to implement all of the core driver functions.

3. A method as recited in claim **2**, in which the step of developing the set of software drivers further comprises the step of developing driver code to implement at least some of the extended driver functions.

4. A method as recited in claim **3**, in which:

non-supported extended driver functions are extended driver functions having no driver code associated therewith; and

the step of generating control commands comprises the step of generating the control commands based on the driver code associated with a combination of the core driver functions to emulate the motion control operations associated with at least some of the non-supported extended driver functions.

5. A method as recited in claim **4**, in which the step of generating the control commands further comprises the steps of:

developing an extended function pointer table that maps the non-supported extended driver functions to the combination of core driver functions employed to emulate the non-supported extended functions; and

generating the control commands further based on the contents of the extended function pointer table.

6. A method as recited in claim **5**, in which the extended function pointer table contains pointers for both supported and non-supported extended driver functions, where the pointers for the supported extended driver functions point to

driver code for implementing the supported extended driver functions and the pointers for the non-supported extended driver functions point to the combination of core driver functions that emulate the non-supported extended functions.

7. A method as recited in claim 1, further comprising the step of:

determining a first unit system employed by the software drivers; wherein

the step of generating the control commands further comprises the step of converting a second unit system employed by the application program into the first unit system.

8. A method as recited in claim 1, further comprising the steps of:

providing a plurality of destinations of control commands;

providing a plurality of streams, where each stream contains transmit stream code that determines how the control commands are to be transferred to at least one of the destinations of control commands;

selecting at least one of the destinations of control commands; and

transferring the control commands to the selected destination of control commands based on the transmit stream code contained by the stream associated with the selected destination of control commands.

9. A method as recited in claim 8, in which certain of the destinations of control commands generate response data, the method further comprising the steps of:

providing response stream code for the streams associated with the destinations of control commands that generate response data; and

processing the response data based on the response stream code.

10. A method of moving an object in a desired manner, comprising the steps of:

defining a first set of first driver functions and a second set of second driver functions, where each first driver function is identifies an incremental motion step performed by motion control devices and each second driver function identifies a plurality of such incremental motion steps;

providing a plurality of software drivers, where each software driver comprises driver code and is provided for a motion control device selected from a group of supported motion control devices;

providing an application program comprising a sequence of program instructions;

selecting a motion control device from the supported group of motion control devices;

mounting the object on the selected motion control device;

selecting the software driver provided for the selected motion control device;

generating control commands based on the application program, the first and second driver functions, and the driver code of the selected software driver; and

operating the selected motion control device based on the control commands to move the object in the desired manner.

11. A method as recited in claim 10, in which the step of providing the set of software drivers comprises the step of developing driver code to implement all of the first driver functions.

12. A method as recited in claim 11, in which the step of developing the set of software drivers further comprises the step of developing driver code to implement at least some of the second driver functions.

13. A method as recited in claim 12, in which:

non-supported second driver functions are second driver functions having no driver code associated therewith; and

the step of generating control commands comprises the step of generating the control commands based on the driver code associated with a combination of the first driver functions to emulate the plurality of incremental motion steps identified by at least some of the nonsupported second driver functions.

14. A method as recited in claim 13, in which the step of generating the control commands further comprises the steps of:

developing an extended function pointer table that maps the non-supported extended driver functions to the combination of core driver functions employed to emulate the non-supported extended functions; and

generating the control commands further based on the contents of the extended function pointer table.

15. A method as recited in claim 14, in which the extended function pointer table contains pointers for both supported and non-supported extended driver functions, where the pointers for the supported extended driver functions point to driver code for implementing the supported extended driver functions and the pointers for the non-supported extended driver functions point to the combination of core driver functions that emulate the non-supported extended functions.

16. A method as recited in claim 10, further comprising the step of:

determining a first unit system employed by the software drivers; wherein

the step of generating the control commands further comprises the step of converting a second unit system employed by the application program into the first unit system.

17. A method of generating a sequence of control commands for controlling a motion control device to perform a given series of motion steps based on an application program defining the given series of motion steps, the method comprising the steps of:

defining a set of motion control operations, where each motion control operation is either a primitive operation that is necessary to perform motion control and that cannot be simulated using other motion control operations or a non-primitive operation that does not meet the definition of a primitive operation;

defining a core set of core driver functions, where each core driver function identifies one of the primitive operations;

defining an extended set of extended driver functions, where each extended driver function identifies one of the non-primitive operations;

defining a set of component functions;

providing component code for each of the component functions, where the component code cross-references at least some of the component functions with at least some of the driver functions;

developing a set of software drivers, where (i) each software driver is developed for a motion control device in a supported group of motion control devices

and (ii) each software driver comprises driver code for implementing the motion control operations identified by at least some of the driver functions;

selecting one motion control device from the group of supported motion control devices;

selecting from the set of software drivers the software driver developed for the selected motion control device; and

generating control commands based on the application program, the component code, and the driver code of the selected software driver.

18. A method as recited in claim **17**, in which the step of providing the set of software drivers comprises the step of developing driver code to implement all of the core driver functions.

19. A method as recited in claim **18**, in which the step of developing the set of software drivers further comprises the step of developing driver code to implement at least some of the extended driver functions.

20. A method as recited in claim **19**, in which:

non-supported extended driver functions are extended driver functions having no driver code developed therefor; and

the step of generating control commands comprises the step of generating the control commands based on the driver code of a combination of the core driver functions to emulate the plurality of incremental motion steps identified by at least some of the non-supported extended driver functions.

21. A method as recited in claim **20**, in which the step of generating the control commands further comprises the steps of:

developing an extended function pointer table that maps the non-supported extended driver functions to the combination of core driver functions employed to emulate the non-supported extended functions; and

generating the control commands further based on the contents of the extended function pointer table.

22. A method as recited in claim **21**, in which the extended function pointer table contains pointers for both supported and non-supported extended driver functions, where the pointers for the supported extended driver functions point to driver code for implementing the supported extended driver functions and the pointers for the non-supported extended driver functions point to the combination of core driver functions that emulate the non-supported extended functions.

23. A method as recited in claim **17**, further comprising the step of:

determining a first unit system employed by the software drivers; wherein

the step of generating the control commands further comprises the step of converting a second unit system employed by the application program into the first unit system.

24. A method as recited in claim **17**, further comprising the steps of:

providing a plurality of destinations of control commands;

providing a plurality of streams, where each stream contains transmit stream code that determines how the control commands are to be transferred to at least one of the destinations of control commands;

selecting at least one of the destinations of control commands; and

transferring the control commands to the selected destination of control commands based on the transmit stream code contained by the stream associated with the selected destination of control commands.

25. A method as recited in claim **24**, in which certain of the destinations of control commands generate response data, the method further comprising the steps of:

providing response stream code for the streams associated with the destinations of control commands that generate response data; and

processing the response data based on the response stream code.

* * * * *

UNITED STATES
PATENT AND TRADEMARK OFFICE

P.O. Box 1450
Alexandria, VA 22313 - 1450
www.uspto.gov

ELECTRONIC ACKNOWLEDGEMENT RECEIPT

APPLICATION #	RECEIPT DATE / TIME	ATTORNEY DOCKET #
90/019,948	**05/06/2025 02:50:18 PM Z ET**	**UNP-16002**

Title of Invention

Distributed Software Defined Networking

Application Information

APPLICATION TYPE	Utility / ex parte reexam	PATENT #	11695823
CONFIRMATION #	2943	FILED BY	Jennifer Lacroix
PATENT CENTER #	70310235	FILING DATE	-
CUSTOMER #	112449	FIRST NAMED INVENTOR	Pouya Taaghol
CORRESPONDENCE ADDRESS	-	AUTHORIZED BY	-

Documents

TOTAL DOCUMENTS: 12

DOCUMENT	PAGES	DESCRIPTION	SIZE (KB)
RequestTransmittalForm-UNP16002.pdf	4	Receipt of Original Ex Parte Request by Third Party	145 KB
RequestIDS-UNP16002.pdf	5	Information Disclosure Statement (IDS) Form (SB08)	53 KB
Warning: This is not a USPTO supplied IDS fillable form. Data in the form cannot be automatically loaded to other USPTO systems.			
APIArticle.pdf	30	Non Patent Literature	947 KB
US5691897.pdf	78	Other reference-Patent/Application/Search Documents	10445 KB
Request-UNP16002.pdf	32	Receipt of Original Ex Parte Request by Third Party	616 KB
RequestAppendixA.pdf	42	Receipt of Original Ex Parte	493 KB

		Request by Third Party	
RequestAppendixB.pdf	35	Receipt of Original Ex Parte Request by Third Party	443 KB
RequestEx1001-823Patent.pdf	41	Copy of patent for which reexamination is requested	2731 KB
RequestEx1002-FileHist823Patent.pdf	434	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	13285 KB
RequestEx1003-DeclGeier.pdf	59	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	10899 KB
RequestEx1004-Stewart.pdf	59	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	10685 KB
RequestEx1005-Malik.pdf	26	Reexam - Affidavit/Declaration/Exhibit Filed by 3rd Party	4289 KB

Digest

DOCUMENT	MESSAGE DIGEST(SHA-512)
RequestTransmittalForm-UNP16002.pdf	B721925F8701FC0CDA038F468F761D2217F8DDE2018182FF8EE5B9F18932A003F9FBA739CADF03099EC4980B3DF17C6670F689A5F30F1DEC2F506D042795E02B
RequestIDS-UNP16002.pdf	D625B19237009FEF44124D82BB2C2B3FD9999D5EBD636137BDA5628B174BC8409232037EC41B4816BFBA60BE974EAE79250D34ECAD32ED0541B4291ED803F927
APIArticle.pdf	ABACBB4DD22D35C668AE006988B009CDB3138F75EC88F335B4C178B31FF83D1CAA0AA12F5EB98CBAE2A327211C06B0042F0EE634A43CDFB7EDB14CAD21FE259A
US5691897.pdf	B6AD291C69C4140B2A75B26998BAEEC0288163CF9657BE57746D595512F9795735D1FFA4DC438F24FEC5A3201CF3965D032E535199286DC3E6BF0B1FD7F1B9E3
Request-UNP16002.pdf	85B3DED5099CB4951789506D2E71DAE504DB6AA875C44BC15D6B4CF30FF1728F488E007809D5B1CE2533B66FEE0BFEA61324EDA8812E38A06D05C2A1AA9F1418
RequestAppendixA.pdf	EB4BF698C2F810FB26077CA2AE9377DB26A0393C8701559004B6D4A1554FD7F4A516D8AC1FD8DC70C3C01173B31AA21D6C32BEA52691082C741000D63D821923

RequestAppendixB.pdf	F13920F8E0C462BD3BB146CD8499552ECED93898ABE14D9C FA6BA6A6C924521259E6C3A5D03B6E021E53CD731C1C7F92 302CE489C3261F57A9A85C456D8EC8F6
RequestEx1001-823Patent.pdf	2B1CF6176E7B1EF7BF0422BA3C3385B0D45535B4EAF571311 C5E3CA99141383E60D164FA06FF59172C5793B5F96119E68F2 B8AC11D6A48D407662F83F42E58EC
RequestEx1002- FileHist823Patent.pdf	31618141726F0EB976233609D3999E01E7B842172BBCA6B096 B169485CF5F58CBBDAEC277B508A1541824A98674C03B22B4 3A1417D27C886B9AFDE2B8F9D131A
RequestEx1003-DeclGeier.pdf	72EF4CDECCF13B25E2840DE6325C7ECE71D5757C89E0A153 B2180276C8D75387F43950016CA159B41C95535E4D6EC17892 F948D2E90FDFB73E4EA43E3C74F397
RequestEx1004-Stewart.pdf	DC635AC33DDD526C343009A2BB10F3AE867977337D8DEBAB CCA366E39BB15C8180A76F9DEEC9D3175FB3869F052FCFD2 EA67820C353EF04022F509DAFCF4B064
RequestEx1005-Malik.pdf	F6B48A7C29E33CE2D54CFA333FE317FE035D1703D1E852E85 DABE9B5397D3156C4DE3BB06183BBDD787D490996F5D3D01 911690DAB83A016E2BE6C40DBB2C8D9

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Ex. 1005

U.S. Pub. No. 20130054682
to Malik *et al*.
(Malik)

US 20130054682A1

(19) **United States**

(12) **Patent Application Publication**

Malik et al.

(10) **Pub. No.: US 2013/0054682 A1**

(43) **Pub. Date: Feb. 28, 2013**

(54) **PLATFORM FOR DEPLOYMENT AND DISTRIBUTION OF MODULES TO ENDPOINTS**

(75) Inventors: **Jatin Malik**, Bangalore (IN); **Thomas Herchek**, Blue Bell, PA (US)

(73) Assignee: **Fiberlink Communications Corporation**, Blue Bell, PA (US)

(21) Appl. No.: **13/598,256**

(22) Filed: **Aug. 29, 2012**

Related U.S. Application Data

(60) Provisional application No. 61/528,503, filed on Aug. 29, 2011.

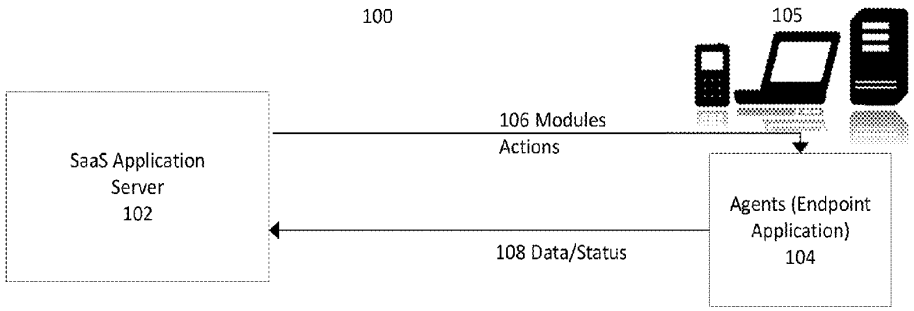
Publication Classification

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/203**

(57) **ABSTRACT**

Systems and methods facilitate the deployment of software modules from a file server to agents on endpoints. Agents can contact a heartbeat server to receive information about a latest manifest file that identifies the software modules that should be installed on the endpoint. Agents can then determine from the manifest information whether new files need to be downloaded and automatically download them. A scripting engine can be used to allow run-time installation of modules without causing significant downtime of the endpoint. The scripting engine can utilize the agent to report back status and data from the software modules to allow modules to be used with a mobile device management platform.



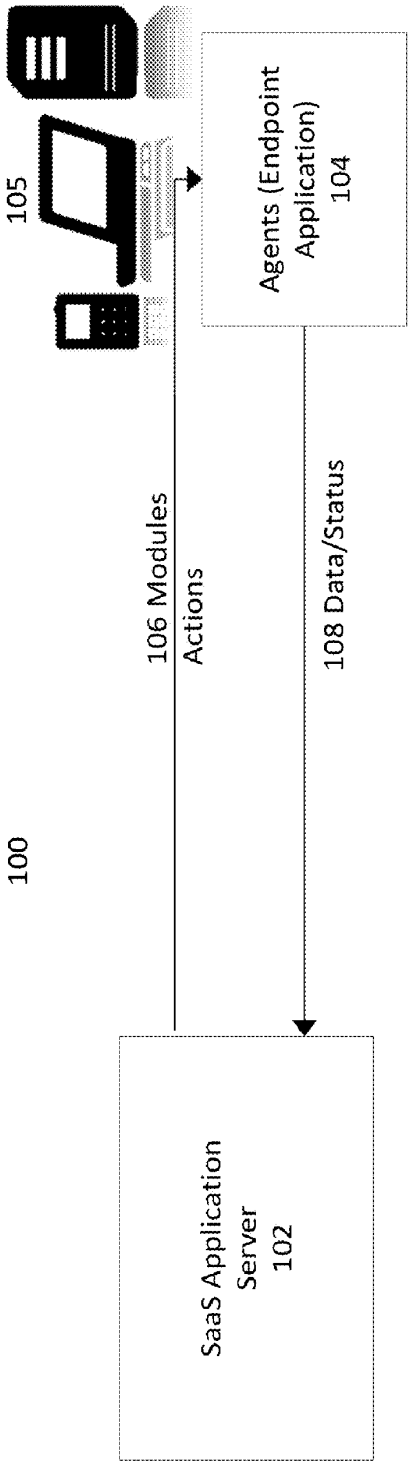

100
105
Agents (Endpoint Application) 104
106 Modules Actions
108 Data/Status
SaaS Application Server 102

FIG. 1

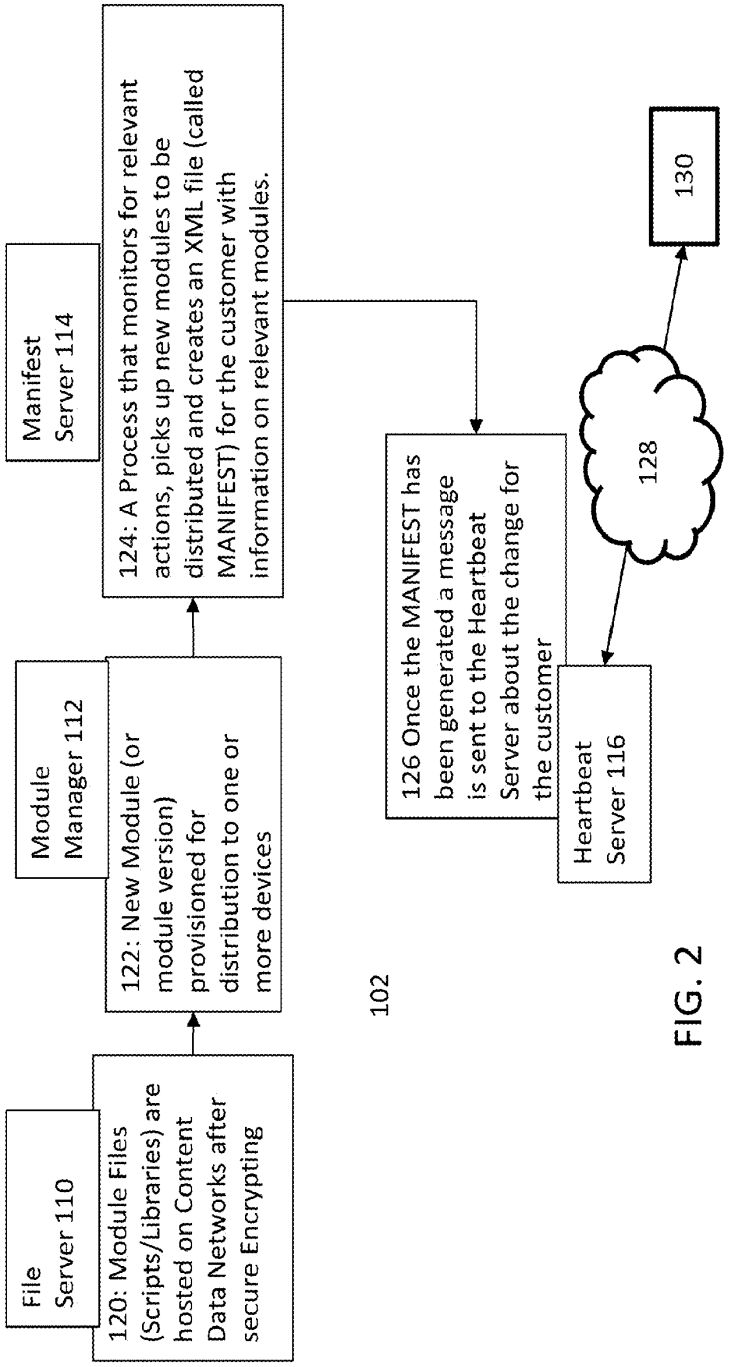

File Server 110
120: Module Files (Scripts/Libraries) are hosted on Content Data Networks after secure Encrypting
Module Manager 112
122: New Module (or module version) provisioned for distribution to one or more devices
Manifest Server 114
124: A Process that monitors for relevant actions, picks up new modules to be distributed and creates an XML file (called MANIFEST) for the customer with information on relevant modules.
126 Once the MANIFEST has been generated a message is sent to the Heartbeat Server about the change for the customer
Heartbeat Server 116
128
130
102

FIG. 2

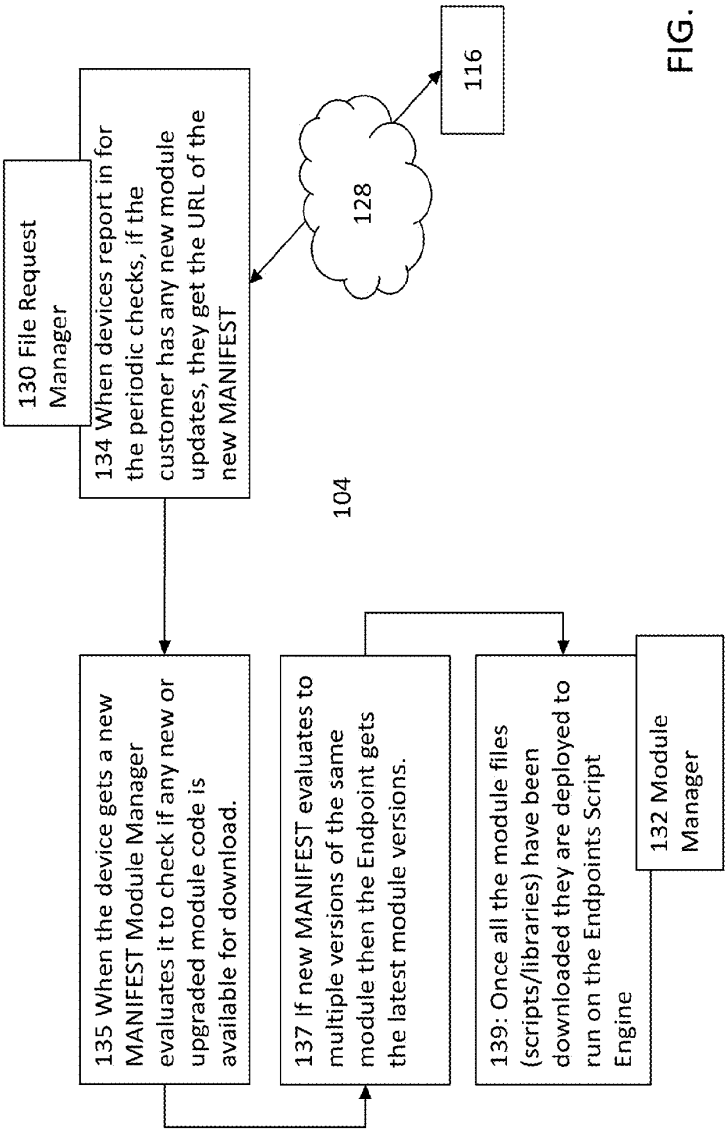

130 File Request Manager
134 When devices report in for the periodic checks, if the customer has any new module updates, they get the URL of the new MANIFEST
128
116
104
135 When the device gets a new MANIFEST Module Manager evaluates it to check if any new or upgraded module code is available for download.
137 If new MANIFEST evaluates to multiple versions of the same module then the Endpoint gets the latest module versions.
139: Once all the module files (scripts/libraries) have been downloaded they are deployed to run on the Endpoints Script Engine
132 Module Manager

FIG. 3

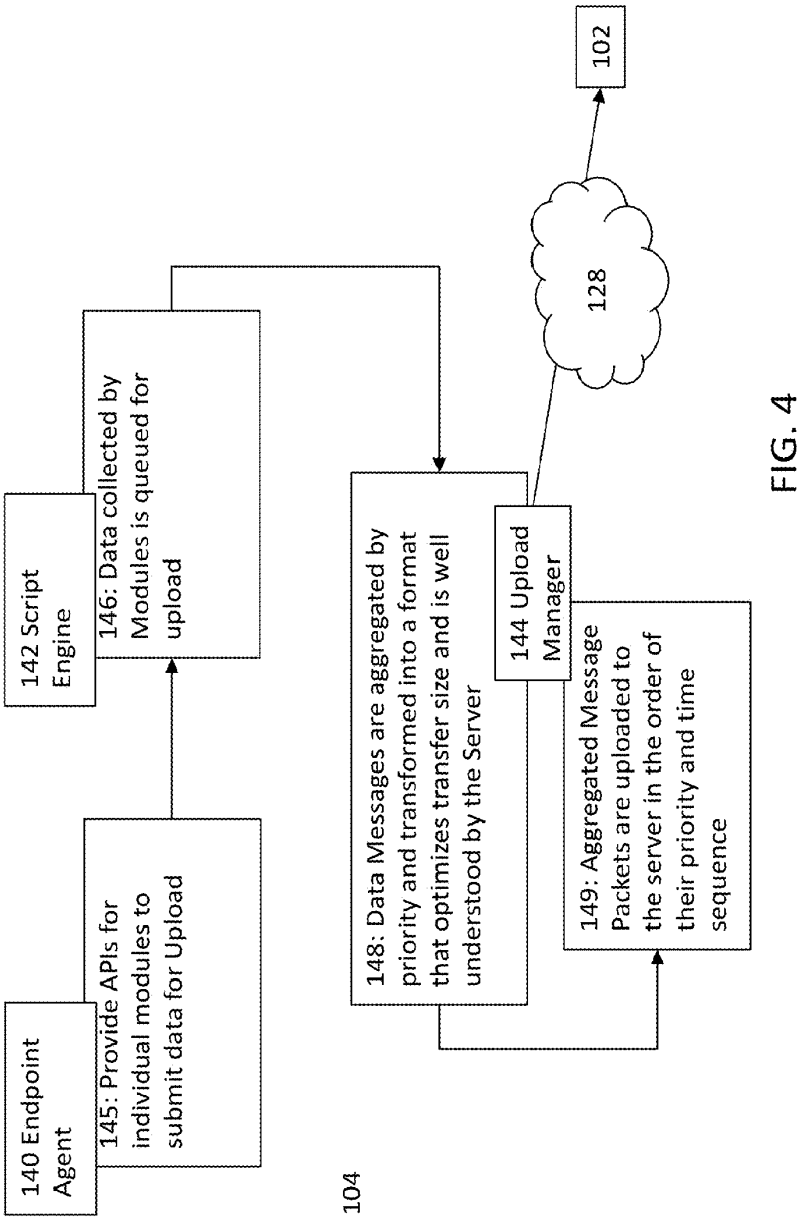

140 Endpoint Agent
145: Provide APIs for individual modules to submit data for Upload
142 Script Engine
146: Data collected by Modules is queued for upload
104
148: Data Messages are aggregated by priority and transformed into a format that optimizes transfer size and is well understood by the Server
144 Upload Manager
149: Aggregated Message Packets are uploaded to the server in the order of their priority and time sequence
128
102

FIG. 4

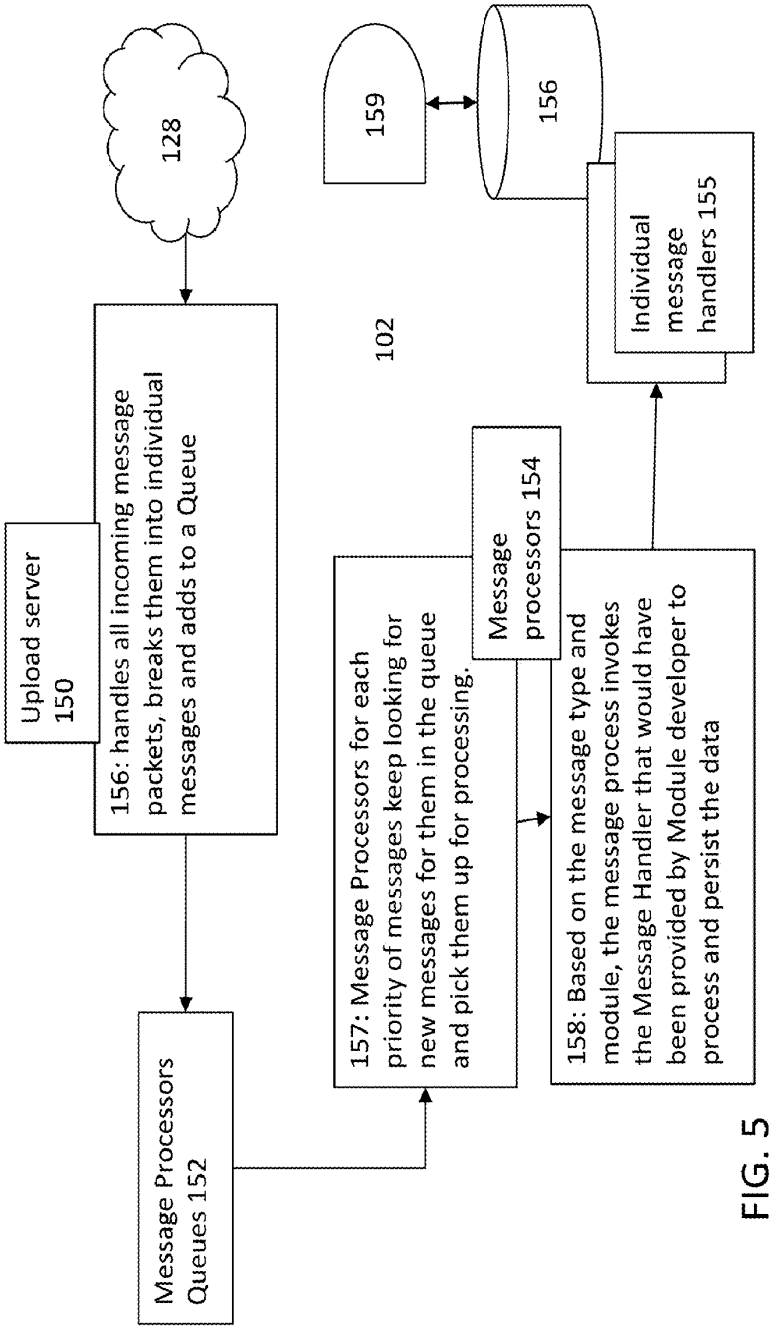

128
159
156
Upload server 150
156: handles all incoming message packets, breaks them into individual messages and adds to a Queue
102
Individual message handlers 155
Message processors 154
Message Processors Queues 152
157: Message Processors for each priority of messages keep looking for new messages for them in the queue and pick them up for processing.
158: Based on the message type and module, the message process invokes the Message Handler that would have been provided by Module developer to process and persist the data

FIG. 5

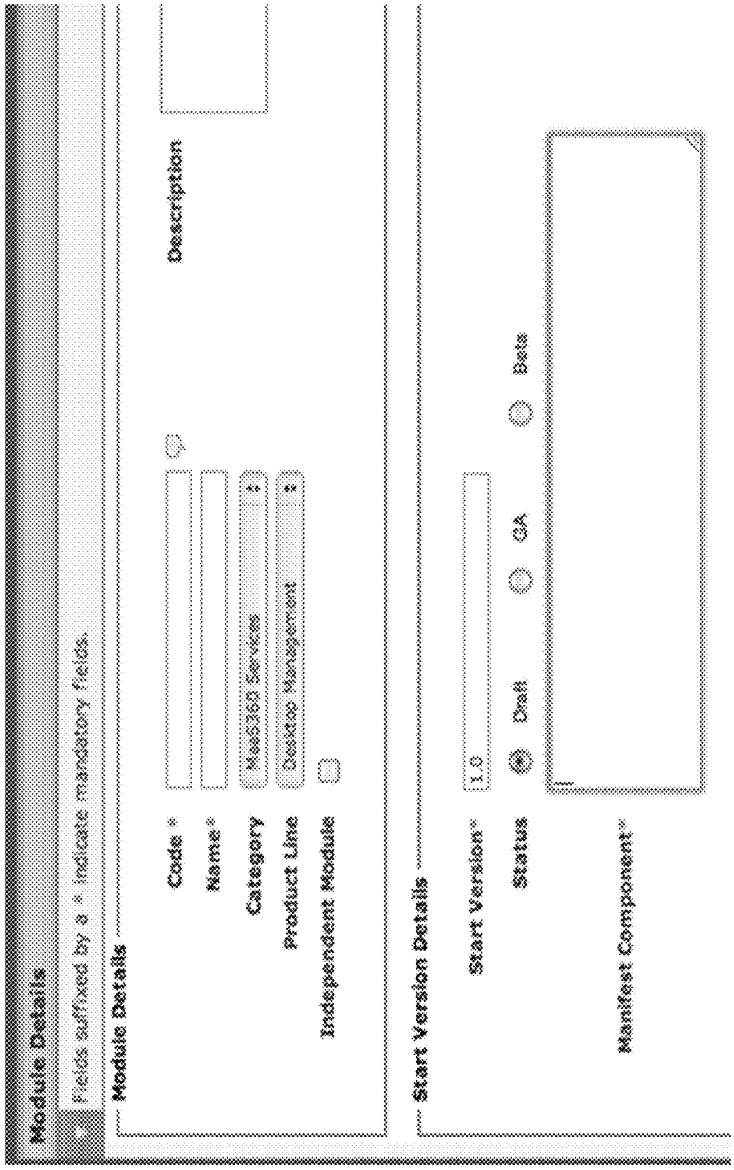
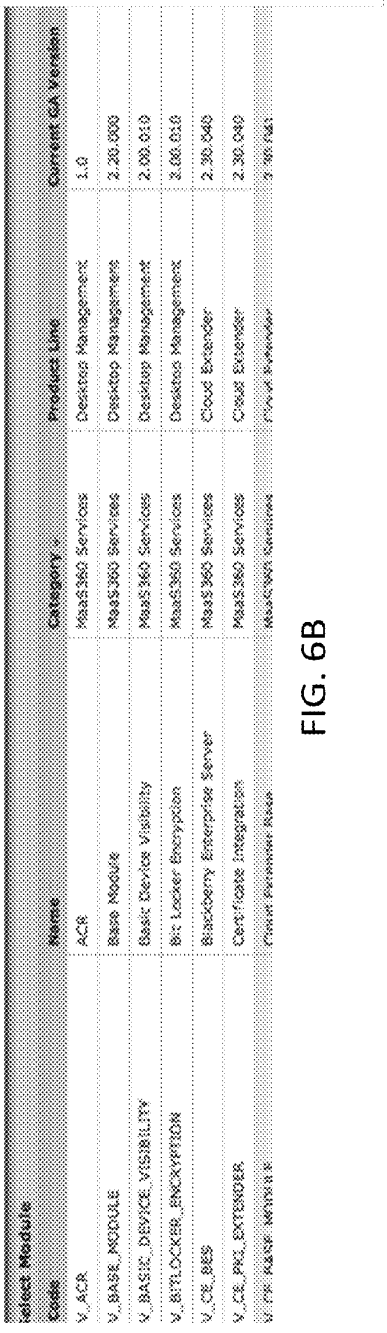

Module Details
Fields suffixed by a * indicate mandatory fields.
Module Details
Code *
Name *
Category
Product Line
Independent Module
Description
Mail360 Services
Desktop Management
Start Version Details
Start Version*
Status
Draft
GA
Beta
1.0
Manifest Component*

FIG. 6A



Select Module				
Code	Name	Category	Product Line	Current GA Version
V_ACR	ACR	MaaS360 Services	Desktop Management	1.0
V_BASE_MODULE	Base Module	MaaS360 Services	Desktop Management	2.20.000
V_BASIC_DEVICE_VISIBILITY	Basic Device Visibility	MaaS360 Services	Desktop Management	2.00.010
V_BITLOCKER_ENCRYPTION	Bit Locker Encryption	MaaS360 Services	Desktop Management	2.00.016
V_CE_BES	Blackberry Enterprise Server	MaaS360 Services	Cloud Extender	2.30.040
V_CE_PKI_EXTENDER	Certificate Integration	MaaS360 Services	Cloud Extender	2.30.040
V_CE_RAPP_MODULE	Cloud Personal App	MaaS360 Services	Cloud Extender	2.30.040

FIG. 6B

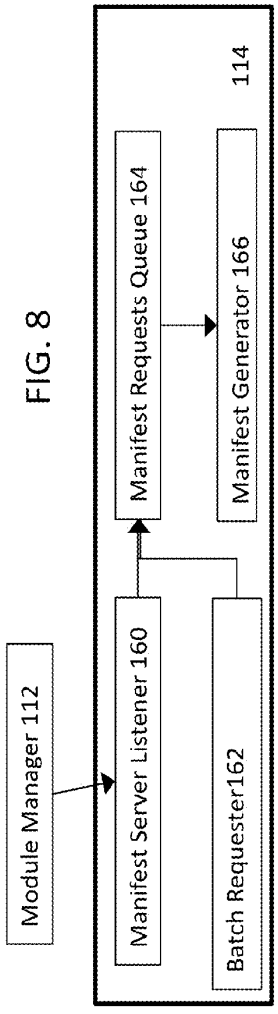


FIG. 8

FIG. 9

```
<heartbeatresponse timestamp="number">
<license value="9438jdkjd3"/>
<urls for="download">
<url value="http://..."/>
<url value="http://"/>
</urls>
<file name="manifest.xml" source="manifest filename"
version=".." key="..." sha1="..." />
<groups>
<group id="..."/>
<group id="..."/>
</groups>
<users>
<user id="..." + additional attributes + >
<groups>
<group id="..."/>
<group id="..."/>
</groups>
</user>
<user id="..." + additional attributes + >
</user>
</users>
</ heartbeatresponse >
```

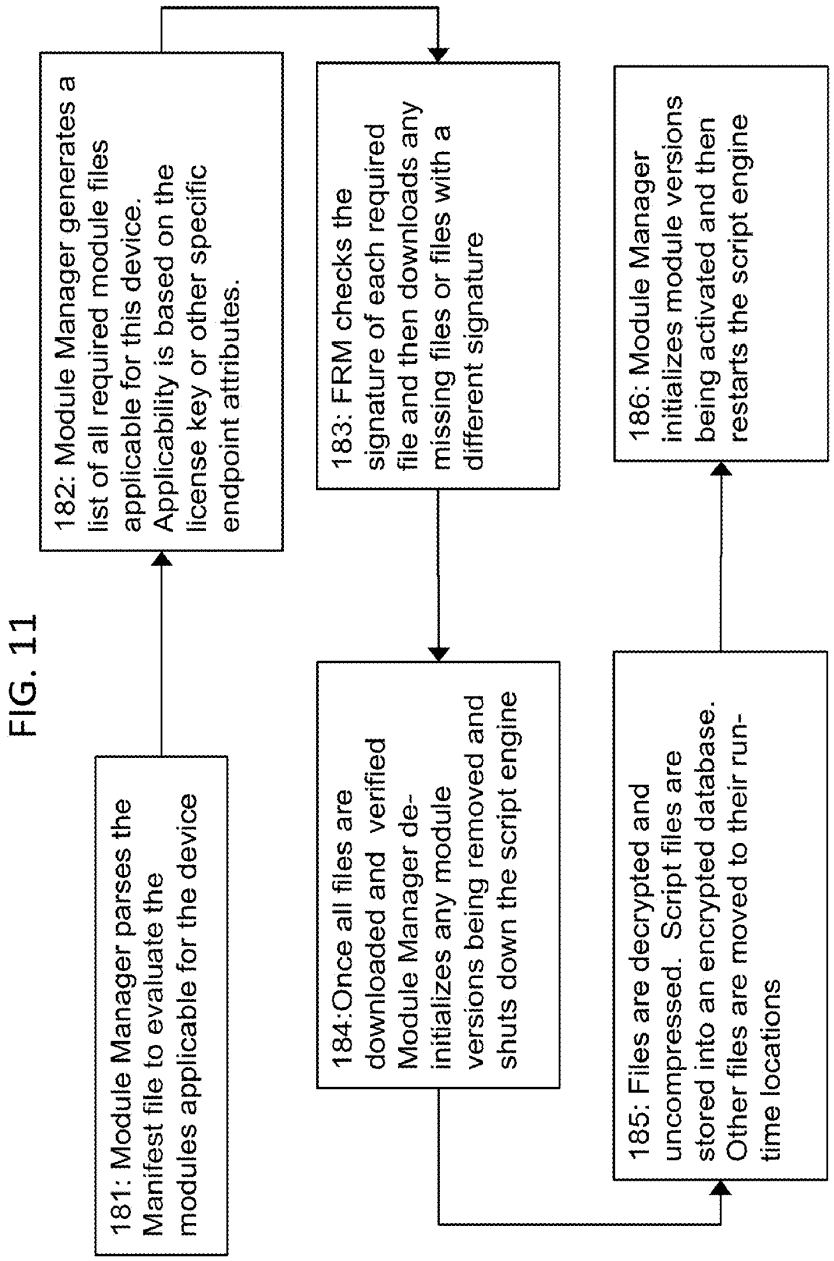
FIG. 7

FIG. 10

Example of Manifest File

```
<Manifest Version= "1.0" AutoUpgrade= "Y">
<LicenseKeys>
 <LicenseKey LicenseID="0001" Key="ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789" ExpireDate="20110714">
 </LicenseKey>
</LicenseKeys>
<Modules>
 <Module ModuleID ="0003" Name="ACDC" Version="3.00.000" ExpireDate="20121231" MSID="0">
  <Applicability><![CDATA[return 0]]></Applicability>
  < ActivateScript > ACDCActivateScript.lua </ActivateScript >
  < DeactivateScript >ACDCDeactivateScript.lua </DeactivateScript >
  <Action ActionID="0000000310" />
  <Action ActionID="0000000311" />
  <Files>
  <File Name="ACDCActivateScript.lua" Crc="30912844bb6eea082a84feec5dab27c05eea498f"
        EncCrc="12376e9f25b34cb2972c48549fc2f75767654321" Target="" Type="Script"" Key="09029895678"
    Compress="N"        Url="/scripts/cust1/ACDCActivateScript.lua"/>
  </Files>
 </Module>
</Modules>
<Files minCoreVersion= "2.0">
 <File Name="policy.pol" Crc="ba4afe9f25b34cb2972c48549fc2f757652fd311"
    EncCrc="65776e9f25b34cb2972c48549fc2f75767654321"          Target="" Type="Policy"
    Key="39029893843" Compress="N Url="/policies/customer/policy.pol"/>
</Files>
</Manifest>
```

FIG. 11


181: Module Manager parses the Manifest file to evaluate the modules applicable for the device
182: Module Manager generates a list of all required module files applicable for this device. Applicability is based on the license key or other specific endpoint attributes.
183: FRM checks the signature of each required file and then downloads any missing files or files with a different signature
184:Once all files are downloaded and verified Module Manager de-initializes any module versions being removed and shuts down the script engine
185: Files are decrypted and uncompressed. Script files are stored into an encrypted database. Other files are moved to their run-time locations
186: Module Manager initializes module versions being activated and then restarts the script engine

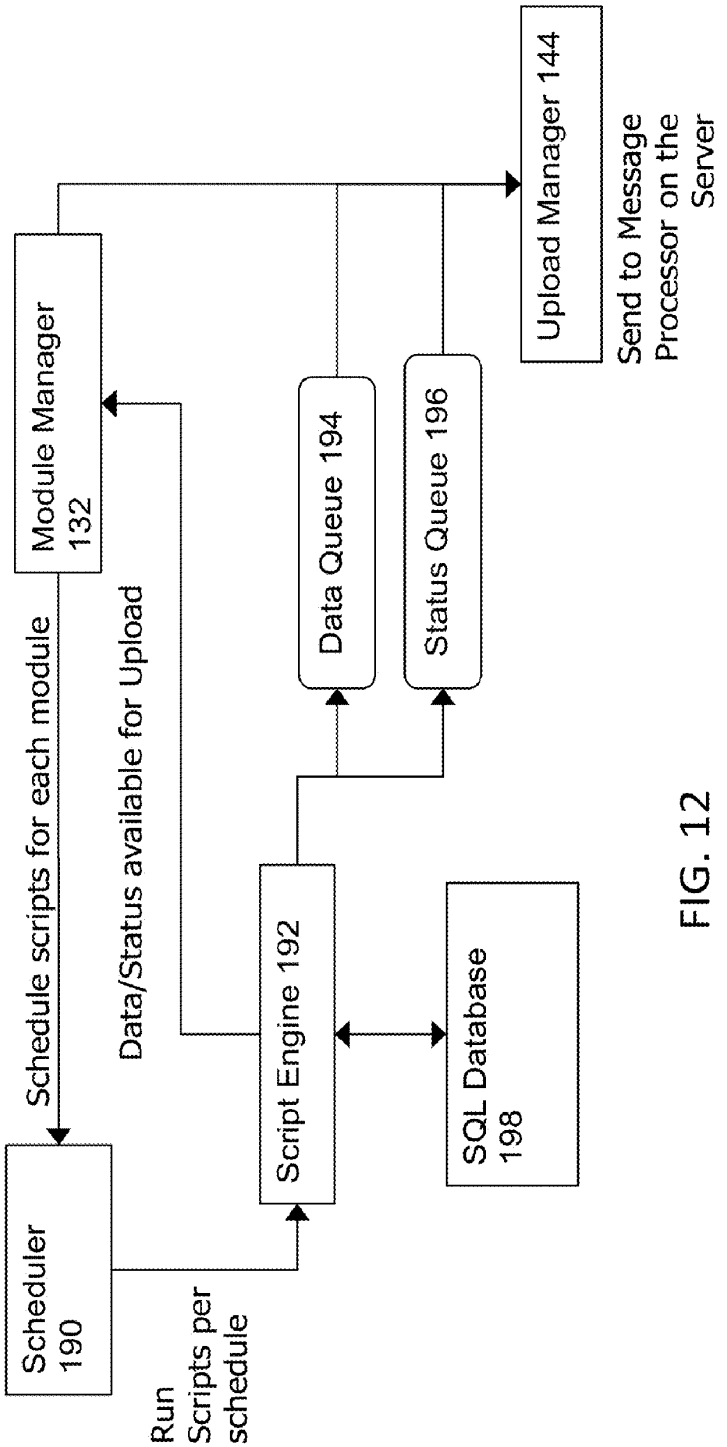

Scheduler 190
Run Scripts per schedule
Schedule scripts for each module
Module Manager 132
Data/Status available for Upload
Script Engine 192
SQL Database 198
Data Queue 194
Status Queue 196
Upload Manager 144
Send to Message Processor on the Server

FIG. 12

PLATFORM FOR DEPLOYMENT AND DISTRIBUTION OF MODULES TO ENDPOINTS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application Ser. No. 61/528,503 filed Aug. 29, 2011, entitled "Saas Platform for Module Development and Delivery to Endpoints," which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] The present invention relates generally to methods and systems for providing mobile device management services and more specifically providing deployment services for deploying software modules to a plurality of managed endpoints.

BACKGROUND

[0003] Mobile device management (MDM) platforms typically allow organizations to control or observe access to corporate resources from mobile devices, such as laptops, tablets, or smartphones. MDM platforms may gather information about mobile devices accessing corporate resources and networks. MDM platforms may also allow organizations to control access to corporate resources based on status information or identity of users or devices. Furthermore, MDM platforms may allow an organization to restrict access to data or applications on the device. For example, a network may use a mobile device management platform to gather information about smart phones that access corporate e-mail. These devices can be restricted, such that, for example, if a user fails to confirm his identity (e.g., entering a password), e-mail apps can be blocked or stored e-mail data may be locked or wiped.

[0004] Because mobile devices change rapidly, it may be difficult for a traditional IT department within an organization to keep abreast of threats and compatibility issues relating to mobile devices used by members of an organization. Accordingly, it may be desirable to outsource MDM tasks to third party vendors that can develop expertise in managing mobile devices. However, unlike management by a traditional IT department of networks containing only wired devices, managing mobile devices may not allow a party managing a device to have access to the physical device. Accordingly, deploying new applications or features to a device can be complicated. One common approach to this challenge is to require a user of a mobile device to manually download certain applications before allowing access to certain corporate resources. For example, an organization may require that a certain e-mail application be used by mobile device to access corporate e-mail.

[0005] However, this may be a cumbersome or obtrusive task, as inexperienced device users must navigate the download and installation process. Applications may need to be upgraded, which may require an invasive, manual update by the user or IT personnel. Furthermore, as organizations seek to add additional features to mobile device management, new applications may need to be separately download and installed. The installation process can be cumbersome and interfere with use of the device. For example, installing applications may require that the device restart after an old version of an application is uninstalled, a new version of the application is reinstalled, and the application is restarted. Furthermore, users may be slow to upgrade or install new applications if they are required to manually install them. This can make it difficult for an organization to timely rollout new applications or new security procedures. Rollout of new applications or updates may therefore be a time intensive process and may be difficult to coordinate within an organization.

[0006] This can cause organizations to avoid adding new or updating existing applications or features in an MDM platform, even though the organization may benefit from a more responsive approach to security or management policies.

SUMMARY

[0007] Applicants have recognized and appreciated that certain problems with the prior art can be solved by providing systems and methods to facilitate the deployment of software modules from a file server to agents on endpoints. Agents can contact a heartbeat server to receive information about a latest manifest file that identifies the software modules that should be installed on the endpoint. The agent can then determine from the manifest information whether new files need to be downloaded and automatically download them. A scripting engine can be used to allow run-time installation of modules without causing significant downtime of the endpoint. The scripting engine can utilize the agent to report back status and data from the software modules to allow modules to be used with a mobile device management platform.

[0008] One embodiment is directed to a method for deploying software modules using device management platform, including providing, at one or more server computers, a file server configured to host a plurality of feature modules, a module manager configured to maintain the plurality of feature modules, and a manifest server configured to maintain a plurality of manifests. Each manifest conveys the feature module components that at least one endpoint should have installed. The method further includes providing one or more agents to a plurality of endpoints for operating thereon, where the agents are configured to communicate with the one or more server computers to receive at least one manifest and subsequently receive a subset of the plurality of feature modules from the file server. An execution engine configured to execute a plurality of scripts contained in the feature modules is provided to endpoint computers for operating thereon. The agents are configured to facilitate reporting of data from the feature modules to the one or more servers.

[0009] One aspect of some embodiments can include providing a web-based portal that allows a user to deploy the subset of feature modules to the endpoint computers. Another aspect of some embodiments can include providing at one or more servers, a heartbeat server configured to receive requests for updates from the plurality of endpoint on a periodic basis. The heartbeat server can be further configured to identify a manifest in response to the requests, when the heartbeat server determines that an endpoint sending a request has not yet received the latest manifest. In another aspect of some embodiments, the module manager can be configured to allow third parties to upload feature modules that can be deployed to the plurality of endpoints using the agents. In another aspect of some embodiments, the agents can be configured to download one or more manifests pertaining to a respective endpoint from the file server. In yet another aspect of some embodiments, the file server can be

hosted by a content delivery network. In still another aspect of some embodiments, the agents can be further configured to associate an identification of the sending endpoint with the reported data from the feature modules. A further aspect of some embodiments can include providing at the server computers, a data store that allows aggregation of data from a plurality of feature modules associated with an individual endpoint. In a further aspect of some embodiments, the data reported can include at least an indication of the status of the deployment of at least one feature module.

[0010] Another embodiment is directed to a software agent stored in non-transitory computer memory comprising software instructions configured to execute on an endpoint and carry out steps of connecting to a heartbeat server at a regular interval to determine whether an updated manifest is available, wherein the manifest identifies a plurality of modules for the agent to install on an endpoint, and downloading a manifest from a file server, in response to an indication that a new manifest is available. The method further includes downloading, in response to the contents of the manifest, a plurality of files relating to the plurality of modules, providing the plurality of files to a scripting engine for execution, and providing messaging services to allow the plurality of modules to report data back to a device management server.

[0011] In one aspect of some embodiments, the agent can be configured to provide the plurality of files to the scripting engine and begin execution of the plurality of modules without requiring the endpoint to restart. In another aspect of some embodiments, the agent can be configured to communicate with the heartbeat server and the device management server using SSL encrypted communication. In yet another aspect of some embodiments, the step of providing messaging services can include an identifier that associates the data from the plurality of data with the endpoint. In still another aspect of some embodiments, the step of providing messaging services can include aggregating data from at least two of the plurality of modules and sending the data to the device management server according to a priority indicated by the at least two modules. In a further aspect of some embodiments, the steps can further include determining which files to download based on an applicability clause contained in the manifest.

[0012] Another embodiment is directed to a system for providing modules to a plurality of endpoints, having software instructions stored on non-transitory computer memory on at least one server, wherein the software instructions include a file server configured to host a plurality of feature modules, a module manager configured to maintain the plurality of feature modules, and a manifest server configured to maintain a plurality of manifests, wherein each manifest conveys the feature module components that at least one endpoint should have installed. The software instructions further include a heartbeat server configured to receive requests from a plurality of agents, wherein each agent operates on an endpoint, and to convey information about at least one latest manifest to each agent when that agent has not yet downloaded the latest manifest and an upload server configured to receive data from modules installed on each endpoint, wherein the modules include those modules identified in the latest manifest.

[0013] In one aspect of some embodiments, the upload server can be configured to provide the received data to device management software. In another aspect of some embodiments, the upload server can be configured to correlate the received data from a plurality of modules operating on an endpoint with the endpoint. In yet aspect of some embodiments, the manifest server can create a manifest in response to a request to deploy at least one feature module to a group of endpoints.

[0014] It should be appreciated that all combinations of the foregoing concepts and additional concepts discussed in greater detail below (provided that such concepts are not mutually inconsistent) are contemplated as being part of the inventive subject matter disclosed herein. In particular, all combinations of claimed subject matter appearing at the end of this disclosure are contemplated as being part of the inventive subject matter disclosed herein.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] The accompanying drawings are not intended to be drawn to scale. In the drawings, each identical or nearly identical component that is illustrated in various figures is represented by a like numeral. For purposes of clarity, not every component may be labeled in every drawing. In the drawings:

[0016] FIG. **1** is a system diagram of an exemplary MDM system for use with some embodiments;

[0017] FIG. **2** is a system and flow diagram of exemplary server components for deploying feature modules to endpoint devices;

[0018] FIG. **3** is a system and flow diagram of exemplary client components for deploying feature modules to endpoint devices;

[0019] FIG. **4** is a system and flow diagram of exemplary client components for reporting status and/or data from feature modules;

[0020] FIG. **5** is a system and flow diagram of exemplary server components for collecting status and/or data from feature modules;

[0021] FIG. **6**A is a screenshot of an exemplary user interface for deploying feature modules;

[0022] FIG. **6**B is a screenshot of an exemplary user interface for deploying feature modules;

[0023] FIG. **7** is an exemplary heartbeat response for use with some embodiments;

[0024] FIG. **8** is a block diagram of an exemplary manifest manager for use with some embodiments;

[0025] FIG. **8** is a block diagram of an exemplary manifest manager for use with some embodiments;

[0026] FIG. **9** is a flow chart of the exemplary use of a manifest manager in some embodiments;

[0027] FIG. **10** is an exemplary manifest file for use with some embodiments;

[0028] FIG. **11** is a flow chart of the exemplary use of a module manager on an endpoint in some embodiments; and

[0029] FIG. **12** is a flow chart of the exemplary use of a scripting engine on an endpoint for use with some embodiments.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

[0030] In some embodiments of the present invention, deployment of applications and data to endpoints can be facilitated by a network-based solution. In some embodiments, a cloud-based approach is used. Agents can be installed on endpoint devices. These agents can facilitate installation of applications/feature modules on-demand. This can allow an organization to selectively add features and

applications to endpoint devices without a complicated manual rollout. This can allow an organization select features for use with a mobile device management (MDM) platform to target the needs of an organization, and allow features to be added or removed as the needs of the organization change. When an MDM system is outsourced to a vendor, or is supplied by vendor, a simplified deployment and distribution approach to applications and features may allow the supplier or operator of the MDM system to offer flexible pricing that is tailored to the features used and the needs of an organization.

[0031] In addition to allowing customers of an MDM platform to selectively purchase and deploy features, in some embodiments, a user of MDM platform may be able to deploy third-party applications and feature packs or develop its own features and applications. The user of the MDM platform (e.g., the organization) can utilize embodiments disclosed herein to facilitate deployment and maintenance of any suitable applications and features. Accordingly, in some embodiments, the MDM platform can facilitate deployment and management of applications and features from one or many software vendors.

[0032] As used herein, a feature module is an executable application, script, or data pack that may be selectively deployed to one or many endpoints, providing a specific function in addition to the core functionality of the MDM system. A feature module may include the executable code and/or data necessary to implement a feature of an MDM platform. These can include new standalone applications or scripts, or may include functionality added to existing applications on an endpoint device or updates to these existing applications. Examples of feature modules include patch management for laptops, Exchange ActiveSync managers, Lotus Traveler managers, user visibility applications, such as applications utilizing Active Directory from Microsoft, etc. Other examples of feature modules include those disclosed throughout this application.

[0033] In some embodiments, feature modules may be supplied by the operator or vendor of the MDM platform. In some embodiments, feature modules may also be supplied by third party vendors or the user of the MDM platform itself. Feature modules may be applications, or modules added to existing MDM applications. In some embodiments, feature modules are additional applications that can be installed on an endpoint. Feature modules may communicate with components on a network, such as communicating with a server component to receive instructions from the MDM platform or to send information, such as status information about the endpoint device to the server, so that the MDM platform can analyze this information from the endpoint. In some embodiments, feature modules may also use the existing reporting and control channels provided by the MDM system and agents on the endpoint to facilitate visibility and management features provided by the feature module.

[0034] As used herein, a user of MDM platform is a customer organization that benefits from the use of an MDM platform. In many instances, the user of the MDM platform is an organization, such as a corporation or campus that includes corporate resources and has individuals within the organization that use endpoint devices.

[0035] As used herein, endpoint devices can include any endpoint devices suitable for accessing or interacting with corporate resources. This may include PCs, laptops, mobile devices, smart phones, tablets, network clients, or the like. Endpoints can be understood to have a processor, memory (which may include magnetic, optical, solid state, or remote memory) for storing data and instructions, and a communication circuit to allow communication with a network, either wirelessly or via a network cable. Modules and software components, such as agents, described herein may be stored in non-transient memory on the endpoint.

[0036] In some embodiments, feature modules can be used for any purpose. In some embodiments, these purposes include allowing security and management policies to be enforced on an endpoint device. In some embodiments, these purposes may include providing visibility information about devices to the MDM platform. In some embodiments, this can allow IT managers to gather and analyze information about the health of endpoint devices connecting to a corporate network and the overall health of the network itself. In some embodiments, feature modules may actively participate in controlling or providing access for endpoint devices to corporate resources, such as e-mail, documents, data, or the like. In some embodiments, feature modules may provide features unrelated to MDM, such as standalone applications that provide non-MDM features that are deployed to endpoints in an organization using the distribution and deployment functionality of the MDM platform.

[0037] Examples of security policies that may be used by the MDM system can include, but are not limited to: requiring users have a passcode to access their device, including defining the minimum passcode strength required; restricting access to devices that have been rooted or are running certain apps (e.g., blacklisted apps, such as file-sharing apps); requiring that the user run certain apps, such as security apps or access clients; remotely wiping the device back to factory settings or wiping a portion of data stored on the device when a device appears compromised, reported missing, or is substantially out of compliance; and limiting access of devices that run versions of a mobile OS or unsupported OS. It should be noted that limiting access can include preventing access, restricting access, changing the nature of the access, or providing access to certain resources, while limiting access to other resources, etc. Examples of resource management policies that may be used with the MDM system include, but are not limited to: alerting the user or an administrator; preventing email access from the device; restricting access to corporate Wi-Fi, VPN, etc.; restricting access to corporate documents on the device; removing all corporate content and configuration from the device; and removing all personal and corporate data and settings from the device.

[0038] While the exemplary MDM systems are described with respect to servers, it should be appreciated that the system can be implemented on a plurality of servers, including cloud-based systems which may deploy the system on a plurality of servers on a dynamic basis, allowing for greater scalability, or a single server that includes many or all of the features disclosed herein. In some embodiments, the servers described herein can be software modules executing on a common or shared server platform. Any suitable portion of the exemplary MDM systems can be executed on one or more processors, which may be separate from the processors executing other portions of the MDM system. A system may be implemented as part of the local IT infrastructure of a corporate environment or as part of a third party solution employed by the organization.

[0039] The components of an MDM system on a server and the feature modules can include software operating on one or

more processors. This software can include software instructions, written in any suitable manner stored on non-transient computer readable media, such as optical, magnetic, solid-state memory devices, or local or remote storage. Computing devices, as described herein can be understood to have one or more processors, memory, and communication devices. Software components can be stored locally or remotely, and be stored in non-transient memory.

[0040] As used herein, a corporate network should be considered a term for any network used by an organization that includes resources, such as Exchange servers, that need to be configured behind a firewall. Generally, a corporate network will include at least one local area network that is used by the organization. This can include campus networks used by universities or other institutions or networks used by corporations or small businesses. The corporate network may be large or small, and some networks may be maintained by a local IT staff. A corporate network can further be composed of multiple networks that interact with shared corporate computing resources, such as BlackBerry Enterprise servers (BES). In some embodiments corporate networks are synonymous with customer premises. It should also be understood that references to corporate, corporations, organizations, etc., may be interchangeable and are not intended to be limited to commercial business organizations. While many of the embodiments described herein are described with reference to corporate environments, these embodiments are merely illustrative and can also apply to the computing environment of any organization that has a need for remote management of network resources.

[0041] Corporate resources can include any of the resources, such as servers, data, credentials, e-mail and exchange services that may be part of, or accessed by, the corporate network. Corporate resources can include those resources that may be ordinarily made available to users of endpoint devices when in compliance with security and management policies.

[0042] By using embodiments of the MDM system described herein, developers of feature modules may be able to develop modules in a development environment and deploy them using the deployment and distribution system of the MDM system when a customer wishes to have them installed on the endpoint devices she manages. To deploy a module, the module can be uploaded to the servers of the MDM platform. Notification of the new module can be conveyed to an agent running on the endpoint device. The agent may be a software component that is required by a corporate network for accessing corporate resources, or maybe a software component voluntarily or manually installed on endpoint devices. The agent can be responsible for communicating with MDM servers, including receiving notification that new or updated feature modules are available for download. Once downloaded to the endpoint, the feature module may be installed and executed by the endpoint.

[0043] By utilizing a modular approach, in some embodiments, the core software for providing MDM services to the endpoint can remain operational and unaffected by the deployed updates or modules. For example, deploying a module that updates features related to e-mail, may not require changes to other MDM features, such as reporting status information or maintaining security and compliance with policies on the endpoint. In some embodiments, modules may provide new features to supplement the operation of core MDM software on the endpoint. Utilizing a modular approach to features may allow an organization to deploy a core MDM component to all devices and then selectively add features to the device, using the deployment system described herein. Furthermore, this may allow an organization to begin with a baseline of MDM features and upgrade the features that the organization uses to meet its needs, without replacing the core MDM software. This can avoid problems that might be caused by changing components that previously worked properly.

[0044] In some embodiments, the feature module may be one or more scripts that can be interpreted by software running on the endpoint. This software may be part of the agent or may be a software module associated or in communication with the agent. This may allow the agent to download new or updated feature modules, while a scripting engine interprets these new modules. In some embodiments, feature modules may be executed/interpreted or updated without requiring the entire endpoint device or all of the MDM components on that device to be restarted. This can provide several advantages.

[0045] First, in embodiments where the device need not be restarted, the end-user of the device may be unaware or not generally inconvenienced by updates. This can allow updates and new modules to be deployed to endpoint without the typical logistic headaches associated with traditional rollouts of software. For example, traditional rollouts may have required system down time, advance notice, restarting of devices on the network, and substantial testing before a rollout. In some embodiments, an update or new module can be deployed to endpoint devices at any time, without requiring system downtime, advance warning to users, or input from users. Second, in some embodiments where MDM components on the endpoint device need not be restarted, MDM components, such as agents or other feature modules, can continue to operate, allowing endpoint agents and management and security policies to continue to function normally during the update process. This can avoid security risks associated with disabling these MDM components. Furthermore, where updates are unobtrusive to the end-user, and where updates can frequently occur without greatly diminishing the user experience, the MDM system may provide advantages to developers of feature modules. For example, substantial beta testing of updates may not be required before deploying new or updated feature modules. In addition, an update can be provided to a group of endpoints and the group of endpoints can act as a real-world beta testing group for these updates. This may greatly reduce the amount of development time required for producing reliable, new or updated feature modules.

[0046] In some prior art systems, features in an MDM platform are provided by core components or by the agents themselves, without using a modular approach. This often required that the agents or core components be removed and replaced whenever an update was needed. Furthermore, new features were difficult to add, because agents often required an update to the entire agent to implement the feature. This could result in down time or security holes during an update process. In some of these prior art systems, because the agent would be effectively replaced, each update may have been a risky installation. Once a deployment is fully operational and the existing features work, there may be impetus to avoid fixing what is not broken. As a result, customers may be less likely to rollout new features or update minor security holes in the MDM system. This can detract from the user experience because desired features may be overlooked, and may detract

from the vendor experience because valuable features are not being deployed even where these features may be desirable to the vendor's customers.

[0047] Agents can be provided for downloading updates in a secure manner from any suitable server resources. For example, communication between agents on endpoints and servers can utilize encryption, such as by utilizing secure socket layer (SSL) or HTTPS communication. The agent can also authenticate any server involved in the delivery of feature modules. This can prevent attacks whereby untrusted servers provide malicious or unsecure feature modules that may leave an endpoint vulnerable. In some embodiments, encryption and authentication can also provide security to server resources by authenticating agents prior to allowing the agents access to modules. This may allow a vendor of an MDM system to prevent unauthorized access to features to require payment for these features. By providing an encrypted and authenticated system, agents and servers can operate confidently that the endpoints and servers are who they purport to be and are secure.

[0048] Agents may also communicate with more than one server. In some embodiments, a cloud-based approach is used, whereby agents can communicate with any number of servers on the Internet to deploy and manage feature modules. By using a cloud-based approach, an MDM system and the deployment and distribution system can be scaled to meet demand. For example, where a new update that broadly applies to a great number of endpoints is needed, a cloud-based content delivery system may be used to scale up to meet the temporary demand for the large number of downloads that are required to deploy the feature modules to all endpoints.

[0049] By providing a system whereby agents can communicate with one or more servers to deploy feature modules, the deployment process can be more easily outsourced. In some embodiments, the servers that the endpoint agents communicate with can be managed by a third-party or vendor that is different than the corporate customer of the MDM platform. This can remove headaches associated with traditional update processes whereby internal IT departments were responsible for rolling out updates. By outsourcing the use updates and the overall deployment process, an organization may not be required to manage server resources, saving cost on hardware and software, as well as allowing personnel (e.g., vendor personnel) with expertise to manage the server resources, without requiring the IT department of a customer to develop such expertise. In some embodiments, customers may not even be required to have an IT presence of their own, allowing the vendor to provide the MDM platform and manage endpoint devices using the MDM platform.

[0050] In some embodiments, third parties may use the same distribution platform as the organization that provides or benefits from the MDM platform. For example, a customer can contract with a vendor to provide core MDM services, allowing the customer organization access to the distribution system described herein and providing agents on endpoints relating to that customer's network. That customer organization can then create its own applications or contract with other parties to provide certain feature applications that may not be available from the MDM vendor. For example, an organization may wish to use a proprietary feature module that provides customized services to endpoint devices. The developer of those customized modules may utilize the distribution mechanisms described herein to deploy applications needed for the new feature. The developer of these new modules can upload the modules to the MDM system. The mechanisms described herein can then be used to convey that module to the agents on the endpoints and distribute the custom module in the same manner as any other feature module.

[0051] By providing a distribution system as part of the MDM platform, the MDM platform can be used to monitor the status of the distribution of new modules. In some embodiments, agents report back the status of new module deployments to the MDM platform, allowing a manager of the MDM platform to monitor, in substantially real-time, the status of deployment of any modules or updates to modules. This can allow the deployment and distribution platform to provide an end-to-end system for deploying feature modules. The administrator using the deployment and distribution system can control the entire lifecycle of the deployment for each endpoint device. The administrator may request that certain groups of devices have certain feature modules installed. Upon submission of the request to have these feature modules installed on the group of endpoint devices, the distribution and deployment system can deploy the applications and data necessary to the endpoint devices. The deployment and distribution system can then receive status information about deployment from agents on the endpoint and report back the status information to the administrator. In some embodiments, this information may be provided in a dashboard that allows an administrator to tell, at a glance, the status of installations of feature modules on a plurality of endpoint devices. In some embodiments, this status reporting can be ongoing, allowing an administrator to determine if a user has disabled or removed certain deployed feature modules.

[0052] An end-to-end system may be a desired feature for generic deployments unrelated to features of an MDM platform. Accordingly, in some embodiments, the distribution system may be used with generic feature modules, such as apps on a smart phone. The module need not be related to the agents in functionality, but the agents can report back status of the deployment to aid in deploying new applications to mobile devices, even where these applications are unrelated to status, security, or compliance. For example, an organization may wish to deploy a sales application to users of Android and iOS mobile devices, to aid them in following sales leads while in the field. In some embodiments, the organization may use the distribution system described herein to assist in rolling out installation of the application to its sales force.

[0053] In some embodiments, the distribution platform can take advantage of the MDM components on the endpoint device and allow modules to be distributed that include compliance policies for using the contents of the module. For example, where a feature module includes a sales application that can access sensitive sales information, the sales application may be sent along with a compliance policy that allows the user of the mobile device endpoint to only access the sales information while on a trusted network, such as a corporate network. In this way, while modules may be data or applications used for any suitable purpose, in some embodiments, the modules may take advantage of the MDM nature of the platform once they are deployed and executed. Furthermore, in some embodiments, in addition to the MDM system reporting the status of the installation process, the module can also be configured to gather and transmit information about the device using the MDM platform.

[0054] In some embodiments, the end-to-end deployment and reporting of installation status of feature modules can

provide further advantages. For example, in some embodiments, the deployment process can be substantially automated using the deployment and distribution system. Once a request is submitted, the feature module will be deployed to a plurality of endpoints and the status of the deployment can be checked at any time by administrator. Furthermore, information about unexpected status of deployments, such as failed installations, can be reported to an administrator as an alert. Because the status of a device and the individual feature modules can be reported, an administrator may be able to monitor status of deployed modules, allowing less-tested modules to be freely distributed without traditional headaches associated with robustly testing modules. For example, an administrator may be able to recall and install feature modules from a plurality of devices if a bug is discovered. This can reduce the development cost of feature modules. Furthermore, because the deployment and distribution system may be used to deploy updates relatively easily, version maintenance of applications on the endpoints can be greatly simplified. An administrator may be able to quickly tell which endpoints are not running updated versions and use the deployment system to send updated versions to these non-compliant endpoints.

Exemplary Architecture

[0055] An exemplary embodiment of the architecture for a deployment and distribution includes agents on the endpoint devices provide certain features to the device. Server components on one or more servers interact with the agents on the devices. On the endpoint, an exemplary embodiment includes a file request manager, a module manager, and an endpoint core agent, which may include a module scripting engine and an upload manager. On the server side, server components include a module manager, a heartbeat server, a manifest server, a file server, an upload server, a message processor, and a plurality of module-specific message handlers.

[0056] Agents may facilitate execution of actions on an end point device. Actions can include commands or instructions that are sent to the endpoint for execution. Examples of action that may be sent to an end point and facilitated by an agent may include a set of scripts or executables asking the agent to install new software on an endpoint device or instructions to wipe the end point device. Agents may also facilitate collection of status information about endpoint devices. Status information may include device attributes about the computing environment of the endpoint device. Exemplary information reported may include the model or hardware information about the device, operating system information, such as version and identity of applications installed, and current network attributes, such as an IP address.

[0057] FIG. **1** shows a high level architecture of the deployment and distribution system as part of an MDM platform. System **100** contains server components and endpoint components. An application server **102** provides server functionality as described herein. This application server can be a SaaS application server providing server functionality in a cloud-based approach, which include a plurality of servers hosting the application or components thereof. Agents **104** on endpoints **105** provide core MDM functionality including allowing actions to be executed on the endpoints and allowing data and status information to be returned to the server. Agents **104** can include a plurality of endpoint applications that provide agent functionality. SaaS application server **102** may transmit actions and modules across a link **106**, which may traverse a network, such as the Internet. Agents **104** may receive these modules and actions and install modules or execute actions accordingly. Agents **104** can also transmit data to the application server across link **108** which may also traverse a network, such as the Internet. This allows agents to report back status of deployments and data collected or generated by the deployed feature modules to application server **102**.

[0058] Embodiments disclosed herein are illustrated with agents that are already installed on endpoint devices. These agents may be previously installed using any suitable means. For example, a user may manually install an agent or have an IT administrator for corporate network install the agent. In some embodiments, users of a mobile device can be directed to a URL for downloading an agent prior to registering with the MDM platform or a corporate network. The URL may allow the user to download and install agent components easily. Once installed, these agents can facilitate MDM functionality and participate in distribution and deployment of feature modules.

[0059] FIG. **2** depicts the server components for facilitating deployment and distribution of feature modules. FIG. **2** illustrates the various server components and the steps that they perform to facilitate distribution of features. File server **110** acts as a repository to store the data/code components that comprise the feature modules. File server **110** allows modules to be elevated to the server infrastructure and later downloaded by agents on endpoints. In some embodiments, file server **110** may be part of a content distribution network (CDN), such as a vendor network (e.g., Akamai). File server **110** may use a cloud-based approach allowing mirroring and dynamic load balancing to facilitate upload and download of module content. Files stored on file server **110** can include the data and instructions that constitute a feature module.

[0060] Server **102** also includes module manager **112**. Module manager **112** can be a software component that keeps track of all modules and their components as they are uploaded and maintained. Module manager **112** can map the locations (e.g., URLs or other network locations) of these components. In some embodiments, it may also participate in determining which endpoints should receive feature modules that are uploaded to file server **110**, while in other embodiments, this aspect may be handled by manifest server **114**. In some embodiments, by separating module manager **112** and manifest server **114** from file server **110**, greater scalability can be achieved. File server **110** can be an off-the-shelf CDN component, while module manager **112** and manifest server **114** can be specialized software components developed to facilitate the deployment and distribution system of an MDM platform. In some embodiments module manager **112** can facilitate maintaining modules on a multitenant database. Multiple customers can share server resources for an MDM platform. Module manager **112** can include different or shared modules for different customers and or groups. Module manager **112** can also keep track of multiple versions of modules to assist in operating endpoints to new versions of modules. In some embodiments, multiple versions of a module can be used to allow a select group of endpoints to participate in a beta test of updated versions of a module.

[0061] In some embodiments, module manager **112** and/or manifest server **114** can provide a simplified mechanism for pushing out existing modules to endpoints. An administrator may request that an update be sent to managed endpoints. The module manager can identify the components of the module

and the location of these components, as well as identify which groups of devices may be authorized to access these components. The module manager can provide this information to the manifest server in response to a request from manifest server **114**, or may push this information to manifest server **114**.

[0062] Manifest server **114** manages pushing updates to endpoints by creating manifests that associate modules to be distributed to the endpoints with the groups of endpoints to receive the modules. Manifests can be small files that can be easily distributed to endpoints. A manifest can associate the locations of module components with an identification of the group of endpoints that should receive the update. In some embodiments, the agent makes the decision of whether each individual endpoint belongs to that group, allowing greater scalability, as the manifest server **114** may not need identify the individual endpoints to receive the update. In other embodiments, the manifest may identify the specific endpoints to receive a module. In some embodiments, a manifest is a file that has XML or metadata that identifies the locations of the components of modules to install and the attributes of a group of endpoints (or the individual endpoints) for which the modules should be installed. Endpoints can work with heartbeat server **116** to determine which manifests apply to each endpoint. Manifests may also include any security credentials, such as encryption keys, necessary for an endpoint to access the module components. Manifests can be distributed to agents across the network in a secure manner to protect these credentials. Manifests may also include a timestamp identifying the last change to the manifest or creation date and time.

[0063] When a manifest server has created a new manifest, it may upload the manifest to file server **110** to allow agents to subsequently download the manifest. In addition, manifest server **114** will inform heartbeat server **116** of the new manifest, and provide the timestamp of the manifest and, in some embodiments, decryption credentials to allow an agent to decrypt encrypted manifest stored on the file server. AES encryption may be used.

[0064] Heartbeat server **116** provides an interface from the server components to the agents **104** (and specifically file request manager **130**, within each agent) across the network **128**, which may include the Internet. In some embodiments, heartbeat server **116** uses a lightweight protocol, whereby each time an agent checks in with the heartbeat server (which may be at a regular interval, such as every 15 minutes) the heartbeat server will identify any manifests that may relate to the endpoint and/or identify a timestamp of the last change to that manifest. The agent may keep track of the manifest or manifests that it has utilized in the past and compare the timestamps of the past manifest to determine if a new manifest is waiting. In some embodiments, the heartbeat server reports the timestamp or ID of the last manifest to the agent, allowing the agent to make the determination of whether a new manifest should be downloaded.

[0065] It should be noted that in some embodiments, most times an agent checks with the heartbeat server **116**, the timestamps or IDs will match, because updates may be less frequent than the requests for manifest information by agents (called heartbeats). By supplying timestamps, the heartbeat server can inform agents of new or updated manifests without needing to itself know the status of modules on the endpoint. This may allow greater scalability, as a heartbeat server may service many thousands of endpoints on a regular, periodic basis (e.g., agents check in every 10 minutes).

[0066] In some embodiments, when an agent identifies a new manifest based on the timestamp information received from heartbeat server **116**, that agent can then go to the file server or CDN to request the actual manifest. This allows the bandwidth intensive operation of sending manifests to endpoints to be offloaded from the heartbeat server, allowing greater scalability. The information provided from the heartbeat server can include the URL of the manifest, the timestamp of the manifest, and any include decryption keys needed to access that manifest. This allows the manifests to be stored in an encrypted form allowing secure transport to agents across the Internet.

[0067] FIG. **2** also depicts the steps used in uploading and distributing new modules. Steps **120** through **126** are carried out by respective components **110** through **116**. At step **120**, module files, including scripts, libraries, data, or other code components are uploaded to a file server **110**. This can include uploading to a CDN. These components may be encrypted. At step **122** when a module is created (or an update or new version is created), module manager **112** will provision the new module by associating the various components and identifying the location of these components on the CDN. Module manager **112** can make this information available to manifest server **114**. At step **124**, manifest server **114** monitors requests to distribute modules. These requests may come from administrators explicitly or may be responsive to updates to existing modules as identified by module manager **112**. Manifest server **114** then creates a manifest file that associates module files with groups of endpoints to receive the module. At step **126** once a manifest has been generated manifest server **114** informs heartbeat server **116**. This informs the heartbeat server **116** that the timestamp of a manifest has changed. The next time relevant endpoints check in with the heartbeat server, heartbeat server **116** will inform these endpoints that a new manifest is waiting, allowing agents on these endpoints to seek out and download new module components.

[0068] The heartbeat server can respond to requests from agents on endpoints that request information at regular intervals. Heartbeat server can verify the identity of the client agent and formulate a response to the client that includes billing and device specific information. The request sent to the heartbeat server can, in an exemplary embodiment, include the following information:

[0069] FL-DEVICE-ID: the device id;

[0070] FL-BILLING-ID: the billing account associated with the endpoint;

[0071] FL-ACCESS-KEY: the unique access key generated for each device when it starts getting managed;

[0072] FL-LICENSE-KEY: the current active license key for MDM services; and

[0073] FL-TIMESTAMP: the timestamp of the last ping.

[0074] In this exemplary embodiment, the heartbeat server determines from the timestamp received from the endpoint whether information about a new manifest should be transmitted to the agent. FIG. **7** depicts an exemplary format for a response from the heartbeat server.

[0075] FIG. **3** depicts the components of an agent on an endpoint that facilitate distribution and appointment of feature modules. It should be appreciated that agents may include other components related to the MDM platform which are not shown. File request manager **130** is responsible

for communicating with heartbeat server **116** to determine whether new manifests are available. In some embodiments, agents periodically communicate with a heartbeat server for other MDM purposes, such as receiving information about configuration changes. File request manager **130** can be invoked to handle information about manifests waiting on the file server. In some embodiments, file request manager **130** will evaluate the timestamp of the manifest, as identified by the heartbeat server, to determine if new content is available or be explicitly told by the heartbeat response that a new manifest is waiting. If new content is available, file request manager **130** will report this information to module manager **132**. In some embodiments, the file request server will also request the files from the file server.

[0076] Module manager **132** resides on the agent and is responsible for managing the components of feature modules. In some embodiments, in response to a new timestamp on the manifest, module manager **132** can reach out to file server **110** to download and decrypt the waiting manifest. In some embodiments, the module manager can receive the new manifest from file request manager **130**. Module manager **132** can then evaluate the contents of the manifest.

[0077] In some embodiments, a manifest includes a list of all modules that should be installed on the endpoint. Module manager **132** can determine whether any of the components listed in the manifest are different than the components already stored on the endpoint. For example, where a manifest identifies several modules, a new manifest may identify only a few modules that are not already on the endpoint. In some embodiments, a manifest can include a hash or version information of the contents of a module component to allow module manager **132** to determine if the content is the same as that is already installed. In some embodiments, this hash information can be provided by the file server.

[0078] In response to a changed module component identified in the manifest, module manager **132** can communicate with file server **110** across a network, such as Internet, to download these modules and utilize encryption keys stored in the manifest to decrypt the contents. In some embodiments, module manager **132** may need to negotiate with file server **110** to determine which module content has changed and should be downloaded, while in others, this information is provided in the manifest. In some embodiments, module manager **132** only downloads changed content to save on network bandwidth, allowing file server **110** to be more scalable and to reduce the overhead data used by each endpoint.

[0079] Once all module components have been downloaded, module manager **132** can forward the updated module to a scripting engine on the endpoint to allow the module to be installed and executed.

[0080] FIG. **3** also depicts a flowchart of the steps for deploying a module on an endpoint. Step **134** is carried out by file request manager **130**. Steps **135-139** are carried out by module manager **132**. At step **134** file request manager **130** receives timestamps associated with manifests during a heartbeat communication. If the timestamp does not match the timestamp of the last manifest, file request manager **130** can invoke module manager **132** to handle the new manifest. At step **135**, the module manager **132** utilizes the encryption key from the heartbeat communication and the URL of the new manifest to download the new manifest from a file server. Module manager **132** then evaluates the content of the manifest to determine if any modules identified in the manifest apply to this endpoint. If so, at step **137**, the module manager determines whether any of these module components have not already been installed on the endpoint. An SHA or MD5 hash of the contents of the module can be provided in the manifest allowing module manager **132** to compare the contents of the module against those already installed in the endpoint. In some embodiments, modules have associated timestamps that can also be compared. Similarly, in some embodiments, version information can be provided in the manifest to allow module manager **132** to evaluate whether the latest version is been downloaded.

[0081] At step **139**, module manager **132** downloads all module files that are not already installed or whose version or hash information conflict with files already stored on the endpoint. Once this download has been completed, a complete module image has been stored on the endpoint and can be released to a scripting engine for installation and execution of the new/updated feature module.

[0082] In some embodiments, the manifest can also be used to remove modules. In some embodiments, a manifest is a complete list of all modules that should be installed on an endpoint. This can be conveyed either through a single manifest for an organization that allows agents to identify the group to which the receiving endpoints belongs, allowing different endpoints within an organization to have different modules, or through a manifest file that includes only the modules that should be installed on the receiving endpoint. The manifest can include specific instructions to delete modules, or may simply exclude modules that are no longer to be used. The module manager can respond to this information by removing any modules that are not identified in the manifest. This can provide a simple way of controlling the modules that are used by a given endpoint.

[0083] In some embodiments, once the module manager has downloaded all components, it can instruct the scripting engine that carries out the scripts in modules to temporarily shut down or restart, allowing the new image of modules to be loaded into the scripting engine. In some embodiments, the scripting engine will only be halted or restarted with respect to modules that have been updated. This can allow certain functionality to continue even during the restarting process. It should be noted, that this restarting process can be minimally invasive, without requiring restarting of the endpoint device. In some embodiments, the user of the endpoint device may not even be aware of the update to its modules and the loading and restarting of the scripting engine can be a background process. This can allow real-time updates to modules.

[0084] FIG. **4** depicts the end point components of a reporting system that can be used as part of the module distribution and deployment system. Reporting component can allow deployed modules to report back status related to their installation as well as report back information to server components during normal execution. This can allow developers of modules to take advantage of APIs as part of the MDM platform that allow reporting of information gathered by modules. For example, modules may be developed that determine status information about an endpoint, apply compliance policies to an endpoint, apply configuration to an endpoint, facilitate access of an endpoint to corporate resources, etc. The MDM platform can be utilized to facilitate many of these functions allowing modules to easily report back data or status information to the MDM platform. In some embodiments, modules may take advantage of these mechanisms during normal operation of the feature module, while in other embodiments certain feature modules may utilize the report-

ing mechanisms shown in FIGS. **4-5** to report back installation status, allowing feature modules include generic applications that may or may not be related to MDM functionality. In some embodiments, feature modules may wrap basic reporting features around non-MDM applications to allow basic reporting of status of the nation and installation to server components. For example, where a non-MDM feature module is deployed, such as a sales application, the feature module used to deploy this sales application can include basic reporting functionality that utilizes the APIs of MDM agents on the endpoints to report back successful installation or other status information about the application. Data reported back can include any data that the module is configured to report.

[0085] As part of the method for reporting back data or status information from module components, the messages can include identifier information for associating the module data with a particular endpoint or agent. This can allow data to be correlated at the server, across a plurality of feature modules. For example, multiple feature modules may report back status or information a device. By collecting and correlating the data returned by multiple modules, the server can present a more complete holistic view of devices to a user of the MDM platform, allowing data from multiple modules to be presented as part of a single device view. Similarly, it may allow compliance policies to be adapted at the server by deploying new modules to report back new kinds of data that may be desirable for consideration of compliance status.

[0086] Endpoint agent **140** provides APIs that allow modules to report information back to server components. Scripting engine **142** is responsible for executing the scripts that are contained in feature modules. As part of the execution process, modules may collect data, such as status information about the endpoint device or any other generic data that the module wishes to report back to a server component. The scripting engine **142** facilitates the collection of this data as it executes each script in the feature modules. In some embodiments, the scripting engine is a Lua interpreter, allowing simple or complex scripts to be easily developed to implement robust module functionality.

[0087] The scripting engine **142** passes this collected data to an upload manager. Each of these is a software component that resides on the endpoint and may be part of the agent. Upload manager **144** is responsible for aggregating, normalizing, formatting, or other processing of data prior to upload. Upload manager **144** then forms aggregated message packets that contain information gathered by a plurality of modules to be passed to the server across a network **128**, such as the Internet, to server components on server **102**. These packets can include one or more identifiers that associate the collected data with the agent or endpoint. This allows server components of feature modules to correlate collected data with the endpoints and more easily share data with other applications or the MDM platform. This can allow reporting of data from endpoints to be used for various purposes on the server by applications, policies, or for visibility provided to a dashboard or for other viewing by an administrator using the MDM platform.

[0088] FIG. **4** also shows the steps involved in passing data from feature modules executing on and points to the server. At step **145** endpoint agent **140** provides APIs that can be accessed by a developer of a feature module to pass data from the endpoint to server. By invoking these APIs, at step **146**, the feature modules executing in the scripting engine can collect data and command the scripting engine to report data back to server.

[0089] At step **148**, upload manager **144** aggregates data collected by modules executing on the scripting engine and prepares it for upload. In some embodiments, this can include aggregating, identifying higher priority messages, reformatting messages to optimize transfer, compressing data, etc. At step **148**, aggregated message packets are formed from the data queued for upload by the scripting engine. These message packets are then queued by priority to facilitate transfer to the server. Subsequently, upload manager **144** passes these messages across the network **128** to server components on servers **102** using a secure connection, such as HTTPS. In some embodiments, upload manager **144**, as part of the aggregation process, authenticates the modules reporting the data. This can provide an additional layer of security to prevent untrusted modules from reporting data back to servers **102**. This authentication process can utilize any suitable authentication technique is known in the art. Similarly, upload manager **144** can authenticate servers **102** prior to transmitting data across network **128** to ensure that data collected by modules on an endpoint is not reported to untrusted entities. The upload manager can also receive confirmation from the server that the data has been successfully uploaded.

[0090] FIG. **5** shows the corresponding components on the server for receiving messages from modules on endpoints. Messages from the upload manager can be received across a network **128** by upload server **150**. Upload server **150** can be a software module running on cloud-based server infrastructure that handles incoming messages from a plurality of upload managers executing on a plurality of endpoints. Upload server **150** handles incoming message packets from the upload managers, breaks them into individual messages and adds them to queues **152**. Message processor queues **152** can include a plurality of queues that can be handled by a plurality of message processors **154**. Message processor queues **152** can include various queues for different priorities and types of data uploaded by the upload managers.

[0091] Message processors **154** check message processor queues **152** to determine that new data has been uploaded. Message processors **154** then handle these messages based on priority and begin processing this data. Message processors **154** determine the type of message received and the identity of the modules that sent the data. Message processors **154** may also determine the identity of the endpoints/agents that hosted the module. Message processors **154** determine which individual message handlers **155** should receive the uploaded data and then parse out the appropriate data to send to the appropriate message handlers. Individual message handlers **155** can include customized message handlers that have been developed as a server component for feature modules that have been installed on endpoints. For example, where a feature module has been developed to send back status information about e-mail applications, individual message handlers **155** can be correspondingly developed to receive e-mail status information and report it to a visibility dashboard as part of MDM platform. It should be appreciated that individual message handlers **155** may include software components that perform any suitable task with the messages received. These can be customized based on the individual feature modules deployed. In some embodiments individual message handlers **155** are developed simultaneously with the individual feature modules to provide rich module functionality. In some

embodiments, individual message handlers **155** may be provided by third parties that develop third-party feature modules.

[0092] Message handlers **155** or message processors **154** can also store the data from the messages in a shared data store **156** (which may store the uploaded data in native or processed form). When stored, the information can be associated with the individual endpoints that sent the data, allowing the data pertaining to a single endpoint to be gathered from a group of feature modules (which may be unrelated or related) and shared and accessed by an administrator or MDM application. This can allow feature modules to add features to existing MDM applications. The shared data store can allow a plurality of applications on the server side to access and mine the data returned. This information be accessed by user interface **159**, which can include a dashboard for viewing attributes and data pertaining to groups or individual devices. This interface may be presented to a user via a webpage, allowing an administrator to application server **102** via a browser and manage endpoints and view information about them that is received from feature modules.

[0093] FIG. **5** also shows a flowchart for the handling of incoming messages at server. At step **156**, messages received by upload server **150** can be parsed into individual data messages from the packets and placed into message processor queues **152** according to priority. At step **157** message processors running on the server can read the messages that have been placed in the message processor queues **152**. Message processors **154** will continue polling these queues on a regular basis as data may be arriving at any time. At step **158** message processors parse the messages, identify the appropriate message handler that should receive the message, and invoke the message handler if it has not already been invoked. These message handlers can be responsible for substantively processing this data and persisting this data on the server, such as aggregating and storing data reported from a plurality of modules on a plurality of endpoints to provide visibility services about these endpoints.

[0094] FIG. **6**A shows an exemplary screen shot that would be presented to a developer when elevating a module to the server for deploying a feature module to endpoint devices. This dashboard allows the administrator or developer to define to configure the module for deployment. Information added using this screen can be handled by the manifest server and module manager. The interface allows selection of a code name, as well as a name for the module to be deployed. The developer can also select a category and product bundle to associate with the module. This can allow modules to be bundled so that they can be deployed together, allowing a simplified pricing and rollout strategy to be used when deploying modules to customers. The developer can select a version and define SQL or XML statements that should be included in a manifest when the module is deployed to endpoints. Once a module is uploaded, the information input can be used to create the manifest once a bundle or module is selected for deployment to endpoints.

[0095] FIG. **6**B is an exemplary portion of a screenshot that would be presented for managing a plurality of modules. As shown in FIG. **6**B, a plurality of modules have been created and uploaded to the server infrastructure. This table reflects the information that can be stored by the module manager to facilitate deployment of these modules.

[0096] Examples of modules that could be uploaded using the interface in FIGS. **6**A and **6**B can include but are not limited to: base module for laptop management (e.g., a set of APIs for allowing management that can be presented to other modules); device visibility for laptops and computers (e.g., collecting hardware information and information about the running software and the operating system); patch management (e.g., verifying that certain patches have been applied); data protection (e.g., verifying that encryption and that peripheral memory devices are being monitored); endpoint security protection (e.g., verifying that firewalls and antivirus measures are being employed and that virus definitions are up to date); user authentication (e.g., verifying that a mobile device, such as a smart phone, requires a password); Exchange ActiveSync Manager (e.g., integrating with Active Directory); Lotus Traveler Manager; and BlackBerry Enterprise Server Manager. Custom modules can include, but are not limited to modules that help manage desktops, laptops, smart phones, tablets, or other mobile devices.

[0097] FIG. **7** depicts an exemplary XML response for conveying information about a manifest file from the heartbeat server to an endpoint agent once a new manifest is available, such as when a new module is ready to be deployed to the endpoints. Exemplary fields include a timestamp for the manifest, a license value, URLs to download new manifests, manifest file names and versions, keys for encryption and hash IDs for the manifest file, groups that correspond to the manifest (e.g., identifying the criteria for selecting and points for which the manifest applies), and identifications of users and groups for which the manifest may apply.

[0098] FIG. **8** shows the internal components of an exemplary manifest server **114**. There are two primary ways by which a manifest can be created. First, if a module has been uploaded or updated, the module manager can inform the manifest server of the change so that the manifest server can propagate the change to any appropriate endpoints. Second, batch requests to deploy modules to devices can also be received. Examples of batch requests may include adding modules to devices where the module has already been uploaded into the file server, but the functionality is now being added to a new group of devices.

[0099] Manifest server listener **160** provides software means to receive requests from module manager **112** to update manifests when a module has been updated or has been uploaded for the first time. That is, changes to modules may prompt updated manifests for devices that should receive those updated modules (e.g., those that have older version of the module or those that are specified when the module is elevated to the server). Batch requester **162** provides a means by which the manifest server may listen to a user interface, such as a web based dashboard that allows an administrator to add functionality to groups of devices, and to push those changes in authorization of devices to the deployment and distribution system. In some embodiments, batch requester **162** will periodically check the authorizations of devices to determine if changes to the manifests for those devices need to occur.

[0100] Manifest server listener **160** and batch requester **162** begin the manifest creation process by placing manifest requests in the manifest request queue **164**. Manifest generator **166** will observe manifest request queue **164** to determine if new manifests need to be created. For example, when an updated module results in a request from module manager **112** for a new manifest, manifest generated **166** may deter-

mine which groups of devices had previous versions of that module and determine that those devices should receive an updated manifest.

[0101] FIG. **9** shows the processing flow **170** for generating manifests for groups of devices using an exemplary manifest server. At step **172**, the manifest server gets a list of modules and checks the criteria for determining availability of the module. Criteria can include a list of customer organizations that should receive the module based on payment authorization. At step **174**, applicability clauses are generated for each of the modules. In some embodiments, an applicability clause is an identification of which types of devices should receive the module. In some instances, the applicability clause can identify whether a module is a beta module for limited distribution or a generally available module that is ready to be distributed to all endpoints within an organization. At step **176**, for a module is that are for general availability, the manifest server gets details about the module files and the version information to prepare inclusion of that module to a general class of endpoint devices. At step **178**, if a customer has access to beta modules and the modules in question are beta modules, the manifest server will obtain details about the module and prepare a manifest file that includes identification of the beta files. At step **179**, one or more manifest files are generated and stored on the file server. The heartbeat server is made aware of the new manifests and it can begin informing the appropriate endpoint agents during the next heartbeat.

[0102] FIG. **10** shows an exemplary manifest file that may be retrieved by an agent in response to a notification from a heartbeat server. Once a new manifest file is discovered, the agent on the endpoint can go to the URL in the heartbeat response and request a new manifest from the file server or CDN. Using decryption keys received from the heartbeat server, the manifest file can be decrypted. Exemplary manifest file includes XML information allowing the agent on the endpoint to determine which files need to be retrieved from the file server to complete installation of a module. Manifest file may include a manifest version and license keys. Inclusion of license keys can allow a manifest file to easily distribute new license keys to agents on endpoints. This can allow an MDM platform to be implemented on a subscription basis or allow features to be turned on automatically using license keys. Once a customer upgrades this license status, new license keys can be distributed to agents on endpoints managed by a customer.

[0103] A manifest file also includes encryption keys, filenames, locations of files, checksums or hashes of the files, and other information to allow a module manager on the endpoint to determine which files it does not yet have, information on where to obtain the file, and how to decrypt these files. A manifest file can also include information about each module that should be installed, including the applicability clause of the module to allow each endpoint receiving the manifest to determine if it should install any beta module, if applicable. An applicability clause may also be used to identify which groups of endpoints within an organization should install which modules, in organizations that include a plurality of groups. Module IDs, names, versions, and expiration dates can also be included.

[0104] FIG. **11** shows the detailed flow of the operation of a module manager on an endpoint agent. At step **181** the module manager receives and parses a manifest file to determine which modules are applicable for that endpoint device. This can include consideration of the applicability clause for each module to determine if the module applies to this endpoint. In some embodiments, a manifest file may include a manifest for all devices within an organization. While large groups of endpoint devices in an organization will share modules, there may be multiple groups. The manifest file can indicate which groups should receive which feature modules. The module manager can parse the manifest file to determine which modules apply to this device based on knowledge of the group to which the device belongs or based on specific identification in the manifest file.

[0105] At step **182**, the module manager generates a list of all required module files that apply to the device. This applicability determination may be based on the license key associated with the endpoint device, attributes of the endpoint device that associate that device with a certain group, an explicit group ID of a group to which the endpoint belongs, etc. At step **183**, the file request manager on the endpoint agent checks a signature of each required file identified in the manifest and then downloads any missing files or files with a different signature. Once the file request manager has downloaded all files and verified the authenticity these files, the module manager can work with the scripting engine to halt execution of any prior versions of the module being updated. Scripting engine can be shut down temporarily while old files are removed.

[0106] At step **185**, files received from the file server are decrypted and uncompressed the script files or module can then be stored locally in an encrypted database and uncompressed form to allow the scripting engine to decrypt and execute these script files quickly, while ensuring security. Other files may also be moved to runtime locations to provide fast local access for execution of module components. At step **186**, module manager initializes the new module version and restarts the scripting engine to allow the new module to begin providing normal functionality. In this manner, new modules can be updated or installed without requiring a substantial installation process or substantial downtime to mobile device. In some embodiments, the user device may not even be aware that updated modules have been installed and have begun executing.

[0107] FIG. **12** shows an exemplary architecture for the components responsible for execution of the scripts of a feature module for use with some embodiments. Scheduler **190** is a software module on the endpoint that determines, based on the metadata associated with a feature module, schedules execution of the scripts of a feature module. For example, metadata may identify the frequency with which to execute a script. Module manager **132** can configure the schedule of the feature module scripts based on the information it gathers when installing the feature module. Schedule **190** can schedule the script to execute on script engine **192**. Script engine **192** can interpret and execute any scripts or other executables in the feature modules. To execute scripts, the script engine may access a SQL database that contains data and code related to the feature modules.

[0108] In response to executing instructions of the feature module, the script engine may gather status information about the endpoint and/or create data that should be shared with corresponding modules on the server. Script engine **192** may place this information in the status queue **196** and data queue **194**, respectively. In some embodiments, the script engine may inform the module manager that the data is ready for upload. For example, in some embodiments, when a script is loaded for the first time, confirmation of installation may be

provided to the status queue for upload to the server and the script engine may inform the module manager, as well. Upload manager **114** can access queues **194** and **196** to facilitate upload of status information and other data.

[0109] Having thus described several aspects of at least one embodiment of this invention, it is to be appreciated various alterations, modifications, and improvements will readily occur to those skilled in the art. Such alterations, modifications, and improvements are intended to be part of this disclosure, and are intended to be within the spirit and scope of the invention. Accordingly, the foregoing description and drawings are by way of example only.

What is claimed is:

1. A method for deploying software modules using device management platform, comprising:

providing at one or more server computers,

a file server configured to host a plurality of feature modules,

a module manager configured to maintain the plurality of feature modules, and

a manifest server configured to maintain a plurality of manifests, wherein each manifest conveys the feature module components that at least one endpoint should have installed;

providing one or more agents to a plurality of endpoints for operating thereon, configured to communicate with the one or more server computers to receive at least one manifest and subsequently receive a subset of the plurality of feature modules from the file server; and

providing an execution engine to endpoint computers for operating thereon, configured to execute a plurality of scripts contained in the feature modules,

wherein the one or more agents are further configured to facilitate reporting of data from the feature modules to the one or more servers.

2. The method of claim **1**, further comprising:

providing a web-based portal that allows a user to deploy the subset of feature modules to the endpoint computers.

3. The method of claim **1**, further comprising providing at the one or more servers, a heartbeat server configured to receive requests for updates from the plurality of endpoint on a periodic basis.

4. The method of claim **3**, wherein the heartbeat server is further configured to identify a manifest in response to the requests, when an endpoint sending a request has not yet received the latest manifest.

5. The method of claim **1**, wherein the module manager is configured to allow third parties to upload feature modules that can be deployed to the plurality of endpoints using the one or more agents.

6. The method of claim **1**, wherein the one or more agents are configured to download one or more manifests pertaining to a respective endpoint from the file server.

7. The method of claim **1**, wherein the file server is hosted by a content delivery network.

8. The method of claim **1**, wherein the agents are further configured to associate an identification of the sending endpoint with the reported data from the feature modules.

9. The method of claim **1**, further comprising providing at the one or more server computers, a data store that allows aggregation of data from a plurality of feature modules associated with an individual endpoint.

10. The method of claim **1**, wherein the data reported includes at least an indication of the status of the deployment of at least one feature module.

11. A software agent stored in non-transitory computer memory comprising software instructions configured to execute on an endpoint and carry out the following steps:

connecting to a heartbeat server at a regular interval to determine whether an updated manifest is available, wherein the manifest identifies a plurality of modules for the agent to install on an endpoint;

downloading a manifest from a file server, in response to an indication that a new manifest is available;

downloading, in response to the contents of the manifest, a plurality of files relating to the plurality of modules;

providing the plurality of files to a scripting engine for execution; and

providing messaging services to allow the plurality of modules to report data back to a device management server.

12. The software agent of claim **11**, wherein the agent is configured to provide the plurality of files to the scripting engine and begin execution of the plurality of modules without requiring the endpoint to restart.

13. The software agent of claim **11**, wherein the agent is configured to communicate with the heartbeat server and the device management server using SSL encrypted communication.

14. The software agent of claim **11**, wherein the step of providing messaging services comprises including an identifier that associates the data from the plurality of data with the endpoint.

15. The software agent of claim **11**, wherein the step of providing messaging services comprises aggregating data from at least two of the plurality of modules and sending the data to the device management server according to a priority indicated by the at least two modules.

16. The software agent of claim **11**, wherein the steps further comprise determining which files to download based on an applicability clause contained in the manifest.

17. A system for providing modules to a plurality of endpoints, comprising software instructions stored on non-transitory computer memory on at least one server, wherein the software instructions comprise:

a file server configured to host a plurality of feature modules,

a module manager configured to maintain the plurality of feature modules,

a manifest server configured to maintain a plurality of manifests, wherein each manifest conveys the feature module components that at least one endpoint should have installed;

a heartbeat server configured to receive requests from a plurality of agents, wherein each agent operates on an endpoint, and to convey information about at least one latest manifest to each agent when that agent has not yet downloaded the latest manifest; and

an upload server configured to receive data from modules installed on each endpoint, wherein the modules include those modules identified in the latest manifest.

18. The system of claim **17**, wherein the upload server is configured to provide the received data to device management software.

19. The system of claim **17**, wherein the upload server is configured to correlate the received data from a plurality of modules operating on an endpoint with the endpoint.

20. The system of claim **17**, wherein the manifest server creates a manifest in response to a request to deploy at least one feature module to a group of endpoints.

* * * * *

Ex. 1004

U.S. Pub. No. 20120278454
to Stewart *et al*.
(Stewart)

US 20120278454A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2012/0278454 A1**

(43) **Pub. Date: Nov. 1, 2012**

Stewart et al.

(54) **GATEWAY DEVICE APPLICATION DEVELOPMENT SYSTEM**

(76) Inventors: **Brett B. Stewart**, Austin, TX (US); **Dirk D. Heinen**, Austin, TX (US); **Margo a. Baxter**, Austin, TX (US)

(21) Appl. No.: **13/450,625**

(22) Filed: **Apr. 19, 2012**

Related U.S. Application Data

(60) Provisional application No. 61/479,105, filed on Apr. 26, 2011.

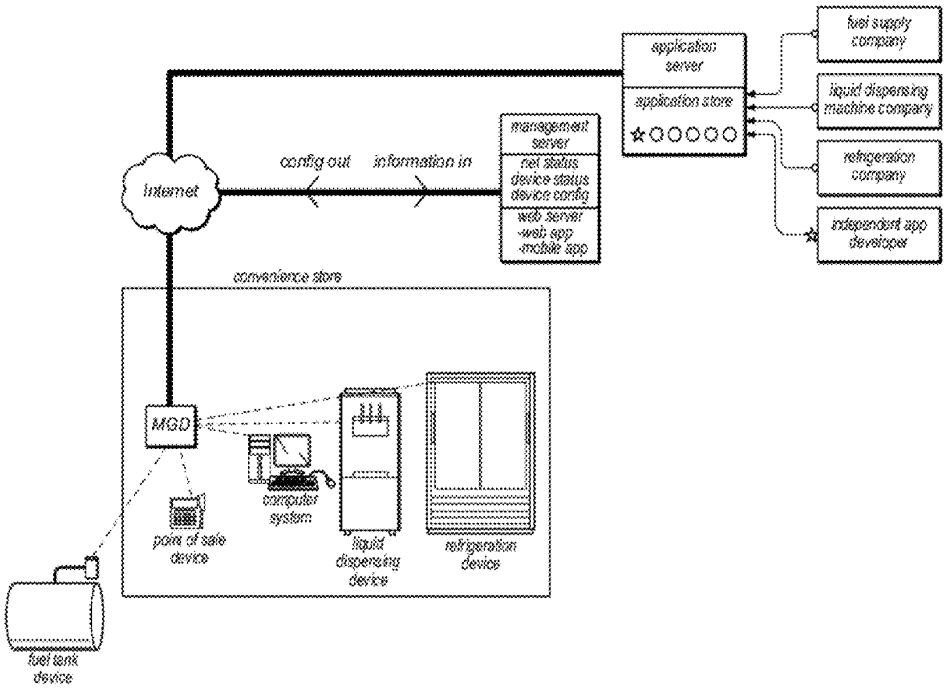
Publication Classification

(51) **Int. Cl.**
G06F 15/177 (2006.01)

(52) **U.S. Cl.** **709/220**

(57) **ABSTRACT**

Configuring and managing gateway devices. A gateway device may be installed at a location, such as a merchant location or home. The gateway device may manage (e.g., monitor and/or control) one or more devices at the location. For example, the gateway device may manage various fuel tank devices, point of sale devices, refrigeration devices, liquid dispensing devices, etc., at a convenience store. The gateway device may provide reported information to and receive commands from a cloud server, which may be in communication with other devices which may be used to view reported information of or control the managed devices. The cloud server may also be configured to install applications, e.g., for execution by the cloud server, the gateway device, and/or other devices. For example, the gateway device may install applications provided by the cloud server to manage the devices coupled to the gateway device.



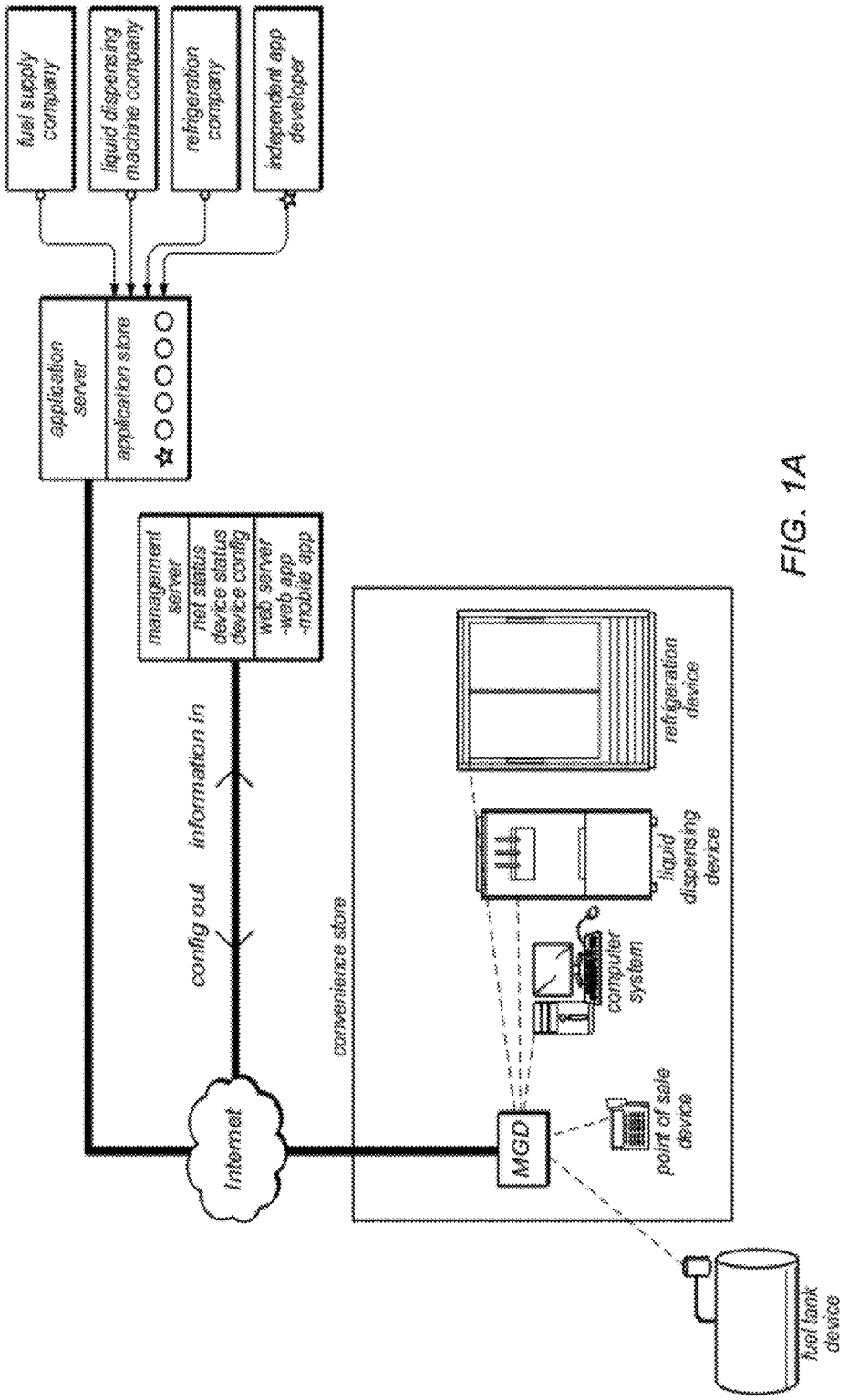

fuel supply company
liquid dispensing machine company
refrigeration company
independent app developer
application server
application store
management server
net status
device status
device config
web server
-web app
-mobile app
information in
config out
Internet
convenience store
refrigeration device
liquid dispensing device
computer system
point of sale device
MGD
fuel tank device

FIG. 1A

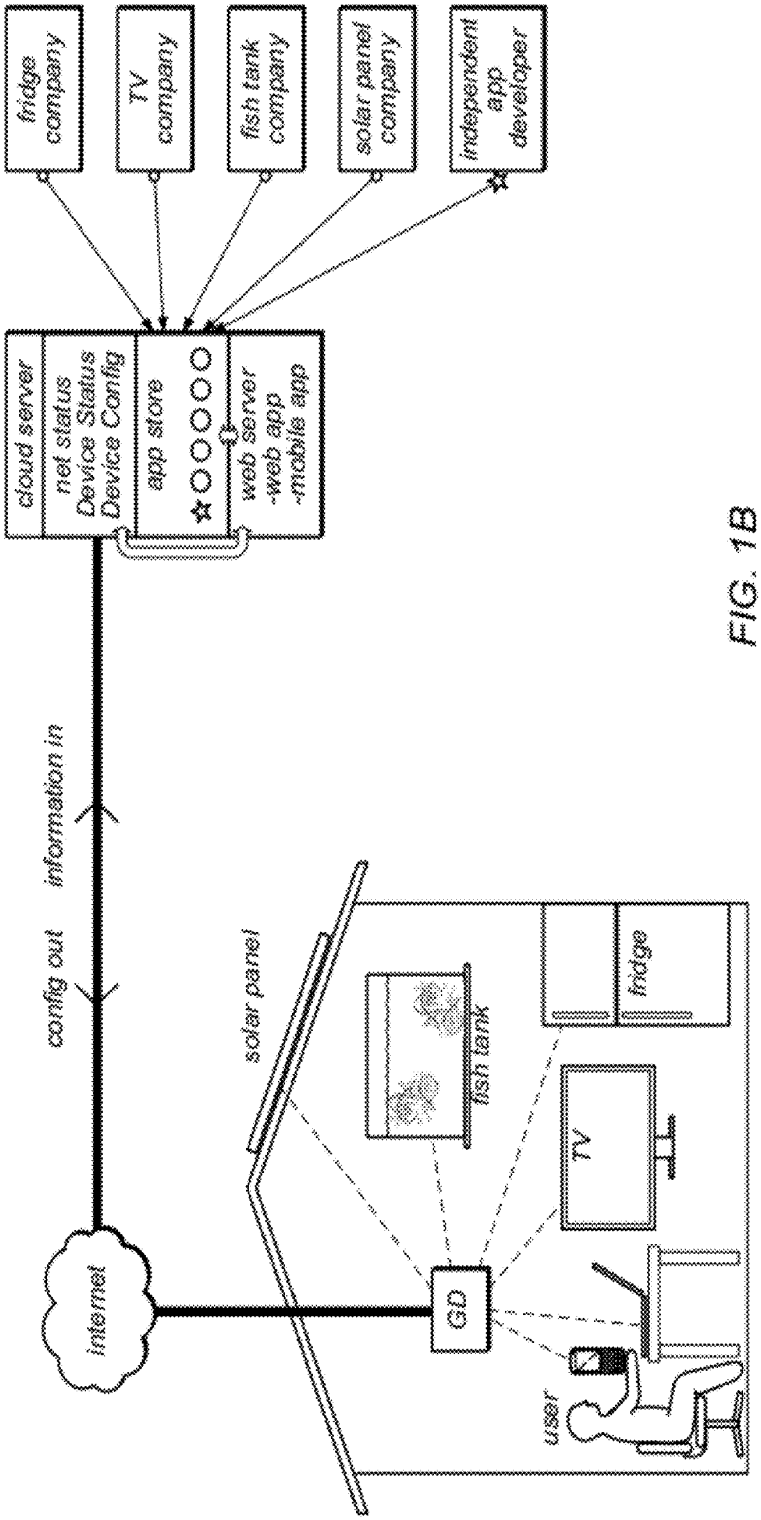

fridge company
TV company
fish tank company
solar panel company
independent app developer
cloud server
net status
Device Status
Device Config
app store
web server
-web app
-mobile app
config out
information in
internet
GD
solar panel
fish tank
TV
fridge
user

FIG. 1B

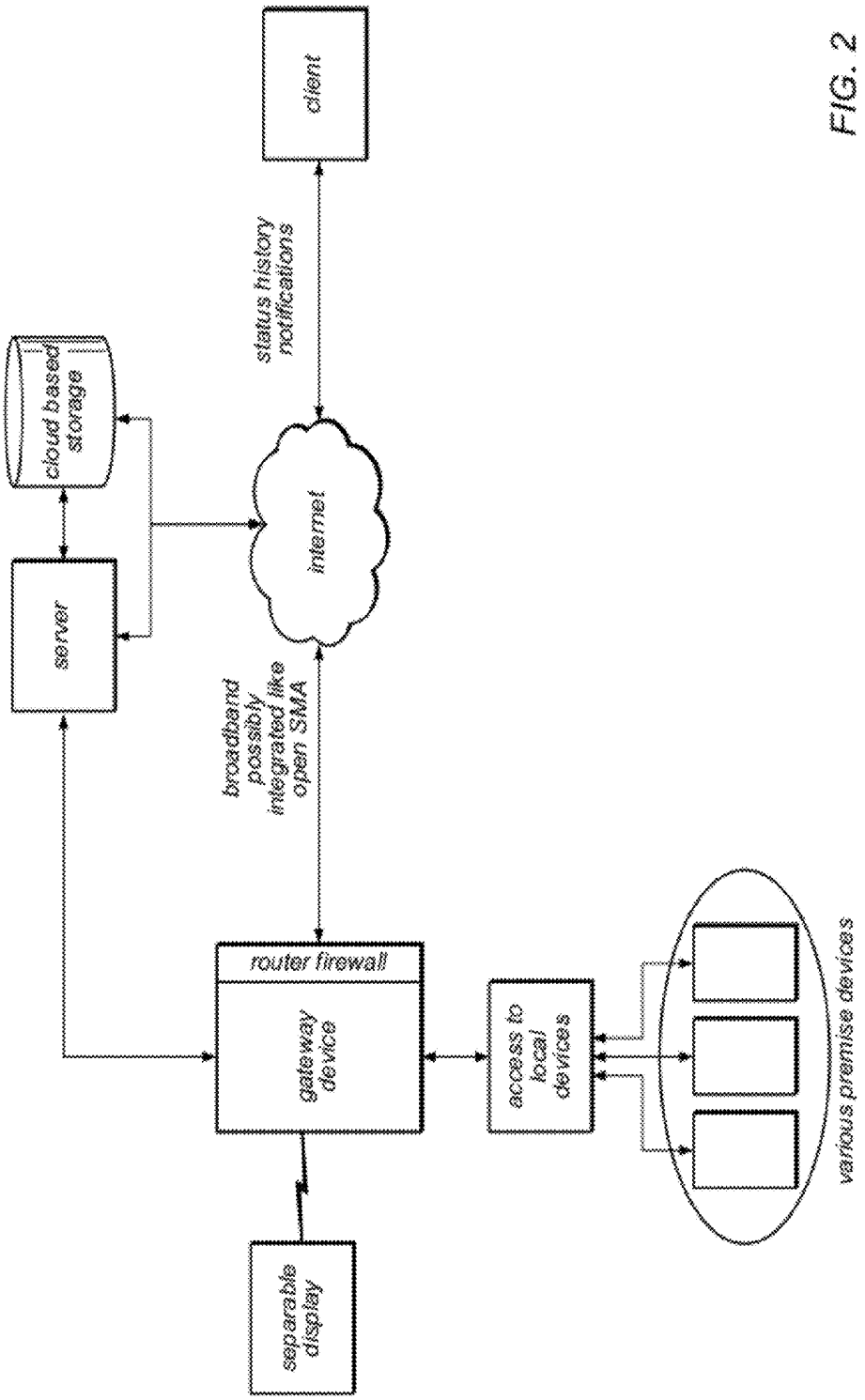

client
status history notifications
cloud based storage
internet
server
broadband possibly integrated like open SMA
router firewall
gateway device
access to local devices
various premise devices
separable display

FIG. 2

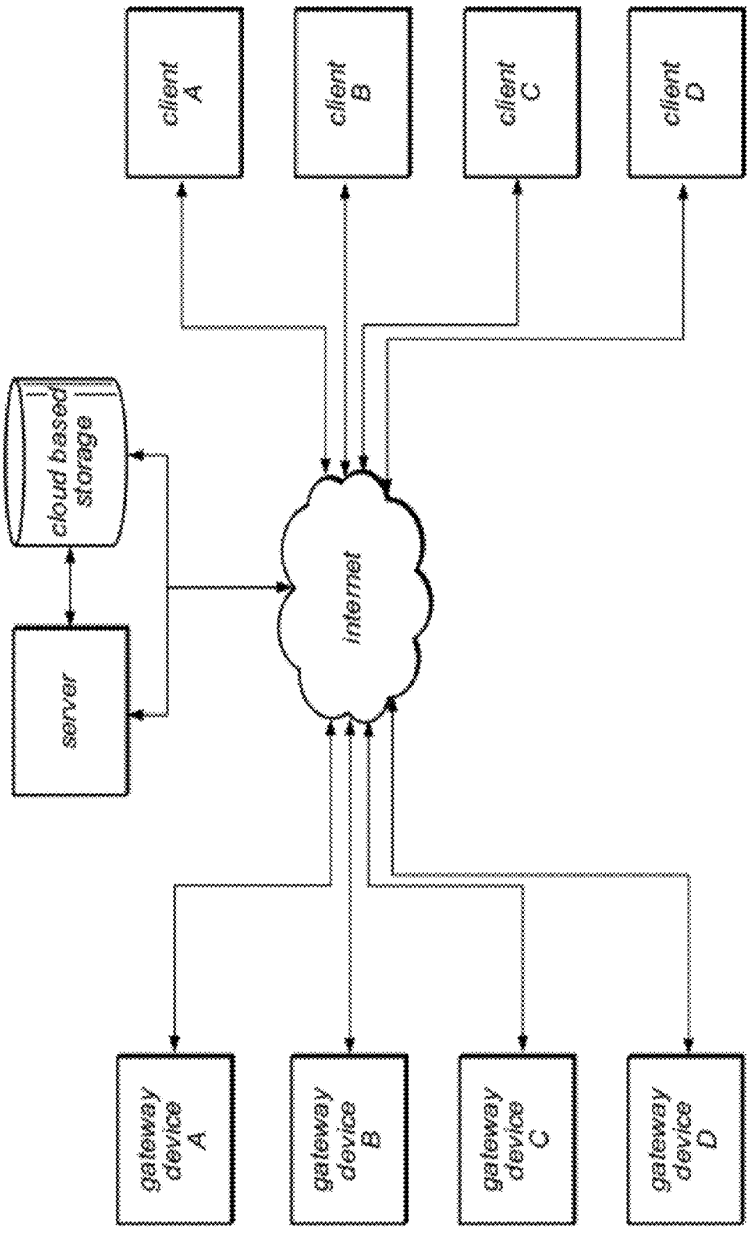

client A
client B
client C
client D
cloud based storage
server
internet
gateway device A
gateway device B
gateway device C
gateway device D

FIG. 3

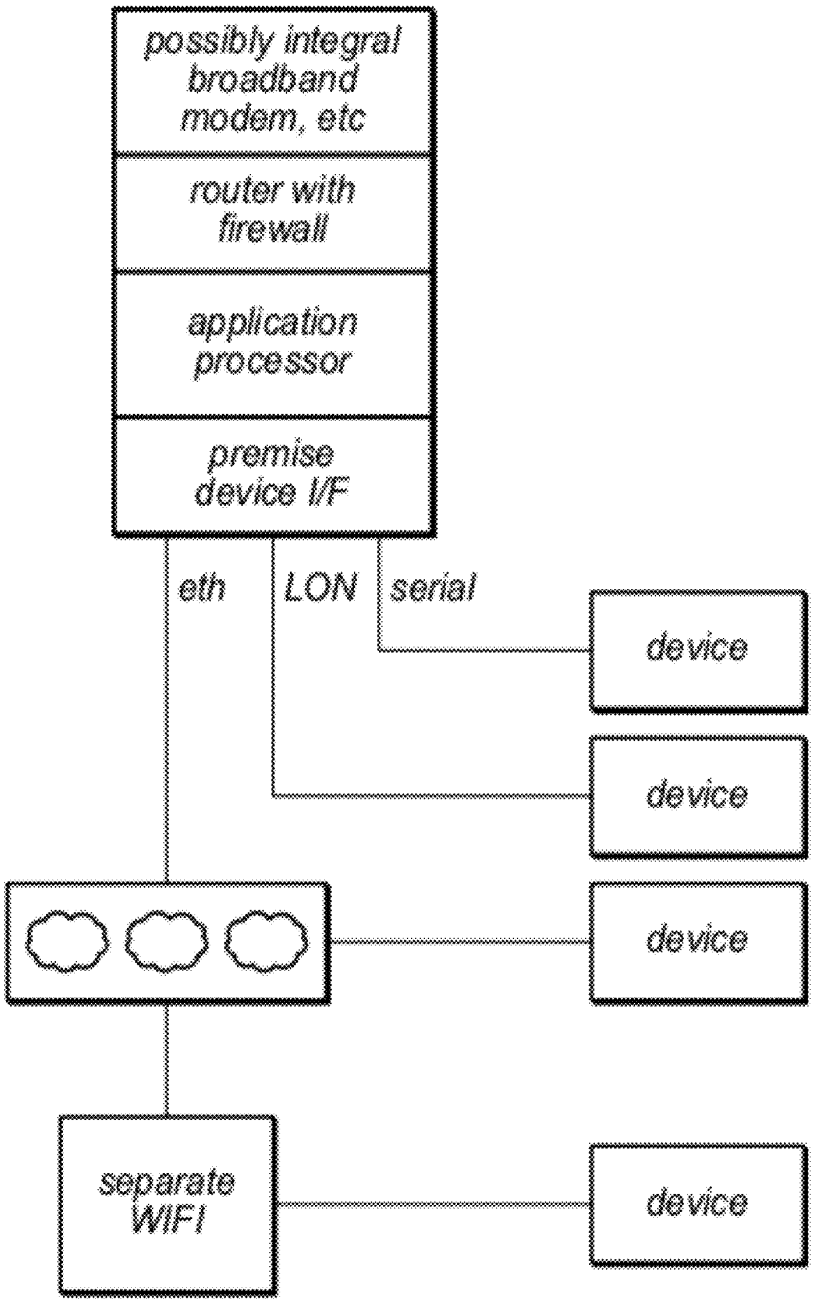

possibly integral broadband modem, etc
router with firewall
application processor
premise device I/F
eth
LON
serial
device
device
device
separate WIFI
device

FIG. 4A

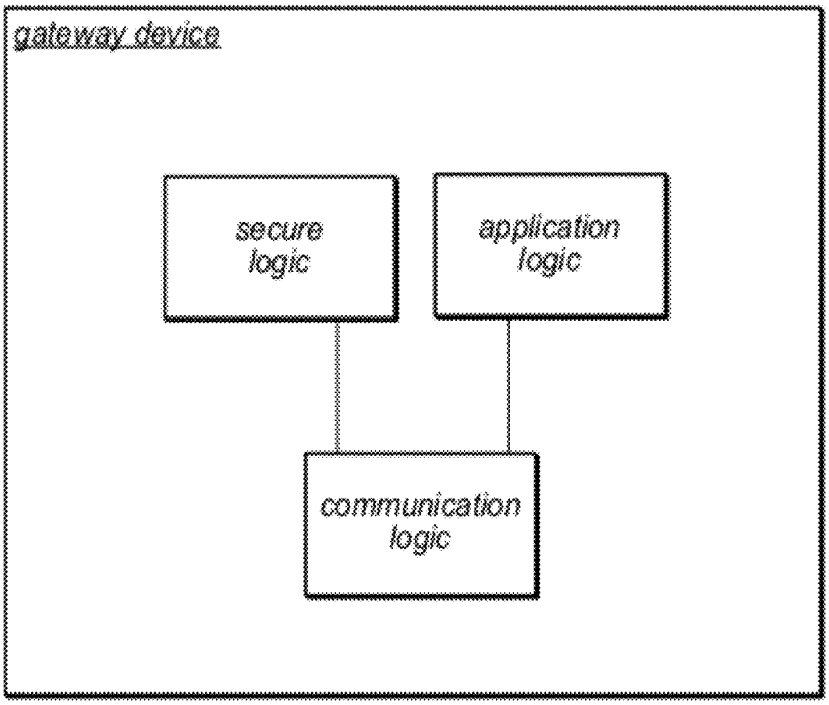

gateway device
secure logic
application logic
communication logic

FIG. 4B

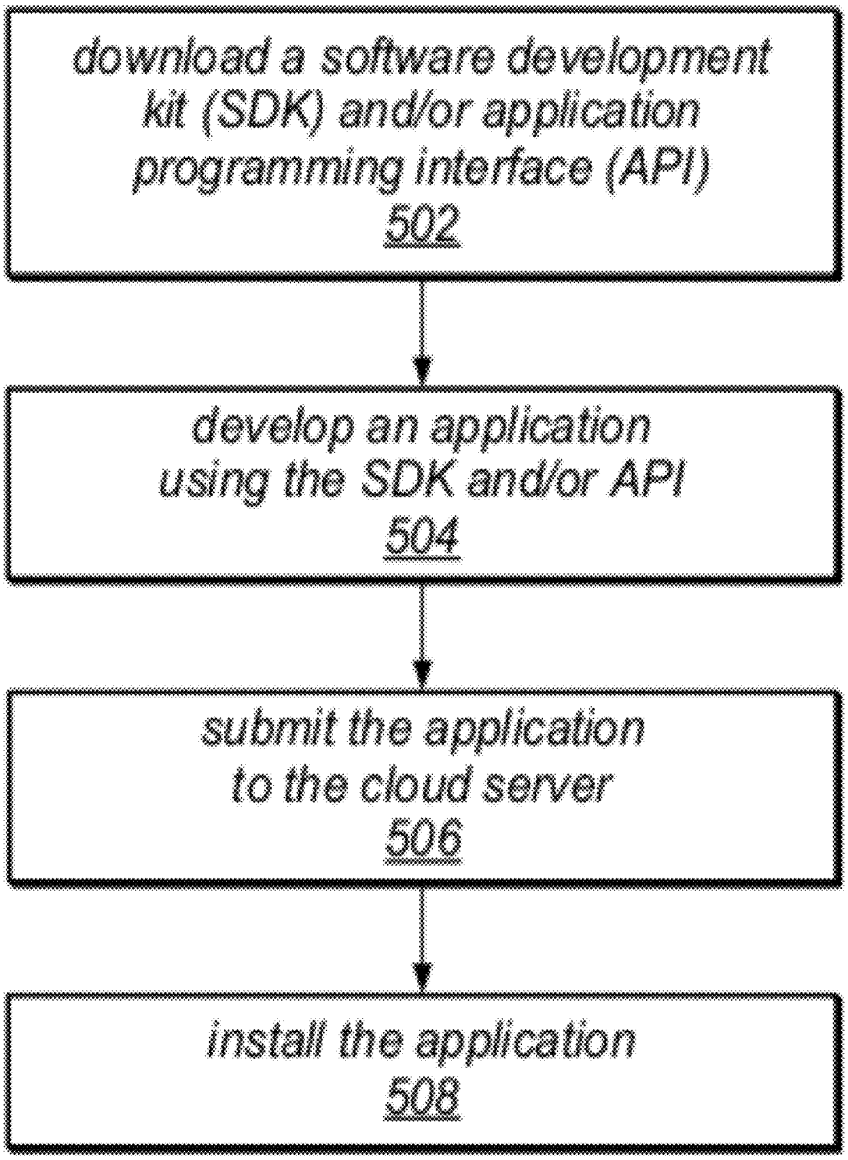

download a software development kit (SDK) and/or application programming interface (API)
502
develop an application using the SDK and/or API
504
submit the application to the cloud server
506
install the application
508

FIG. 5

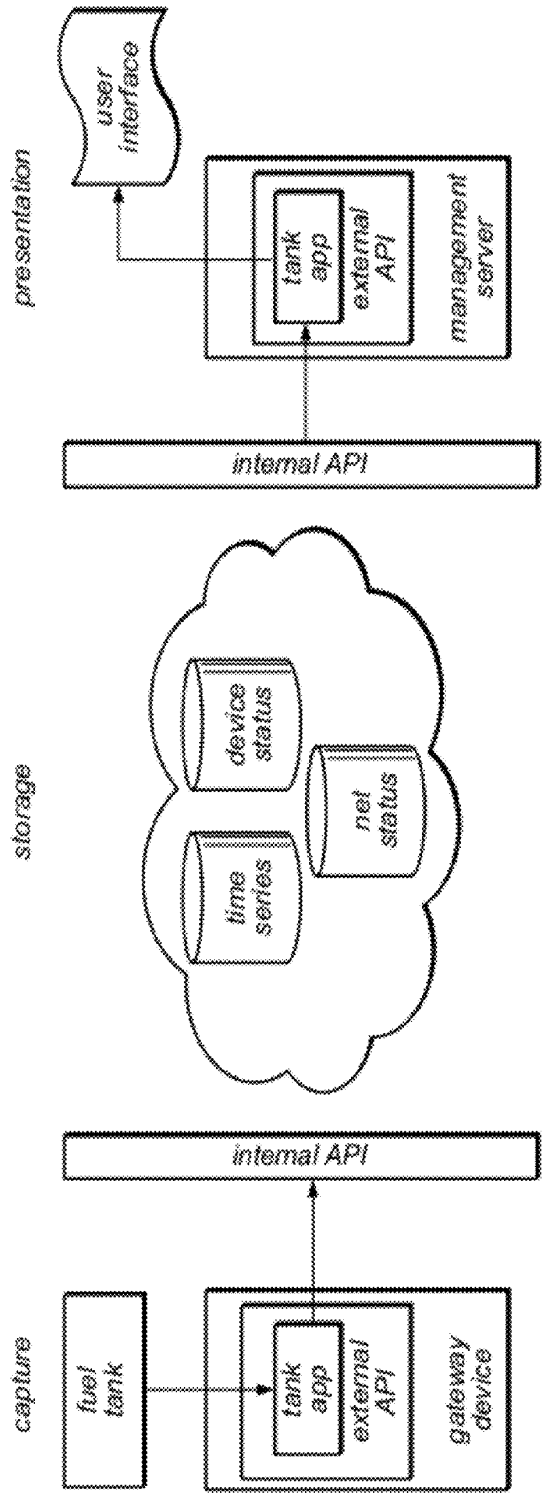

presentation
user interface
management server
tank app
external API
internal API
storage
device status
net status
time series
internal API
capture
fuel tank
tank app
external API
gateway device

FIG. 6

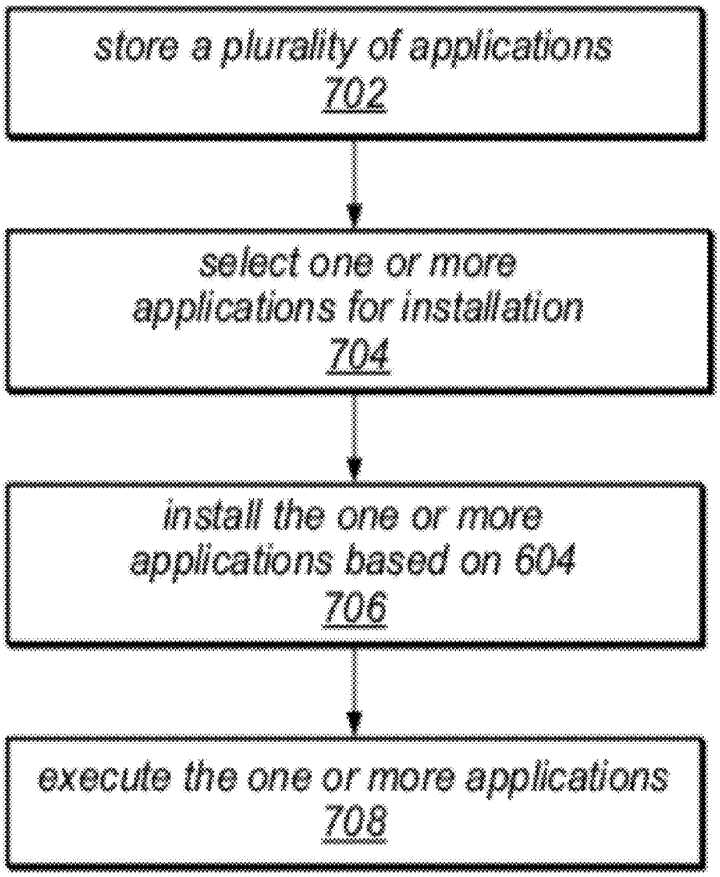

store a plurality of applications
702
select one or more applications for installation
704
install the one or more applications based on 604
706
execute the one or more applications
708

FIG. 7

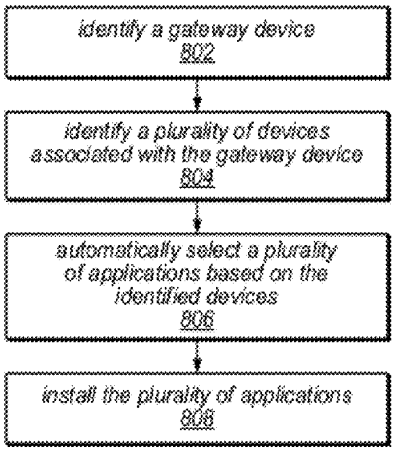


FIG. 8

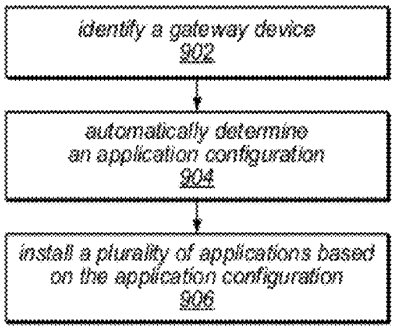


FIG. 9

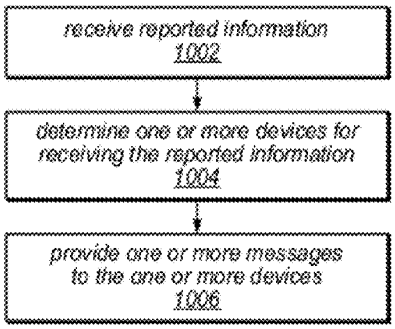


FIG. 10

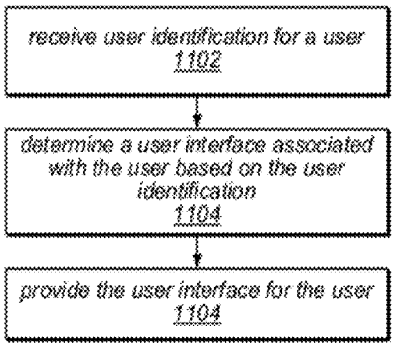


FIG. 11

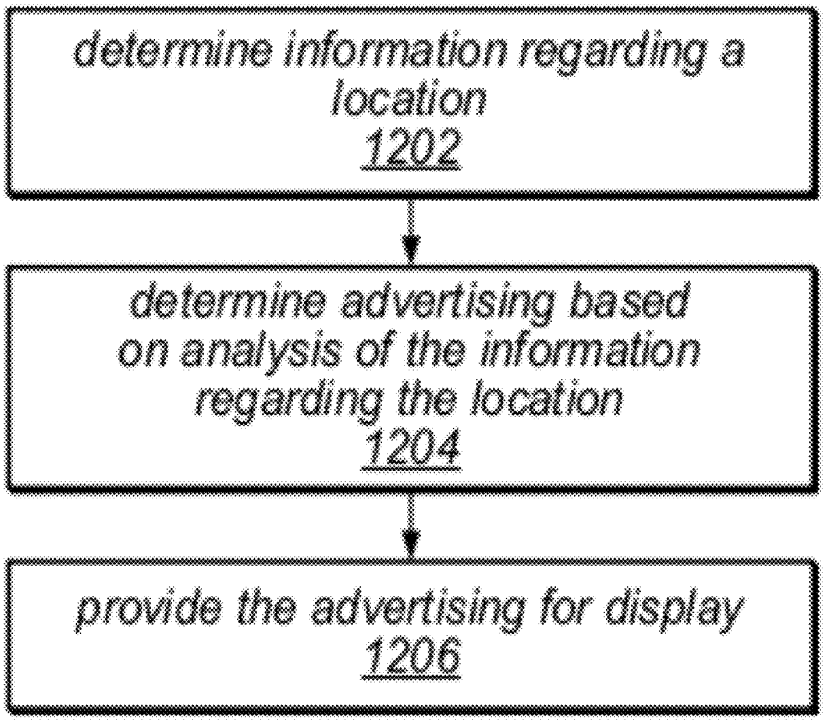

determine information regarding a location
1202
determine advertising based on analysis of the information regarding the location
1204
provide the advertising for display
1206

FIG. 12

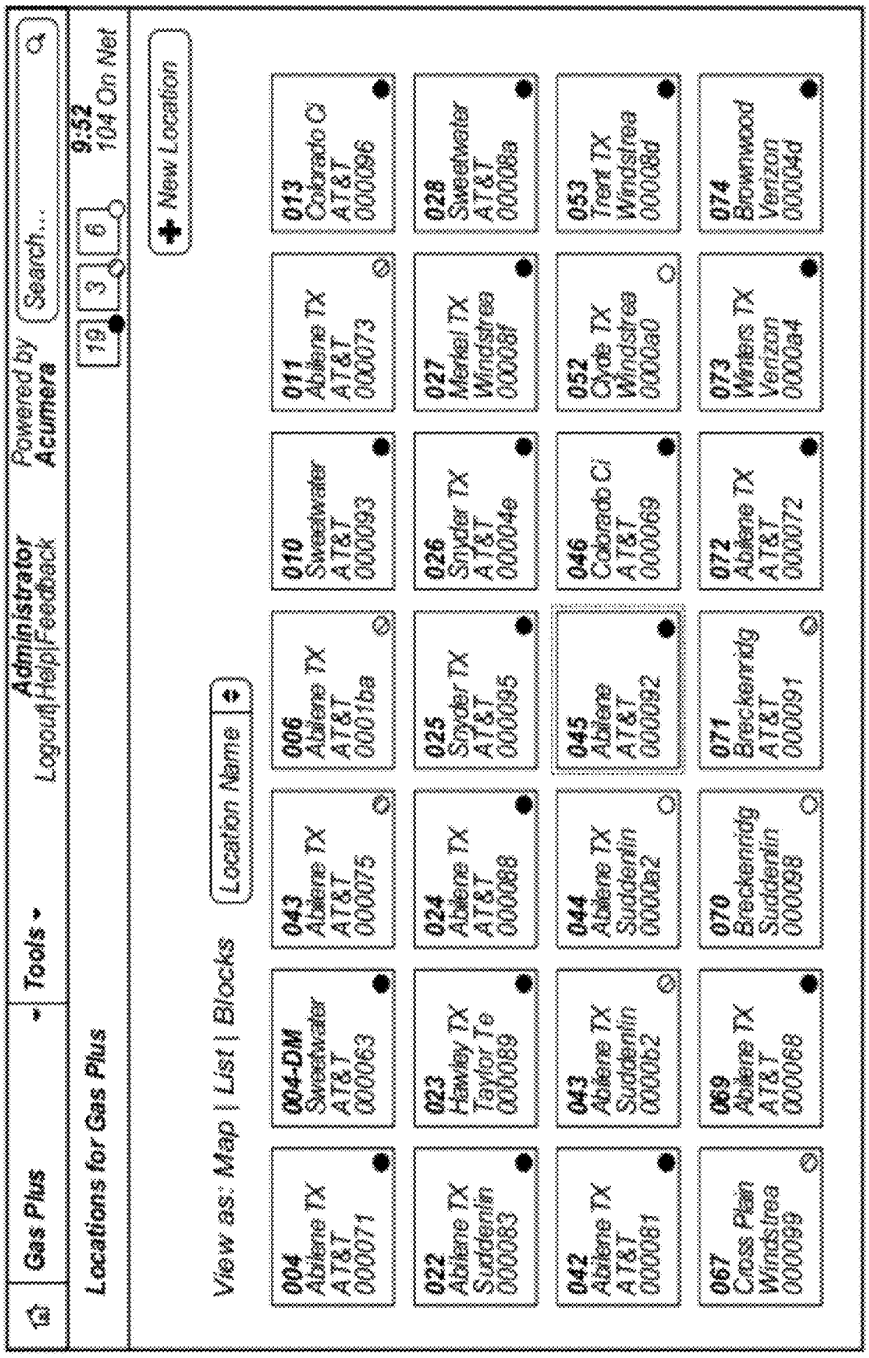

Gas Plus
Tools
Administrator
Logout|Help|Feedback
Powered by Acumera
Search...
Locations for Gas Plus
19
3
6
9:52
104 On Net
New Location
View as: Map | List | Blocks
Location Name
004 Abilene TX AT&T 000071
004-DM Sweetwater AT&T 000063
043 Abilene TX AT&T 000075
006 Abilene TX AT&T 0001ba
010 Sweetwater AT&T 000093
011 Abilene TX AT&T 000073
013 Colorado Ci AT&T 000096
022 Abilene TX Suddenlin 000083
023 Hawley TX Taylor Te 000089
024 Abilene TX AT&T 000088
025 Snyder TX AT&T 000095
026 Snyder TX AT&T 00004e
027 Merkel TX Windstrea 00008f
028 Sweetwater AT&T 0000ba
042 Abilene TX AT&T 000081
043 Abilene TX Suddenlin 0000b2
044 Abilene TX Suddenlin 0000a2
045 Abilene AT&T 000092
046 Colorado Ci AT&T 000069
052 Cyde TX Windstrea 0000a0
053 Trent TX Windstrea 00008d
067 Cross Plain Windstrea 000099
069 Abilene TX AT&T 000068
070 Breckenridg Suddenlin 000098
071 Breckenridg AT&T 000091
072 Abilene TX AT&T 000072
073 Winters TX Verizon 0000a4
074 Brownwood Verizon 00004d

FIG. 13A

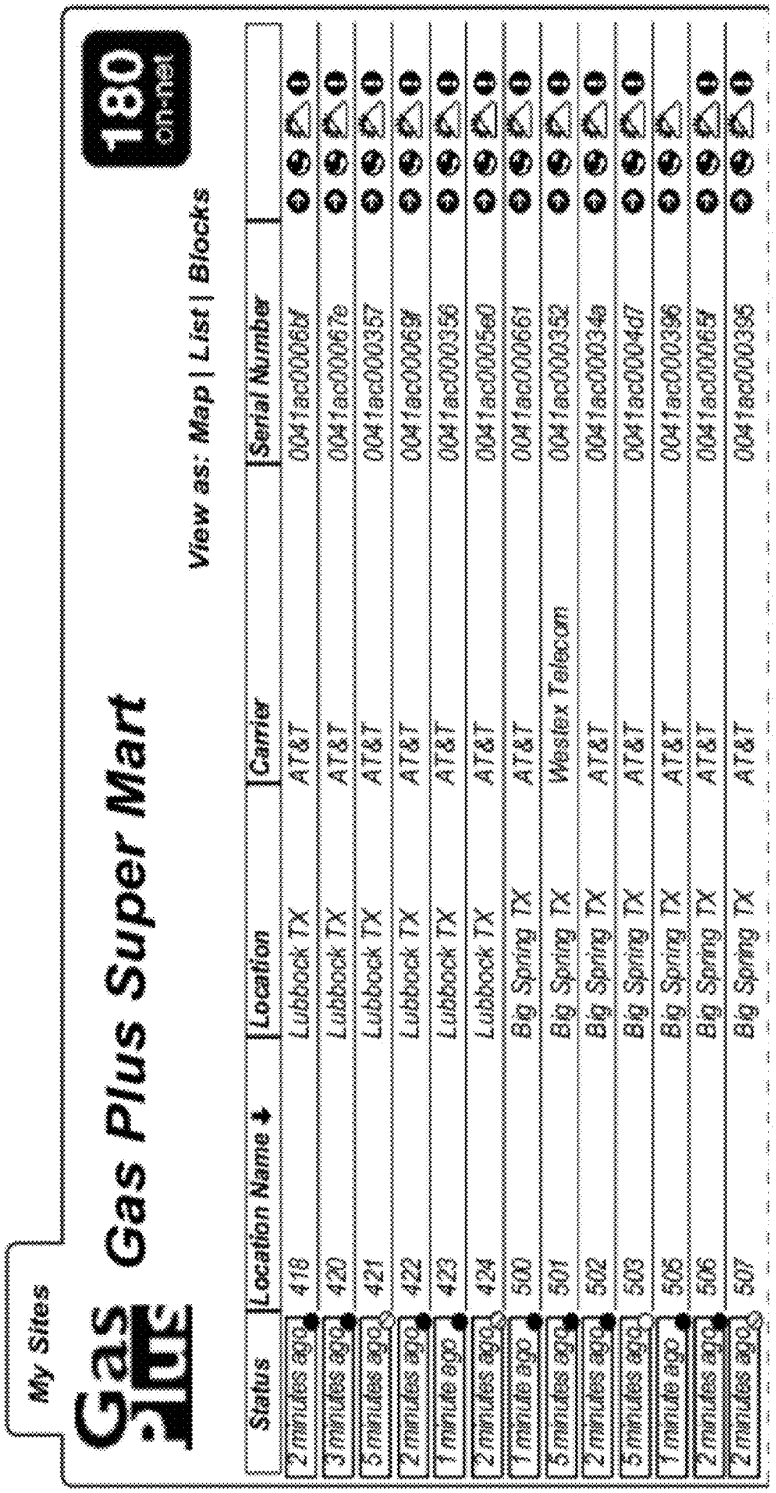

My Sites
Gas Plus Super Mart
180 online
View as: Map | List | Blocks
Status
Location Name
Location
Carrier
Serial Number
2 minutes ago 418 Lubbock TX AT&T 0041ac00061f
3 minutes ago 420 Lubbock TX AT&T 0041ac00067e
5 minutes ago 421 Lubbock TX AT&T 0041ac000357
2 minutes ago 422 Lubbock TX AT&T 0041ac00069f
1 minute ago 423 Lubbock TX AT&T 0041ac000356
2 minutes ago 424 Lubbock TX AT&T 0041ac0005e0
1 minute ago 500 Big Spring TX AT&T 0041ac000661
5 minutes ago 501 Big Spring TX Westex Telecom 0041ac000332
2 minutes ago 502 Big Spring TX AT&T 0041ac00034e
5 minutes ago 503 Big Spring TX AT&T 0041ac0004d7
1 minute ago 505 Big Spring TX AT&T 0041ac000396
2 minutes ago 506 Big Spring TX AT&T 0041ac000659
2 minutes ago 507 Big Spring TX AT&T 0041ac000395

FIG. 13B

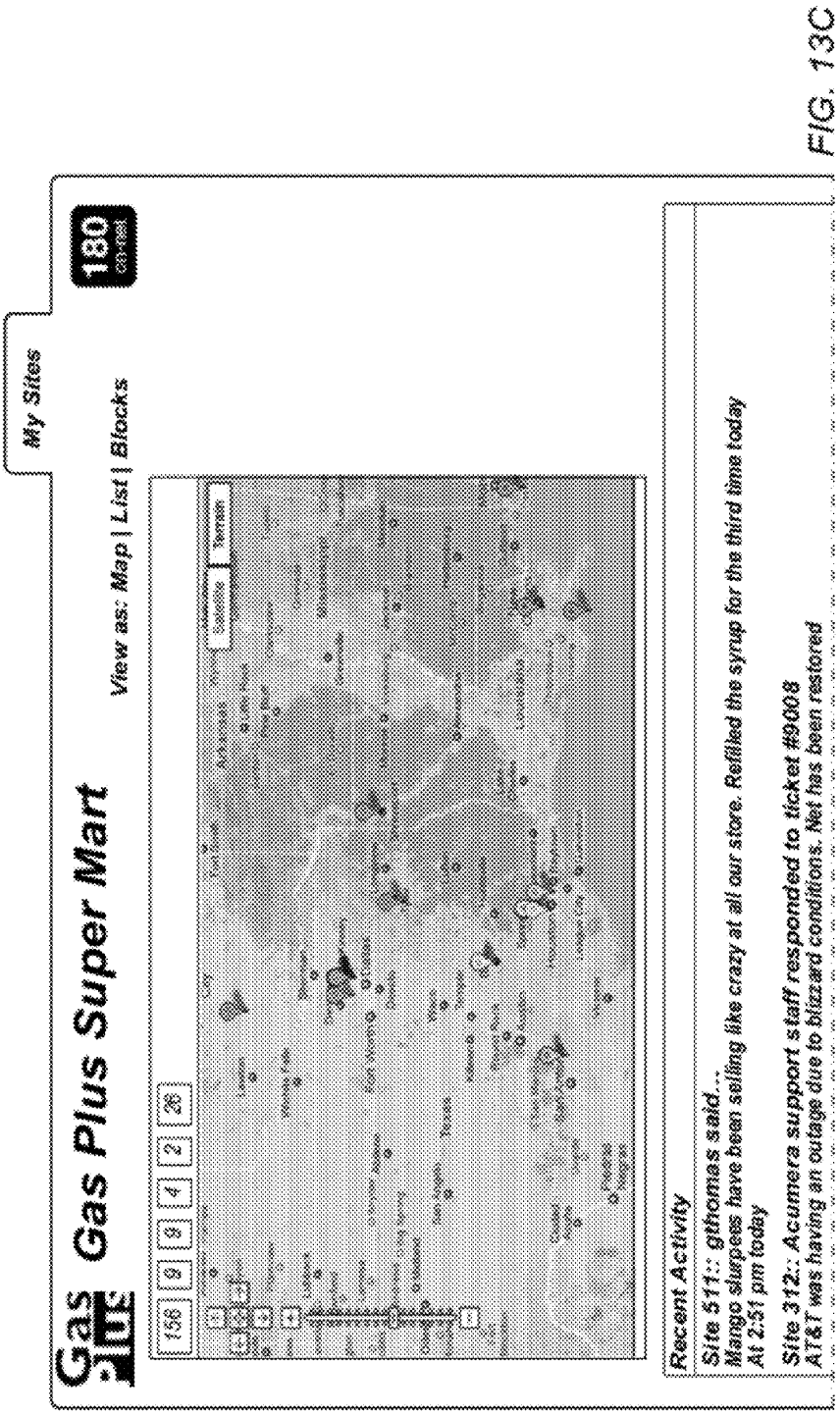

My Sites
Gas Plus Super Mart
View as: Map | List | Blocks
156
0
0
4
2
26
Recent Activity
Site 511:: gthomas said...
Mango slurpees have been selling like crazy at all our store. Refilled the syrup for the third time today
At 2:51 pm today
Site 312:: Acumera support staff responded to ticket #9008
AT&T was having an outage due to blizzard conditions. Net has been restored

FIG. 13C

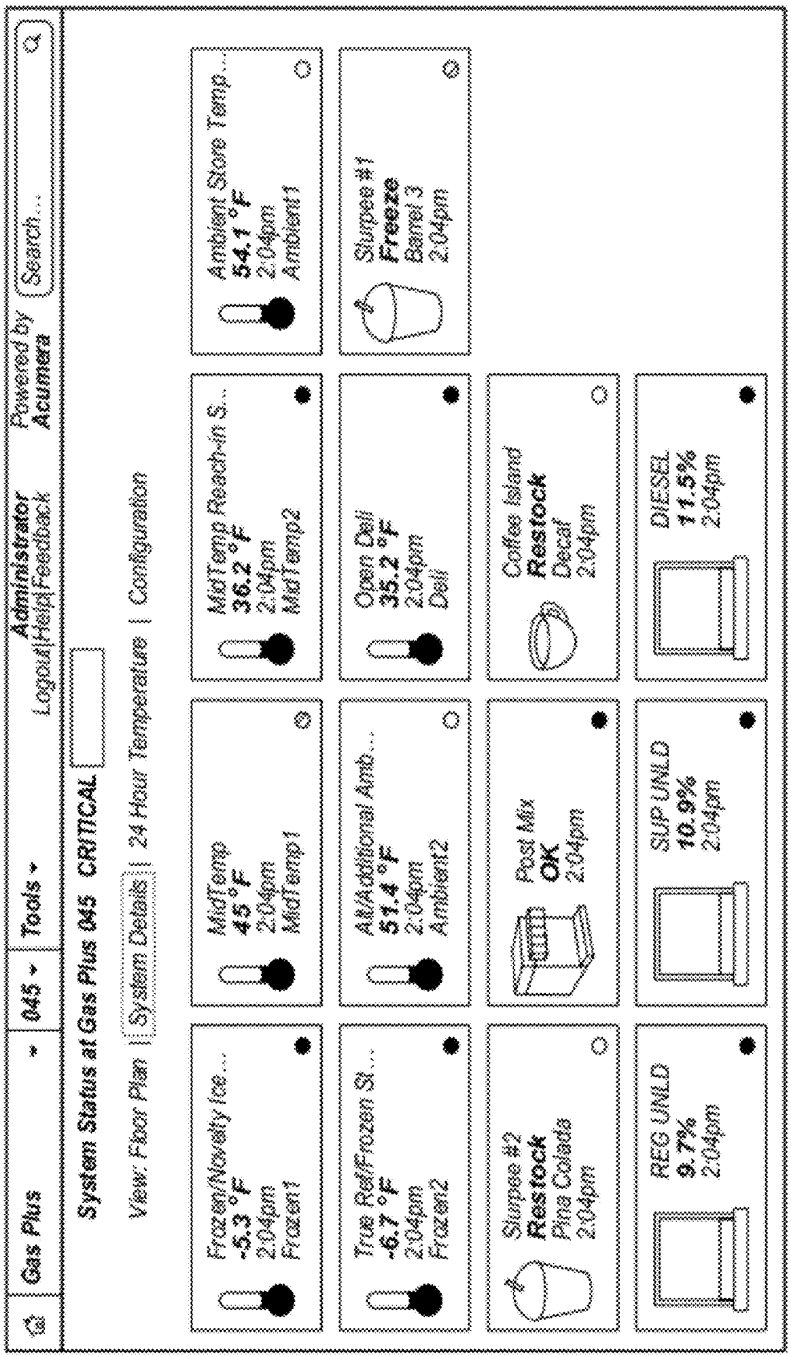
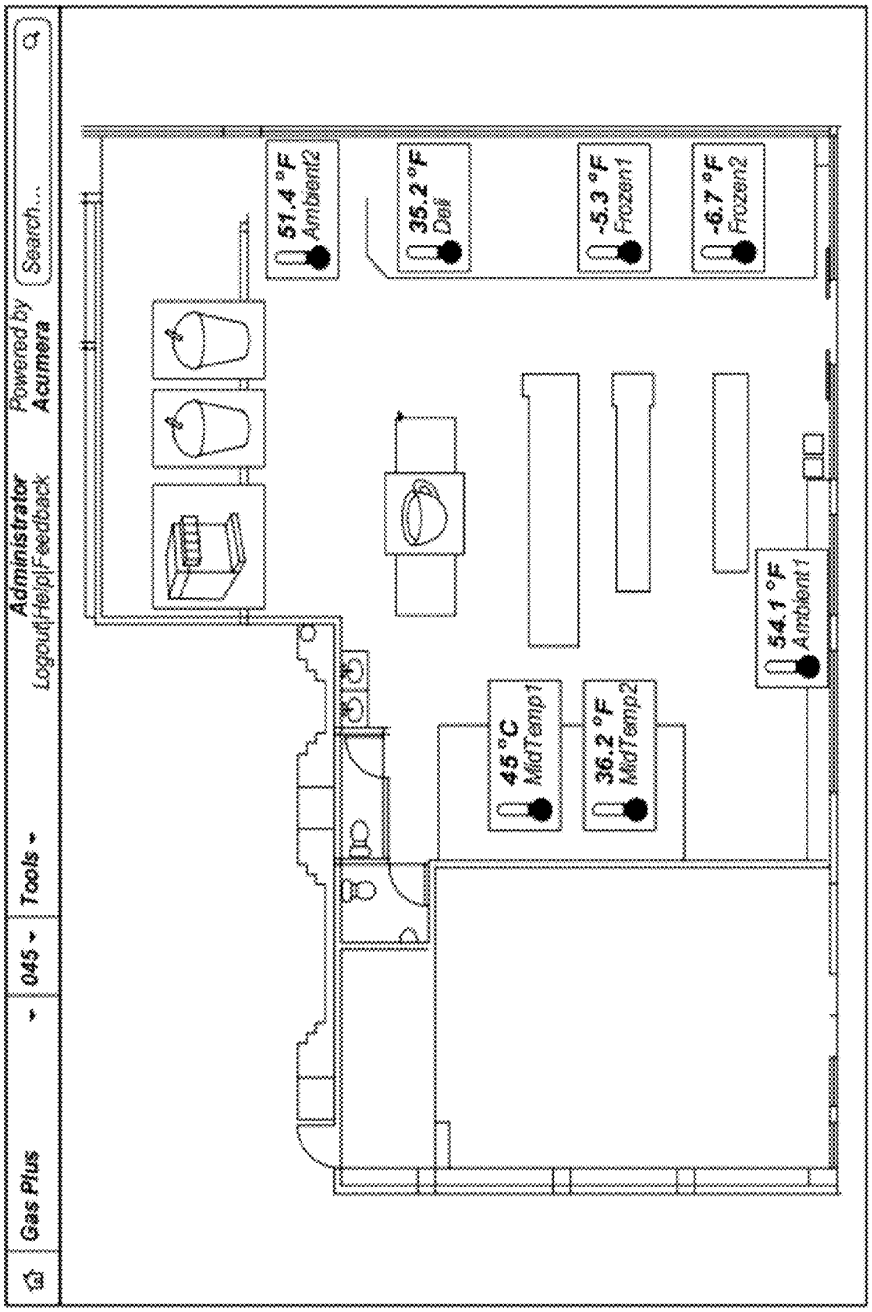

Gas Plus
045
Tools
Administrator
Logout|Help|Feedback
Powered by Acumera
Search...
System Status at Gas Plus 045 CRITICAL
View: Floor Plan | System Details | 24 Hour Temperature | Configuration
Frozen/Novelty Ice...
-5.3 °F
2:04pm
Frozen1
MidTemp
45 °F
2:04pm
MidTemp1
MidTemp Reach-in S...
36.2 °F
2:04pm
MidTemp2
Ambient Store Temp...
54.1 °F
2:04pm
Ambient1
True Ref/Frozen St...
-6.7 °F
2:04pm
Frozen2
Alt/Additional Amb...
51.4 °F
2:04pm
Ambient2
Open Deli
35.2 °F
2:04pm
Deli
Slurpee #1
Freeze
Barrel 3
2:04pm
Slurpee #2
Restock
Pina Colada
2:04pm
Post Mix
OK
2:04pm
Coffee Island
Restock
Decaf
2:04pm
REG UNLD
9.7%
2:04pm
SUP UNLD
10.9%
2:04pm
DIESEL
11.5%
2:04pm

FIG. 13D

FIG. 13E


Gas Plus
045
Tools
Administrator
Logout|Help|Feedback
Powered by Acumera
Search...
51.4 °F Ambient2
35.2 °F Deli
-5.3 °F Frozen1
-6.7 °F Frozen2
54.1 °F Ambient1
45 °C MidTemp1
36.2 °F MidTemp2

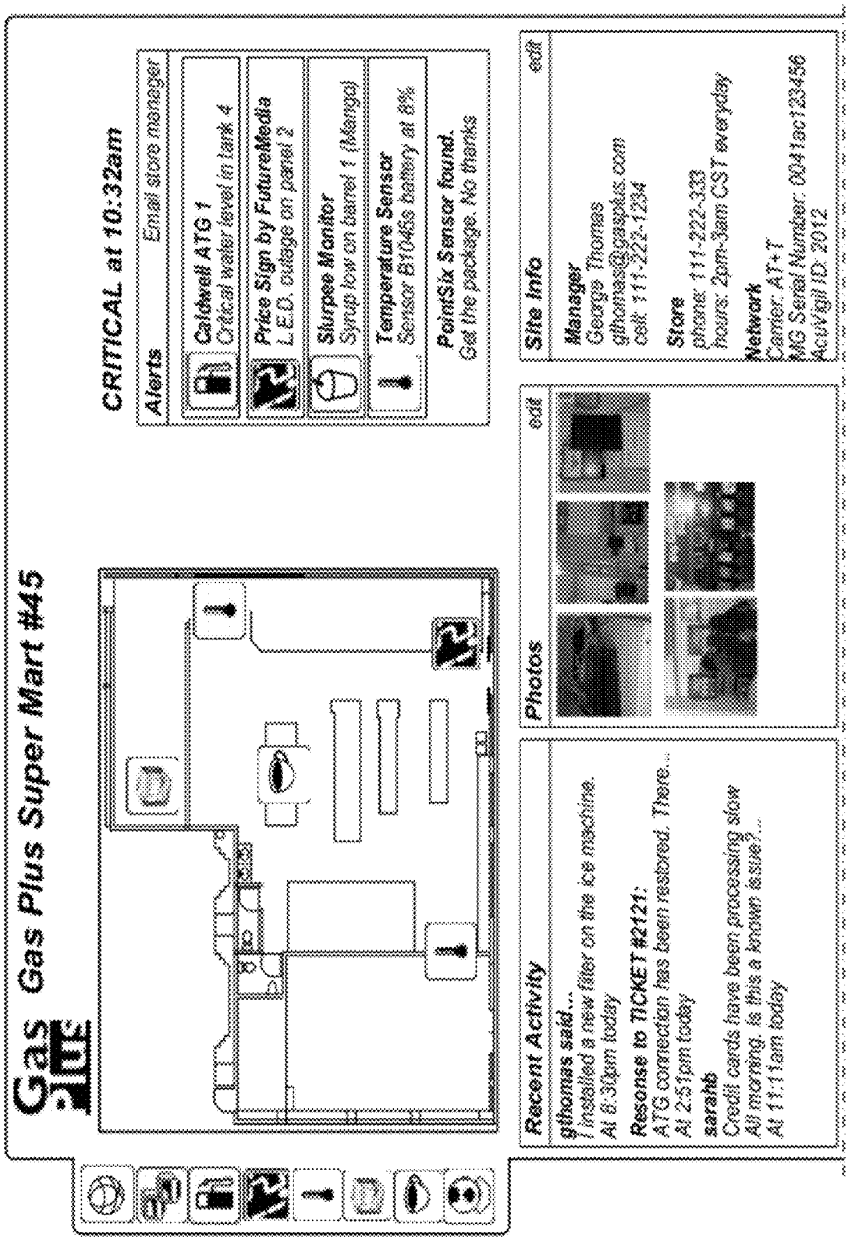

Gas Plus Super Mart #45
CRITICAL at 10:32am
Alerts
Email store manager
Caldwell ATG 1
Critical water level in tank 4
Price Sign by FutureMedia
L.E.D. outage on panel 2
Slurpee Monitor
Syrup low on barrel 1 (Mango)
Temperature Sensor
Sensor B1046s battery at 8%
PointSix Sensor found.
Get the package. No thanks
Site Info
edit
Manager
George Thomas
gthomas@gasplus.com
cell: 111-222-1234
Store
phone: 111-222-333
hours: 2pm-3am CST everyday
Network
Carrier: AT+T
MG Serial Number: 0041ac123456
Accuvigil ID: 2012
Photos
edit
Recent Activity
gthomas said...
I installed a new filter on the ice machine.
At 6:30pm today
Resonse to TICKET #2121:
ATG connection has been restored. There...
At 2:51pm today
sarahb
Credit cards have been processing slow
All morning. Is this a known issue?...
At 11:11am today

FIG. 13F

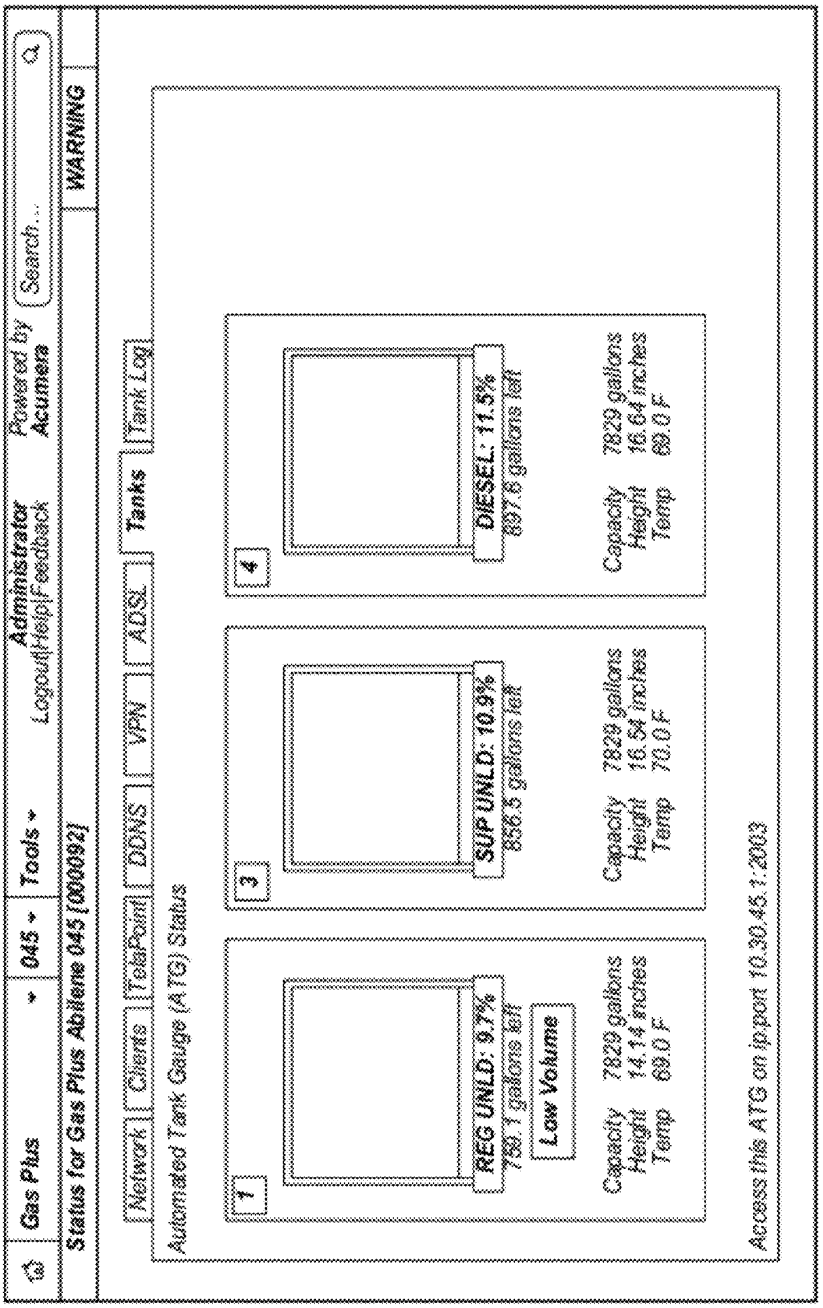

Gas Plus
045
Tools
Administrator
Logout|Help|Feedback
Powered by Acumera
Search...
Status for Gas Plus Abilene 045 (000092)
WARNING
Network
Clients
TelaPoint
DDNS
VPN
ADSL
Tanks
Tank Log
Automated Tank Gauge (ATG) Status
1
REG UNLD: 9.7%
750.1 gallons left
Low Volume
Capacity 7829 gallons
Height 14.14 inches
Temp 69.0 F
3
SUP UNLD: 10.9%
856.5 gallons left
Capacity 7829 gallons
Height 16.54 inches
Temp 70.0 F
4
DIESEL: 11.5%
897.6 gallons left
Capacity 7829 gallons
Height 16.64 inches
Temp 69.0 F
Access this ATG on ip:port 10.30.45.1:2003

FIG. 13G

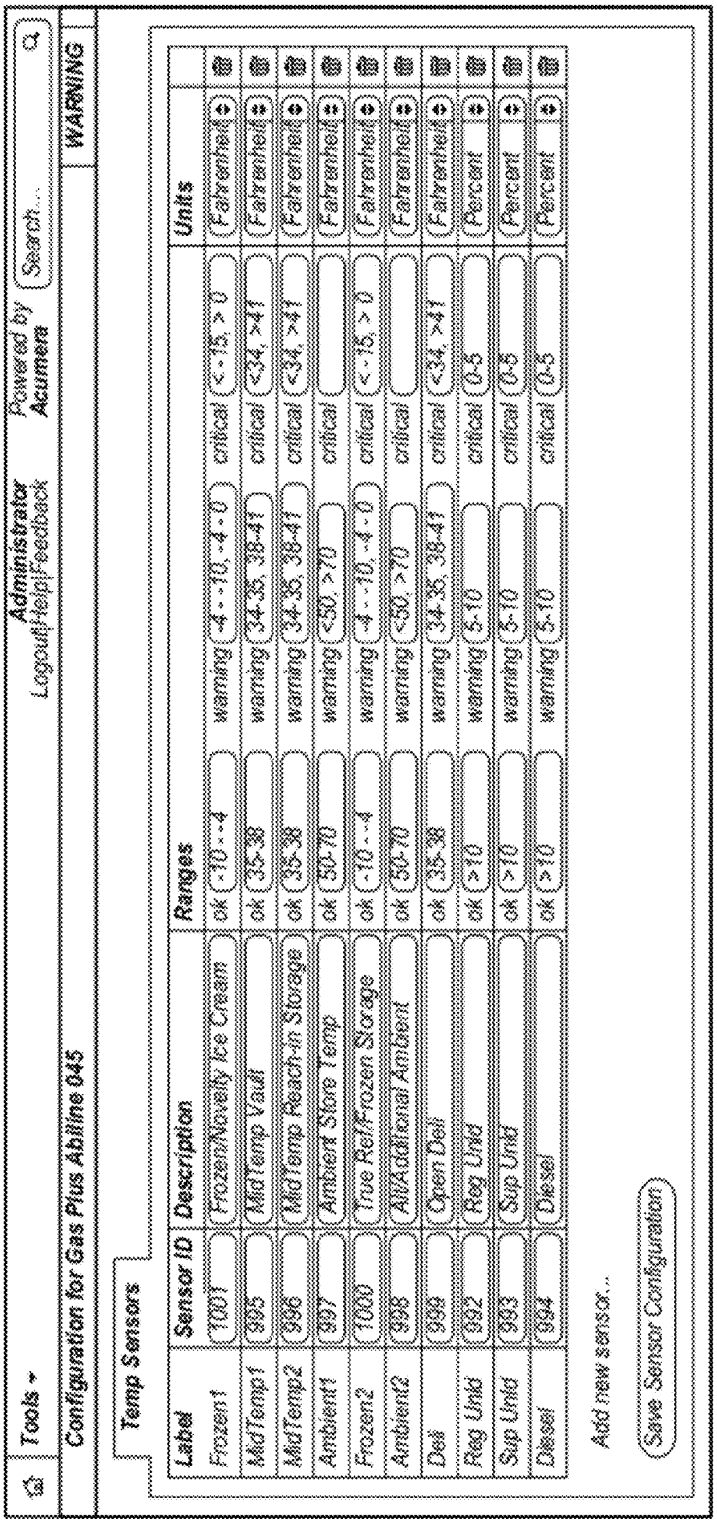

Tools
Administrator
Logout|Help|Feedback
Powered by Acumera
Search...
Configuration for Gas Plus Abilene 045
WARNING
Temp Sensors
Label
Sensor ID
Description
Ranges
Units
Frozen1 1001 Frozen/Novelty Ice Cream ok -10 - -4 warning -4 - -10, -4 - 0 critical < -15, > 0 Fahrenheit
MidTemp1 995 MidTemp Vault ok 35-38 warning 34-35, 38-41 critical <34, >41 Fahrenheit
MidTemp2 996 MidTemp Reach-in Storage ok 35-38 warning 34-35, 38-41 critical <34, >41 Fahrenheit
Ambient1 997 Ambient Store Temp ok 50-70 warning <50, >70 critical Fahrenheit
Frozen2 1000 True Ref/Frozen Storage ok -10 - -4 warning -4 - -10, -4 - 0 critical < -15, > 0 Fahrenheit
Ambient2 998 Additional Ambient ok 50-70 warning <50, >70 critical Fahrenheit
Deli 999 Open Deli ok 35-38 warning 34-35, 38-41 critical <34, >41 Fahrenheit
Reg Unit 992 Reg Unit ok >10 warning 5-10 critical 0-5 Percent
Sup Unit 993 Sup Unit ok >10 warning 5-10 critical 0-5 Percent
Diesel 994 Diesel ok >10 warning 5-10 critical 0-5 Percent
Add new sensor...
Save Sensor Configuration

FIG. 13H

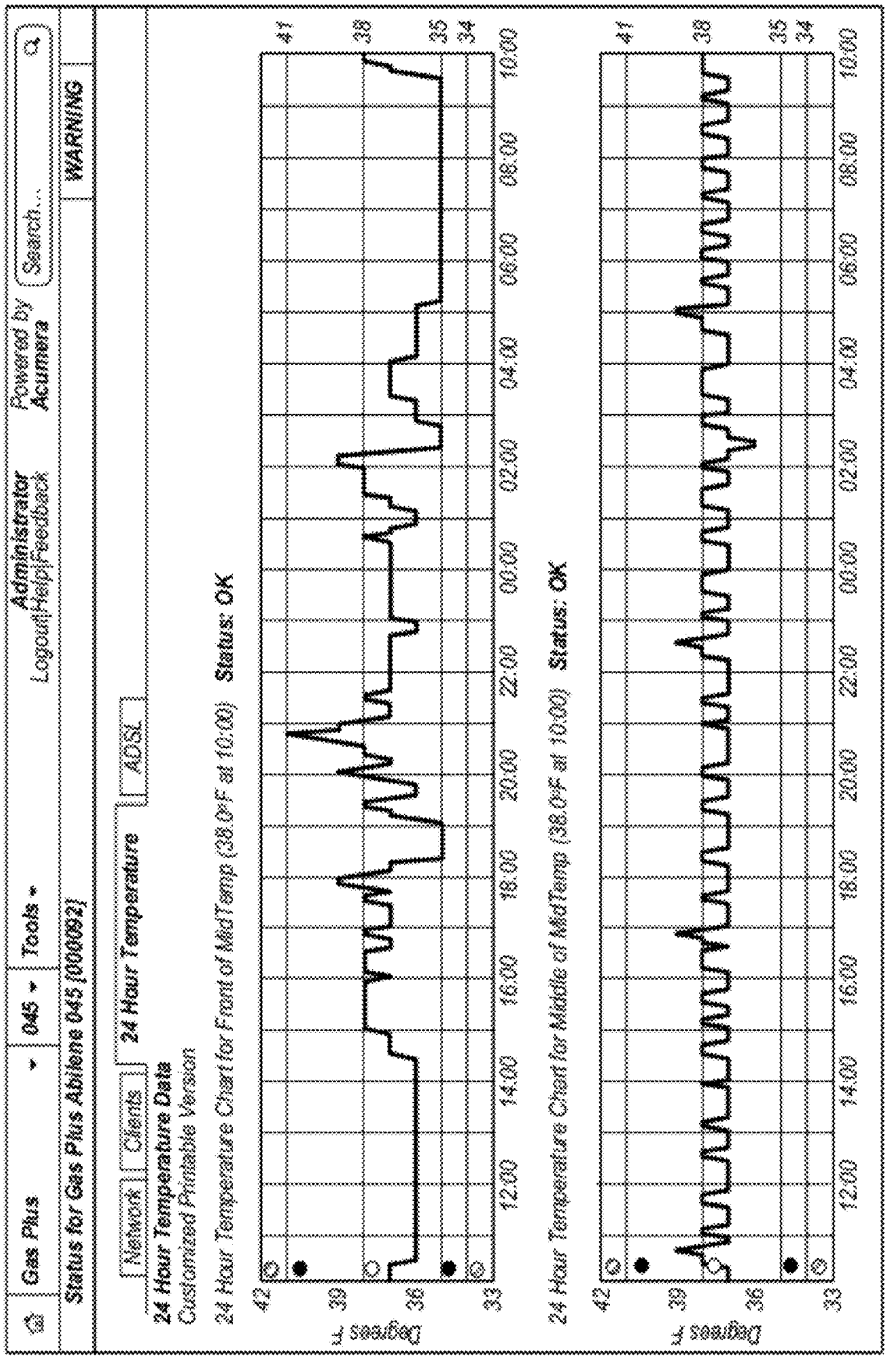
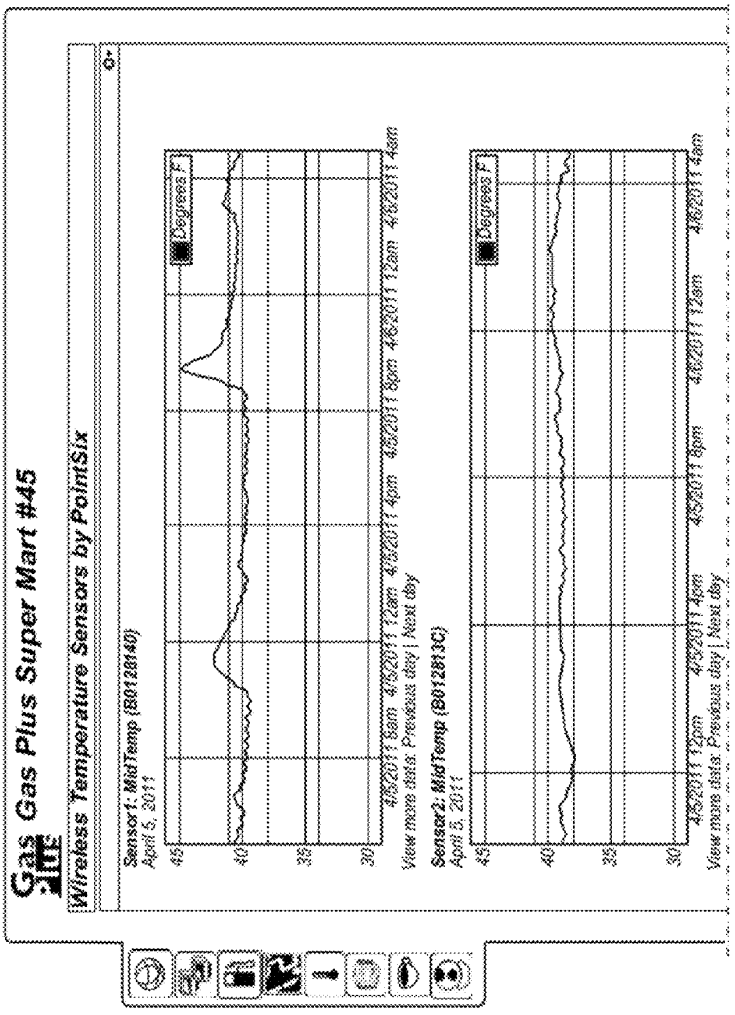

Gas Plus
045
Tools
Logout|Help|Feedback
Administrator
Powered by Acumera
Search...
Status for Gas Plus Abilene 045 [000092]
WARNING
Network
Clients
24 Hour Temperature
ADSL
24 Hour Temperature Data
Customized Printable Version
24 Hour Temperature Chart for Front of MidTemp (38.0°F at 10:00) Status: OK
24 Hour Temperature Chart for Middle of MidTemp (38.0°F at 10:00) Status: OK
Degrees F
42
39
36
33
41
38
35
34
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
08:00
10:00

FIG. 13I

FIG. 13J


Gas Plus Super Mart #45
Wireless Temperature Sensors by PointSix
Sensor1: MidTemp (B012B140)
April 5, 2011
Degrees F
45
40
35
30
4/5/2011 8am
4/5/2011 12am
4/5/2011 4pm
4/5/2011 8pm
4/6/2011 12am
4/6/2011 4am
View more data: Previous day | Next day
Sensor2: MidTemp (B012B13C)
April 5, 2011
Degrees F
45
40
35
30
4/5/2011 12pm
4/5/2011 4pm
4/5/2011 8pm
4/6/2011 12am
4/6/2011 4am
View more data: Previous day | Next day

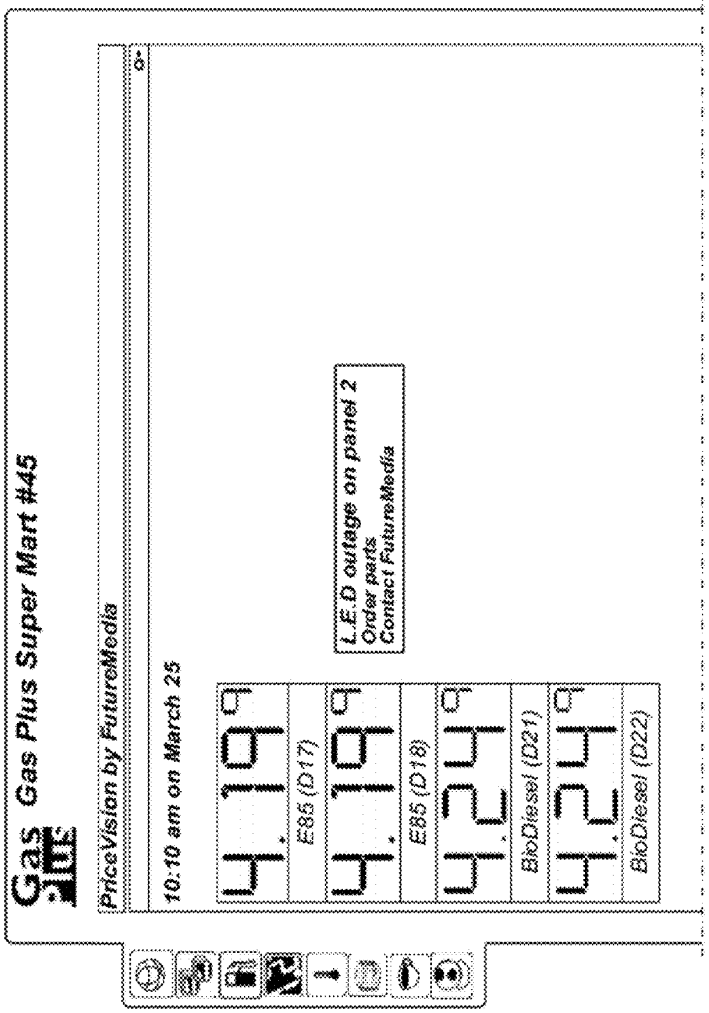
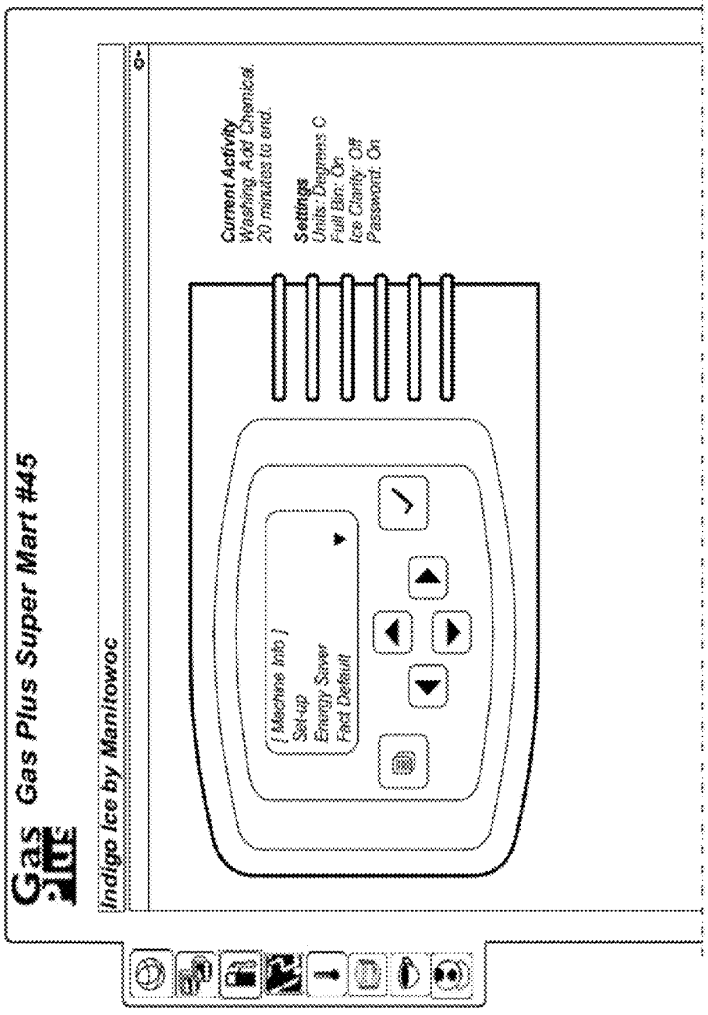

Gas Plus Super Mart #45
PriceVision by FutureMedia
10:10 am on March 25
4.10
E85 (D17)
4.10
E85 (D18)
4.24
BioDiesel (D21)
4.24
BioDiesel (D22)
L.E.D outage on panel 2
Order parts
Contact FutureMedia

FIG. 13K

FIG. 13L


Gas Plus Super Mart #45
Indigo Ice by Manitowoc
Current Activity
Washing. Add Chemical.
20 minutes to end.
Settings
Units: Degrees C
Full Bin: On
Ice Clarity: Off
Password: On
[Machine Info]
Set-up
Energy Saver
Fact Default

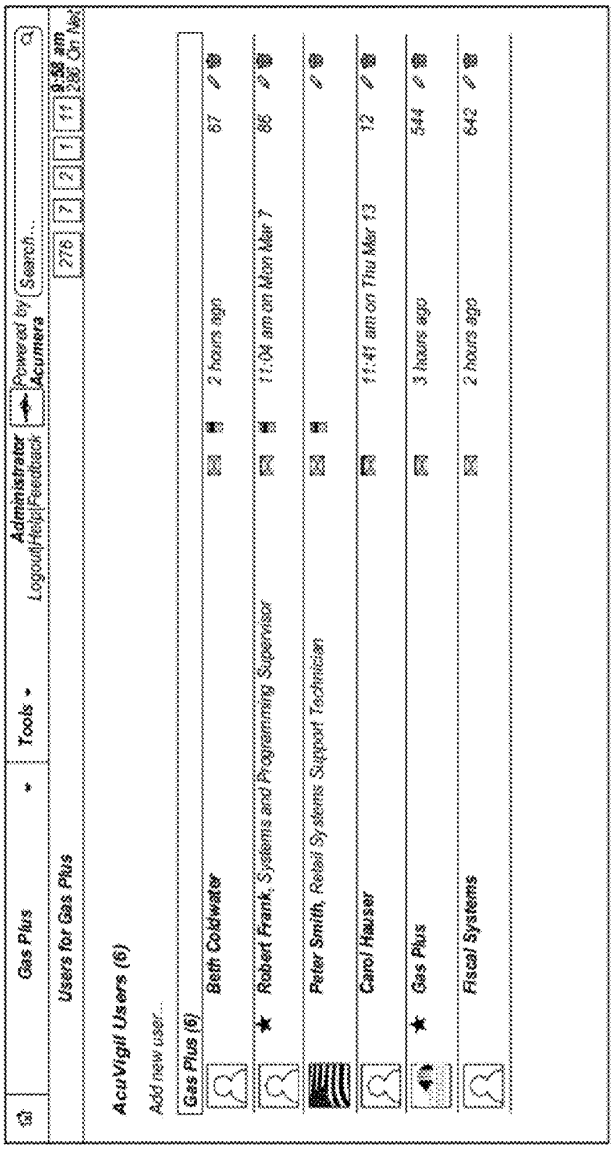

Gas Plus
Tools
Administrator
Logout|Help|Feedback
Powered by Acumera
Search...
Users for Gas Plus
276 7 2 1 11 8:58 am 28c Sri Nov
AcuVigil Users (6)
Add new user...
Gas Plus (6)
Beth Coldwater
2 hours ago
67
Robert Frank, Systems and Programming Supervisor
11:04 am on Mon Mar 7
86
Peter Smith, Retail Systems Support Technician
11:41 am on Thu Mar 13
12
Gas Plus
3 hours ago
544
Fiscal Systems
2 hours ago
642

FIG. 14A

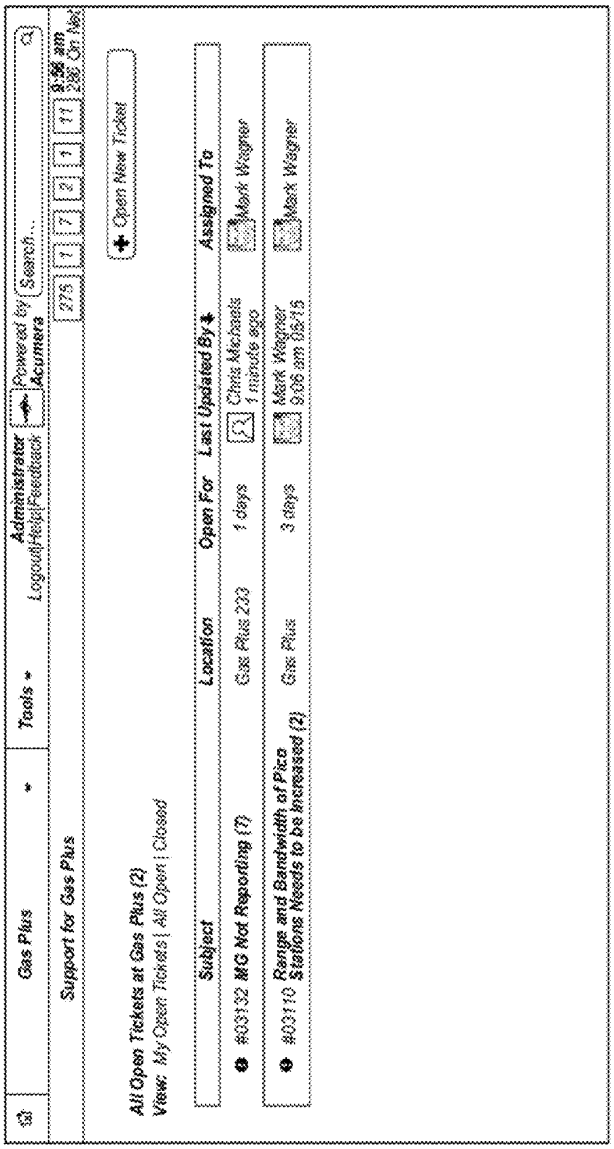

Gas Plus
Tools
Administrator
Logout|Help|Feedback
Powered by Acumera
Search...
Support for Gas Plus
275
7
7
2
1
11
8:56 am
286 On Net
Open New Ticket
All Open Tickets at Gas Plus (2)
View: My Open Tickets | All Open | Closed
Subject
Location
Open For
Last Updated By
Assigned To
#03132 MG Not Reporting (7)
Gas Plus 233
1 days
Chris Michaels
1 minute ago
Mark Wagner
#03110 Range and Bandwidth of Pico Stations Needs to be Increased (2)
Gas Plus
3 days
Mark Wagner
9:06 am 05/15
Mark Wagner

FIG. 14B

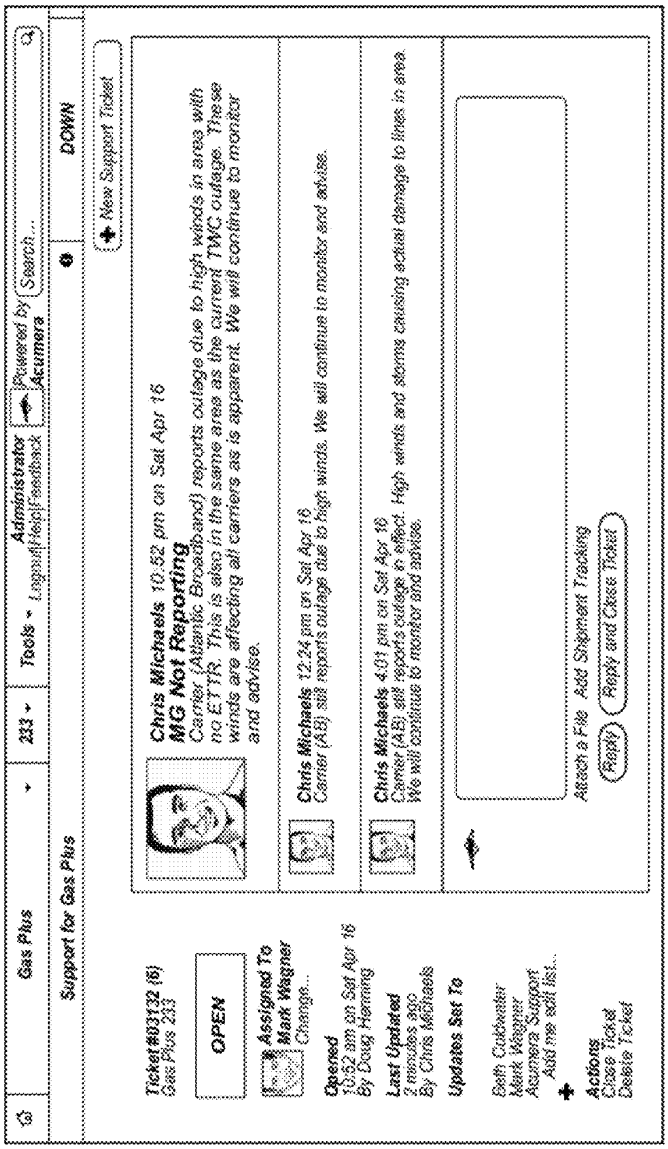

Gas Plus
233
Tools
Logout
Help
Feedback
Administrator
Powered by Acumera
Search...
Support for Gas Plus
DOWN
Ticket #03132 (6)
Gas Plus 233
OPEN
Assigned To
Mark Wagner
Change...
Opened
10:52 am on Sat Apr 16
By Doug Herring
Last Updated
2 minutes ago
By Chris Michaels
Updates Set To
Beth Callwater
Mark Wagner
Acumera Support
Add me still list...
Actions
Close Ticket
Delete Ticket
New Support Ticket
Chris Michaels 10:52 pm on Sat Apr 16
MG Not Reporting
Carrier (Atlantic Broadband) reports outage due to high winds in area with no ETTR. This is also in the same area as the current TWC outage. These winds are affecting all carriers as is apparent. We will continue to monitor and advise.
Chris Michaels 12:24 pm on Sat Apr 16
Carrier (AB) still reports outage due to high winds. We will continue to monitor and advise.
Chris Michaels 4:01 pm on Sat Apr 16
Carrier (AB) still reports outage in effect. High winds and storms causing actual damage to lines in area. We will continue to monitor and advise.
Attach a File
Add Shipment Tracking
Reply
Reply and Close Ticket

FIG. 14C

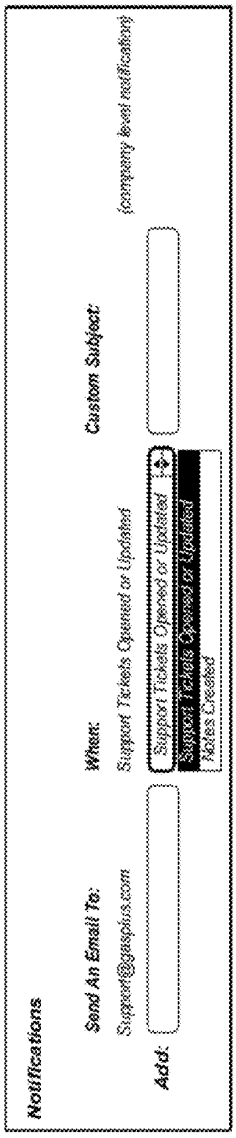

Notifications
Send An Email To:
Support@gasplus.com
Add:
When:
Support Tickets Opened or Updated
Support Tickets Opened or Updated
Notes Created
Custom Subject:
(company level notification)

FIG. 14D

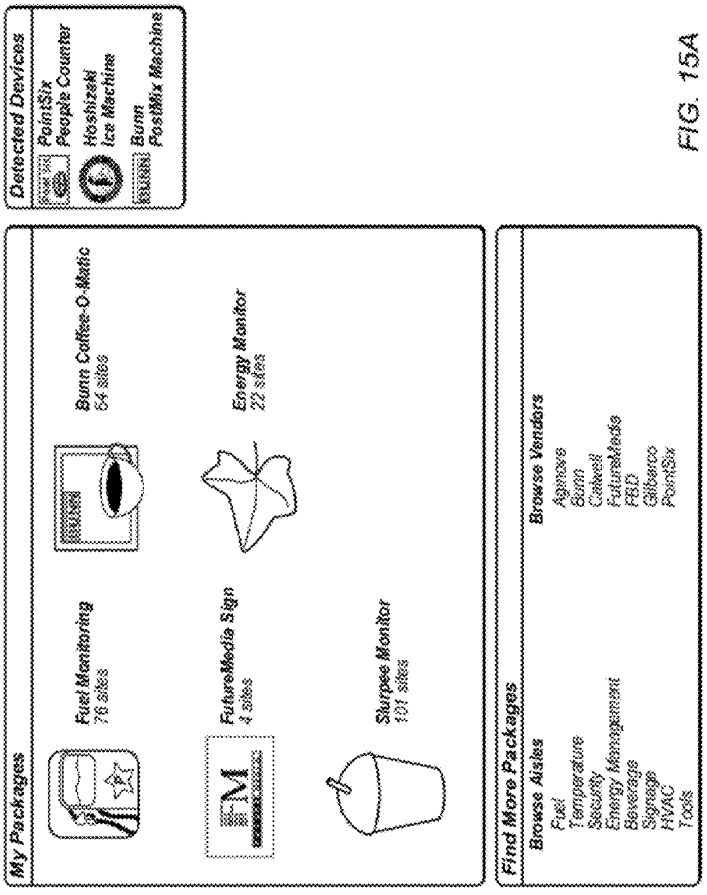

My Packages
Fuel Monitoring
76 sites
Bunn Coffee-O-Matic
54 sites
FutureMedia Sign
4 sites
Energy Monitor
22 sites
Slurpee Monitor
101 sites
Find More Packages
Browse Aisles
Fuel
Temperature
Security
Energy Management
Beverage
Signage
HVAC
Tools
Browse Vendors
Agmore
Bunn
Cakell
FutureMedia
FBD
Gilbarco
PointSix
Detected Devices
PointSix
People Counter
Hoshizaki
Ice Machine
Bunn
FastMix Machine

FIG. 15A

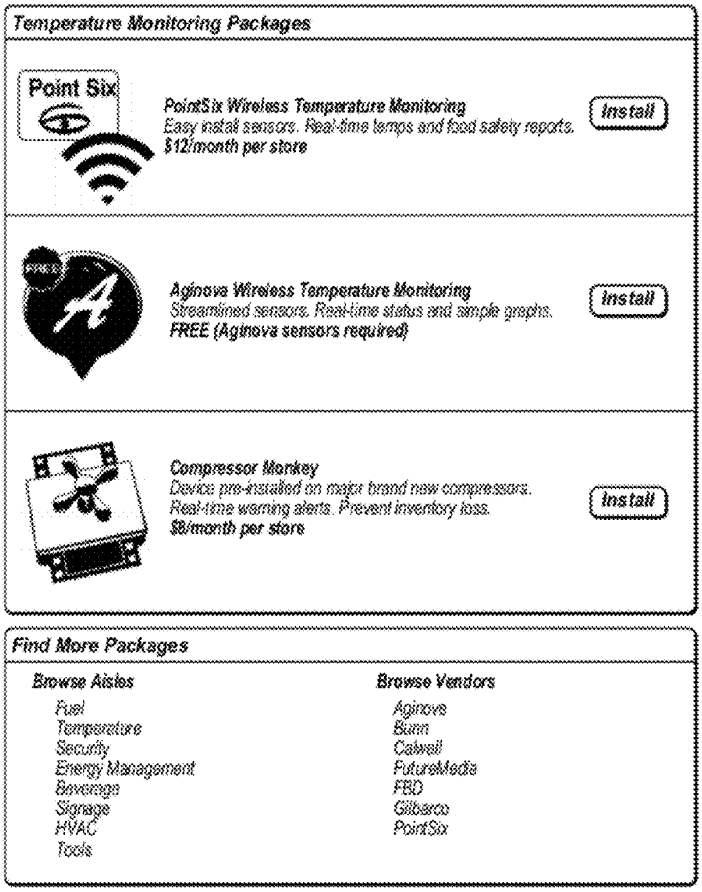

Temperature Monitoring Packages
Point Six
PointSix Wireless Temperature Monitoring
Easy install sensors. Real-time temps and food safety reports.
$12/month per store
Install
Aginova Wireless Temperature Monitoring
Streamlined sensors. Real-time status and simple graphs.
FREE (Aginova sensors required)
Install
Compressor Monkey
Device pre-installed on major brand new compressors.
Real-time warning alerts. Prevent inventory loss.
$8/month per store
Install
Find More Packages
Browse Aisles
Fuel
Temperature
Security
Energy Management
Beverage
Signage
HVAC
Tools
Browse Vendors
Aginova
Bunn
Calwall
FutureMedia
FBD
Gilbarco
PointSix

FIG. 15B

30 / 31

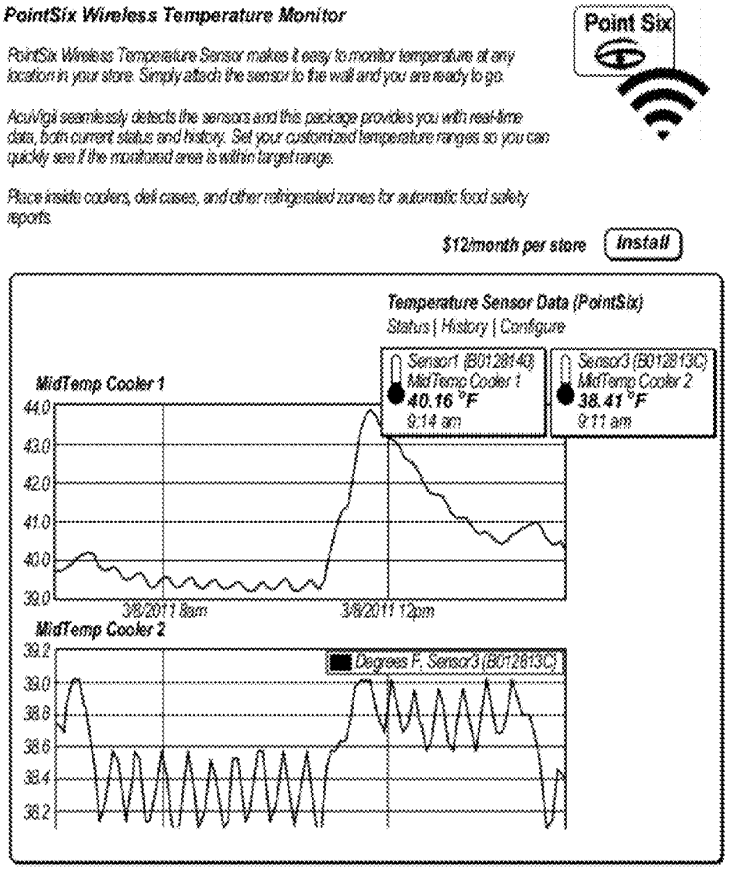

PointSix Wireless Temperature Monitor
PointSix Wireless Temperature Sensor makes it easy to monitor temperature at any location in your store. Simply attach the sensor to the wall and you are ready to go.
Acuvigi seamlessly detects the sensors and this package provides you with real-time data, both current status and history. Set your customized temperature ranges so you can quickly see if the monitored area is within target range.
Place inside coolers, deli cases, and other refrigerated zones for automatic food safety reports.
Point Six
$12/month per store
Install
Temperature Sensor Data (PointSix)
Status | History | Configure
Sensor1 (B012814)
MidTemp Cooler 1
40.16 °F
9:14 am
Sensor3 (B012813C)
MidTemp Cooler 2
38.41 °F
9:11 am
MidTemp Cooler 1
44.0
43.0
42.0
41.0
40.0
39.0
3/8/2011 8am
3/9/2011 12pm
MidTemp Cooler 2
39.2
39.0
38.8
38.6
38.4
38.2
Degrees F, Sensor3 (B012813C)

FIG. 15C

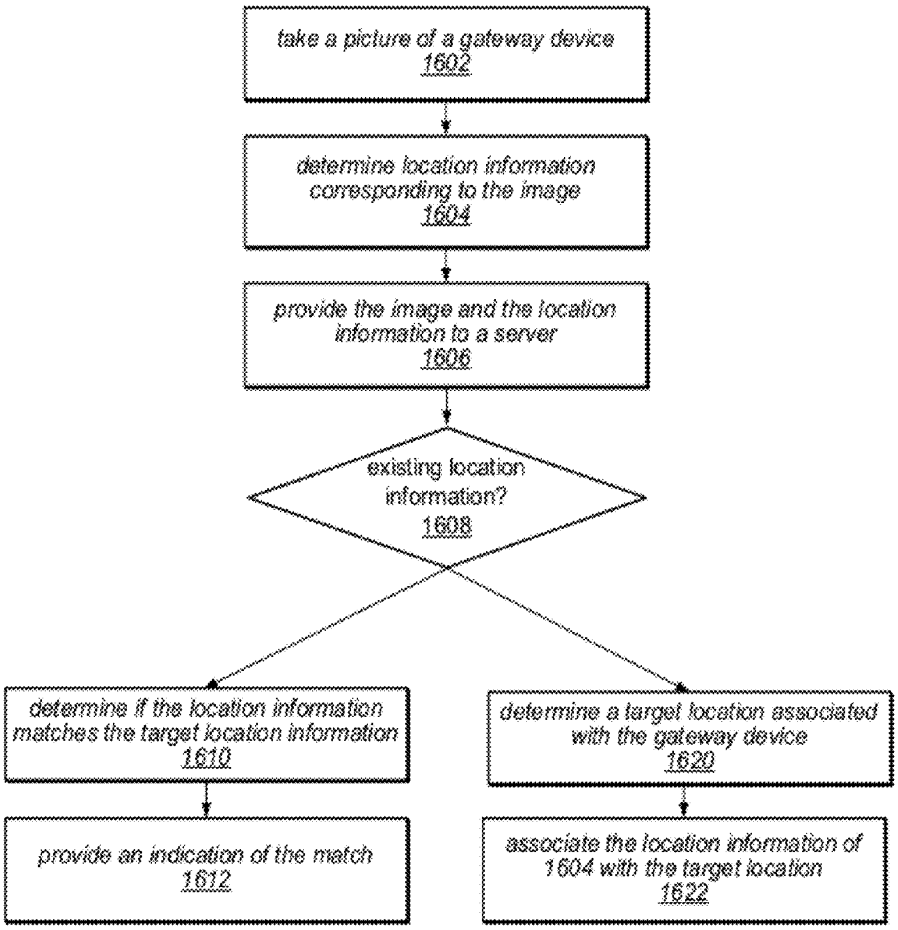

take a picture of a gateway device
1602
determine location information corresponding to the image
1604
provide the image and the location information to a server
1606
existing location information?
1608
determine if the location information matches the target location information
1610
provide an indication of the match
1612
determine a target location associated with the gateway device
1620
associate the location information of 1604 with the target location
1622

FIG. 16

GATEWAY DEVICE APPLICATION DEVELOPMENT SYSTEM

[0001] This application claims benefit of priority of U.S. provisional application Ser. No. 61/479,105 titled "Gateway Device Application Development System" filed Apr. 26, 2011, whose inventors were Brett B. Stewart, Dirk D. Heinen, Margo A. Baxter, and Thomas E. Yemington, which is hereby incorporated by reference in its entirety as though fully and completely set forth herein.

FIELD OF THE INVENTION

[0002] 1. Field of the Invention

[0003] The present invention generally relates to software application development and deployment on remote devices, and more specifically to systems and methods for configuring or managing gateway devices at various locations.

[0004] 2. Description of the Related Art

[0005] Merchants, such as gas stations or convenience stores, often have to manage or monitor many different premise devices (devices at a location or premises) in order to properly operate. For example, a gas station may use several refrigeration units, point of sale (POS) devices, food or drink dispensing machines, electronic signs, fuel tank monitors, etc. It may be difficult for employees or managers of the gas station to perform their day to day activities as well as monitor the status of all of these different premise devices. For example, the temperature of many convenience store refrigeration units may exceed safe or desirable temperatures for food items stored in the refrigeration units, especially ones that can easily spoil. Further, other than spot checking these conditions or devices, a manager of many stores often has difficulty verifying that store managers are properly monitoring or maintaining all of these premise devices. Similar problems exist for different types of merchants, including department stores, retail stores, grocery stores, etc. Additionally, these problems can also be extended to users in their own home, e.g., for managing various computer devices, refrigerators, lights, swimming pools, HVAC equipment, entertainment equipment, plug in automobiles, solar power generation equipment, power meters, etc.

[0006] Accordingly, improvements in monitoring and controlling devices are desired.

SUMMARY OF THE INVENTION

[0007] Various embodiments are described of a system and method for configuring or managing premise devices at various locations (e.g., merchant locations), using gateway devices.

[0008] One or more gateway devices may be installed at various locations. For example, a first gateway device may be installed at a merchant location, such as a convenience store. A second gateway device may be installed at a different merchant location, e.g., a different convenience store operated by the same or different merchant. As another example, gateway devices may be installed at respective users' homes. The gateway devices may be installed in any location where a user or entity wishes to manage (e.g., monitor and/or control) premise devices at the location.

[0009] Each gateway device may be coupled to a plurality of premise devices at the location. For example, the gateway device and the plurality of premise devices may be connected to a same network (e.g., a local area network, possibly provided by the gateway device). The premise devices may be any type of device. For example, a convenience store may include refrigeration devices, cooking devices, point of sale (POS) devices, electronic signs, fuel tank monitoring devices, etc. A home may include various lighting devices, A/C or heating devices, refrigerators, security systems, audio/video systems, gaming consoles, electricity systems, etc.

[0010] The gateway device may also be coupled to a server (e.g., a cloud server) over a wide area network, such as the Internet. It is noted that references to "server" herein may refer to a dedicated server or a cloud server. The server may be used to configure and/or control the gateway device (or premise devices coupled to the gateway device). More specifically, in some embodiments, in order to manage these different premise devices, applications may be developed for execution on gateway devices (or devices coupled to a gateway device). For example, a server may host various ones of these applications and may facilitate selection and deployment of these applications on the gateway device, or devices coupled to the gateway device. In another embodiment, applications may be developed that execute on a server coupled to a gateway device, where the application executes on the server to manage or control a gateway device (and/or devices associated with the gateway device), or multiple gateway devices. The application may instead execute on the server to manage or control premise devices that are connected to one or more gateway devices.

[0011] For example, a software development kit (SDK) or application programming interface (API) may be used to develop applications that can be installed on the gateway device. As noted above, these applications may also be executable by other devices as well. For example, the applications may be executable by the gateway device, a server, and/or other devices (e.g., user devices). The applications may be used to communicate with and control the premise devices at the location, to present data gathered by the premise devices (e.g., related to the premise devices themselves, the environment around the premise devices, etc.), to provide an interface for viewing data or controlling premise devices, and/or to manage or view data corresponding to the location of the gateway device, among other possibilities. The applications may be configured for any possible use regarding the gateway device (or multiple gateway devices), including use by the premise devices managed by the gateway device, presentation of data of the gateway device, presentation of data of the premise devices, presentation of data regarding (e.g., from) a plurality of different gateway devices, presentation of data from different locations (e.g., as gathered from the gateway devices), etc. Accordingly, the applications may be selected and installed on the various devices.

[0012] These applications may be developed and uploaded to a server, referred to as an application server. The application server may execute software to operate as an "application store". The applications described herein are also referred to as "packages", and thus the server may be said to host a "package store" or "PACKAGE WAREHOUSE™". Thus various different application developers (or "package developers") may create various different applications that the upload to the application server for purchase by customers (e.g., merchants) who are operating gateway devices.

[0013] The applications may be selected for installation in various manners. For example, the applications may be selected for installation manually, via user selection. In one

embodiment, the application server hosting the application store maintains a website for the application store (or otherwise presents a user interface) that is accessible by various client systems, e.g., over the Internet. A user may browse (using a web browser) through a plurality of available applications in the application store and then select desired applications for installation, e.g., on the gateway device, the server, and/or other devices. The user may also specify the targets for installation of the selected applications, although this may be determined automatically, as desired. Alternatively, or additionally, the applications may be selected for installation in an automatic fashion.

[0014] For example, the gateway device may provide identification information of the premise devices coupled to the gateway device. In one embodiment, the gateway device may automatically discover these premise devices and then provide corresponding identification information to the application server. The application server may in turn automatically select appropriate applications based on the identification information of the premise devices. These applications may, for example, allow the gateway device to communicate with and/or control the premise devices coupled to the gateway device. The applications may also be used to aggregate and/or present data (e.g., of the environment in the location) gathered by the various premise devices.

[0015] In another embodiment, an application configuration may be automatically determined for the gateway device. The application configuration may be a default application configuration that is associated with the entity or merchant that operates the gateway device, a default application associated with the type of location of the gateway device, etc. The application configuration may also be automatically generated based on other application configurations (e.g., based on the plurality of applications installed on other gateway devices). For example, an application configuration may be generated based on the plurality of applications installed on (or otherwise associated with) other similar gateway devices (e.g., that are also associated with the same entity).

[0016] Once the gateway device has been configured with one or more applications, and hence has the ability to manage the coupled premise devices (e.g., via the installed applications), it may provide information reported by the premise devices to a server and/or control the premise devices in response to messages. For example, the reported information may be presented to a user, e.g., via the server. In one embodiment, the server may provide that information to a user that has logged in to the server (e.g., a website of the server). Alternatively, or additionally, the server may provide reported information to a device of the user (e.g., such as a cell phone or device controlled by the user), which may execute an application (or a plurality of applications) for providing an interface to the server or gateway device. In further embodiments, the user may be able to receive reported information and control various devices via a device coupled to the gateway device, e.g., via local area connection, such as when the Internet is down. In some embodiments, the applications described above may be executed in a distributed fashion, e.g., among the gateway device, the server, and/or other devices, such as user devices.

[0017] In addition, as noted above, the server which interfaces to a respective gateway device may be configured with one or more applications to allow it to manage the gateway device (or multiple gateway devices), manage premise devices coupled to the gateway devices, receive and process data received from one or more gateway devices, etc. For example, applications executing on the server (or the gateway device) may retrieve data for a respective managed device and provide that data for display. In some embodiments, the application may retrieve data for a respective premise device from another source (e.g., not from the premise device or possibly even the gateway device managing the premise device) and provide this data for display, while indicating that the data relates to the particular premise device. As one example, instead of receiving the information from the premise device or gateway device, information regarding the premise device may be provided from a third party, e.g., which stores information related to the premise device. Thus, information related to the premise device may be received from any number of sources, as desired.

[0018] As indicated above, reported information or notifications may be provided to users or devices. For example, certain users may wish to receive reported information or notifications of reported information of the devices managed by the gateway device. This information may be provided to users, e.g., via devices of each user. For example, each user may have a plurality of devices for receiving the reported information. The user may set up rules for determining when and what type of reported information should be provided. Examples of the reported information rules may be, e.g., provision of all of the information, only for specific devices, only during certain time intervals (such as during work hours), only when there is an alarm condition, only when user action is required, etc. The user may also set up rules for determining which of the devices should receive the notification (e.g., specifying a default device, specifying based on various conditions, such as work schedule, vacation schedule, location of the user, location of the user's devices, etc.) as well as how the notification should be received (e.g., email, text message, phone call, instant message, etc.). Note that the reported information or notifications may be provided to devices that are not associated with users as well. For example, such information may be provided to a computer system that is present at the location, but may not be associated with any particular user. The same possibilities for rules apply to devices not associated with users.

[0019] Thus, embodiments of the invention may provide a much more convenient and powerful mechanism for enabling merchants to manage and control the various premise devices in their store. For example, by managing the premise devices at the location, a user may be able to monitor and/or control the environment at the location (e.g., temperature, security, lighting, music, etc.). In particular, embodiments may provide a platform for application developers to create and provide for sale application programs used for monitoring and/or controlling premise devices, such as premise devices commonly found at a merchant location. Further, embodiments may provide a convenient platform for customers/merchants to browse and select application programs for deployment and execution on their merchant gateway devices (or servers or managed devices) for monitoring/controlling the various premise devices present in their merchant location.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] A better understanding of the present invention may be obtained when the following detailed description is considered in conjunction with the following drawings, in which:

[0021] FIGS. **1**A and **1**B illustrate two exemplary systems, according to some embodiments;

[0022] FIGS. **2** and **3** are block diagrams of exemplary systems, according to some embodiments;

[0023] FIGS. **4**A and **4**B are block diagrams of a management device, according to one embodiment;

[0024] FIG. **5** is a flowchart diagram illustrating an embodiment of a method for developing an application for a gateway device;

[0025] FIG. **6** is an exemplary diagram illustrating an embodiment of use of an application programming interface for an application;

[0026] FIG. **7** is a flowchart diagram illustrating an embodiment of a method for storing and providing a plurality of applications to gateway devices;

[0027] FIG. **8** is a flowchart diagram illustrating an embodiment of a method for automatically installing a plurality of applications on a gateway device based on detected devices;

[0028] FIG. **9** is a flowchart diagram illustrating an embodiment of a method for automatically installing a plurality of applications based on an application configuration;

[0029] FIG. **10** is a flowchart diagram illustrating an embodiment of a method for notifying users of reported information of managed devices;

[0030] FIG. **11** is a flowchart diagram illustrating an embodiment of a method for providing a user interface to a user;

[0031] FIG. **12** is a flowchart diagram illustrating an embodiment of a method for providing advertising for display;

[0032] FIGS. **13**A-**15**C are exemplary screen shots corresponding to the described methods, according to one embodiment; and

[0033] FIG. **16** is a flowchart diagram illustrating an embodiment of a method for installing a gateway device.

[0034] While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and are herein described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

DETAILED DESCRIPTION OF THE EMBODIMENTS

Incorporation by Reference

[0035] U.S. Patent Application titled "Multiserver Merchant Gateway", Ser. No. 11/267,566, which was filed Nov. 4, 2005, whose inventors are Brett B. Stewart and Charles O. Golson,

[0036] U.S. Pat. No. 7,724,154, titled "System and Method for Food Service Storage Bin Monitoring", which was filed Oct. 30, 2006, whose inventors work Brett B. Stewart and Drik D. Heinen, is hereby incorporated by reference in its entirety as though fully and completely set forth herein.

Terms

[0037] The following is a glossary of terms used in the present application:

[0038] Memory Medium—Any of various types of memory devices or storage devices. The term "memory medium" is intended to include an installation medium, e.g., a CD-ROM, floppy disks, or tape device; a computer system memory or random access memory such as DRAM, DDR RAM, SRAM, EDO RAM, Rambus RAM, etc.; or a non-volatile memory such as a magnetic media, e.g., a hard drive, or optical storage. The memory medium may comprise other types of memory as well, or combinations thereof. In addition, the memory medium may be located in a first computer in which the programs are executed, or may be located in a second different computer which connects to the first computer over a network, such as the Internet. In the latter instance, the second computer may provide program instructions to the first computer for execution. The term "memory medium" may include two or more memory mediums which may reside in different locations, e.g., in different computers that are connected over a network.

[0039] Carrier Medium—a memory medium as described above, as well as a physical transmission medium, such as a bus, network, and/or other physical transmission medium that conveys signals such as electrical, electromagnetic, or digital signals.

[0040] Computer System—any of various types of computing or processing systems, including a personal computer system (PC), mainframe computer system, workstation, network appliance, Internet appliance, personal digital assistant (PDA), smart phone, television system, grid computing system, or other device or combinations of devices. In general, the term "computer system" can be broadly defined to encompass any device (or combination of devices) having at least one processor that executes instructions from a memory medium.

[0041] Automatically—refers to an action or operation performed by a computer system (e.g., software executed by the computer system) or device (e.g., circuitry, programmable hardware elements, ASICs, etc.), without user input directly specifying or performing the action or operation. Thus the term "automatically" is in contrast to an operation being manually performed or specified by the user, where the user provides input to directly perform the operation. An automatic procedure may be initiated by input provided by the user, but the subsequent actions that are performed "automatically" are not specified by the user, i.e., are not performed "manually", where the user specifies each action to perform. For example, a user filling out an electronic form by selecting each field and providing input specifying information (e.g., by typing information, selecting check boxes, radio selections, etc.) is filling out the form manually, even though the computer system must update the form in response to the user actions. The form may be automatically filled out by the computer system where the computer system (e.g., software executing on the computer system) analyzes the fields of the form and fills in the form without any user input specifying the answers to the fields. As indicated above, the user may invoke the automatic filling of the form, but is not involved in the actual filling of the form (e.g., the user is not manually specifying answers to fields but rather they are being automatically completed). The present specification provides various examples of operations being automatically performed in response to actions the user has taken.

[0042] Gateway Device—refers to a device that is configured to couple to a wide area network, such as the Internet, and also couple (either wired or wirelessly) to one or more devices, and which is configurable with one or more application programs to monitor and/or control the one or more devices. The term "Merchant Gateway Device" refers to a Gateway Device that is configured to be placed at a merchant

location, e.g., which is configurable with one or more application programs to monitor and/or control one or more devices that would typically be located at a merchant location.

FIGS. 1A and 1B—Exemplary Systems

[0043] FIGS. 1A and 1B illustrate various exemplary systems that may implement embodiments described herein. However, these systems are exemplary only and other systems and configurations are envisioned.

[0044] In FIG. 1A, an exemplary merchant location (in this case, a convenience store) is illustrated. The convenience store includes a gateway device (in this case, a "merchant gateway device", shown as "MGD"), which may manage various premise devices of the convenience store. As used herein, "management of devices" (or similar phrases) refers to monitoring and/or controlling devices. For example, management may include receiving information gathered by the devices (e.g., status information or reported information) and/or modifying operation of the devices (e.g., by providing control messages). As shown, the gateway device may be coupled (in a wired or wireless fashion) to various premise devices in the convenience store. The gateway device may be installed at the location in a relatively straightforward manner. For example, the gateway device may simply be screwed into the wall and connected to a local Internet connection or network connection. In some embodiments, the gateway device may be a ruggedized device and may be stored in an electrical closet.

[0045] The various premise devices to which the gateway device may be coupled include: one or more point of sale devices (POSs, such as credit card readers for purchasing fuel, e.g., located at the pump, or items sold by the convenience store, e.g., located in the store), one or more liquid dispensing devices (e.g., a Slurpee™ machine, coffee dispenser, fountain drink dispenser, etc.), one or more refrigeration devices (e.g., for cooling various food or drink items), or one or more fuel tank devices (e.g., for monitoring the level of fuel in the tank, dispensing the fuel to customers, etc.). The gateway device may manage other premise devices than those shown in FIG. 1A; for example, the gateway device may manage vending machines (e.g., for food items or other types of items, such as movies, devices, peripherals, etc.), heating devices (e.g., for keeping food items hot, such as hot dogs), washing devices (e.g., car washes or laundry machines), electronic signs (e.g., advertising the price of the gas at the convenience store, among other possibilities), ATM machines, air conditioning units or heating units of the convenience store, store security devices/systems (e.g., cameras or alarm systems, etc.), ice machines, soft serve dispensing machines, cash registers, safes, any local computers or servers, the local area network of the convenience store, any network connection(s) (e.g., the Internet) of the convenience store, wireless network(s), etc.

[0046] As indicated above, the gateway device may monitor or control these various premise devices. For example, the gateway device may monitor the temperature inside or near a refrigeration device. The gateway device may be further configured to control the refrigeration device, e.g., by turning on the refrigeration when the temperature readings are too high (e.g., when the temperature is outside of food safety levels), providing an alert when the refrigeration device is working improperly, etc. Note that while the gateway device may interact directly with the refrigeration device, e.g., where the refrigeration device includes circuitry that is intelligent enough to communicate with other devices, the gateway device may alternatively interact with various sensors that are coupled or connected to the refrigeration device. For example, the gateway device may be in communication with a temperature sensor inside the refrigeration device in order to monitor the temperature of the refrigeration device. In some embodiments, the gateway device may monitor food storage bins, such as in the manner described in U.S. Pat. No. 7,724,154, which was incorporated in its entirety above. Other devices that do not include communication circuitry (or other required circuitry) may be monitored via sensors as well.

[0047] As another example, the gateway device may manage a fuel tank device. For example, the fuel tank device (or a sensor within the fuel tank) may provide reported information, such as readings indicating the level of fuel stored in the fuel tank. The gateway device may monitor the status of the liquid dispensing device. For example, the gateway device may monitor the status of various nozzles that dispense different liquids, the status of the level of ice in the liquid dispensing device, the temperature of the liquid stored in or provided by the liquid dispensing device, the level of liquid stored in the liquid dispensing device, etc. Again, as noted above, the gateway device may monitor one or more sensors coupled to the liquid dispensing device to accomplish the above functions.

[0048] Additionally, the gateway device may manage various POS devices at the convenience store. For example, the gateway device may monitor or control POS devices that are at the gas pump(s) of the convenience store, POS devices at a cash register of the convenience store, etc. The gateway device may monitor transactions that are performed by the POS devices. In some embodiments, the gateway device may perform the functionality described in U.S. patent application Ser. No. 11/267,566, which was incorporated by reference in its entirety above. Thus, the gateway device may manage any type of device that it is coupled to.

[0049] Generally, while the above descriptions are specific to monitoring devices of the location, the methods herein may relate to monitoring the environment or specific conditions of the location where the gateway device resides. For example, while a refrigerator may be the device that is being monitored, the temperature sensors may simply report the temperatures at specific locations inside the refrigerator in order to report the condition of that area in the location. Thus, while descriptions herein focus on the status or condition of individual premise devices, ultimately, the information may relate to specific conditions or environments of the location. Thus, the reported information provided from a device may reflect the environment of the location near the device (e.g., as measured by the device), although it can also include status or condition information of the device itself, depending on the type of device or measurement.

[0050] As shown, the gateway device may be in communication with a network, e.g., a wide area network (WAN), such as the Internet. The gateway device may be in communication with the Internet via a router or modem that provides the Internet connection. Alternatively, the gateway device may itself include logic (e.g., circuitry such as an ASIC or FPGA and/or a processor and memory storing program instructions) to perform these wide area network connectivity functions. For example, the gateway device may include a broadband modem for providing a communication link between the convenience store and the Internet. The gateway device may also

act as a router (e.g., a wired and/or wireless router) which couples to the wide area network and premise devices at the location.

[0051] In one embodiment, the gateway device may provide a local area network (LAN) that at least a subset of the premise devices may connect to, e.g., in order to communicate with the gateway device and/or the Internet. The gateway device may provide a wireless hotspot for the convenience store, e.g., which may be accessible by employees of the convenience store and/or customers of the convenience store, as desired. In further embodiments, the gateway device may include security software or circuitry, e.g., to provide a firewall for the Internet connection, perform communication encryption, protect credit card information provided over the network connection, etc.

[0052] As also shown, the gateway device may be in communication with a computer system in the convenience store (e.g., in a wired or wireless fashion, as desired). For example, a manager or employee of the convenience store may use the computer system to perform various tasks related to the convenience store, e.g., viewing security feeds, performing bookkeeping, etc. In some embodiments, the gateway device may be configured to manage the computer system. In one embodiment, the gateway device may monitor various programs that are executing on the computer system and/or the Internet traffic that the computer system is receiving or transmitting, among other possibilities.

[0053] Alternatively, or additionally, the computer system in the convenience store may be used to manage or monitor the various programs executing on the gateway device. For example, the gateway device may provide status messages (e.g., containing reported information) or control messages (e.g., for controlling one or more devices) regarding the plurality of devices of the convenience store to the computer system. For example, an employee or manager of the store may monitor or control the various devices of the store using an application that executes on the computer system and which is in communication with the gateway device. For example, the gateway device may include a web server that hosts a website used for these monitor/control functions. The computer system may execute a web browser and may visit the website (e.g., hosted by the gateway device or elsewhere) that is usable to manage the devices at the convenience store. Alternatively, a custom application may be executed by the computer system to manage the devices via the gateway device. Note that while the computer system is shown as a typical desktop computer system, any type of system may be used, e.g., laptop computers, portable or mobile devices, such as cell phones, netbook computers, tablet computers, etc. In general, in this embodiment, the computing system may be any device that is able to provide an interface to the gateway device for a user. For example, it may simply be a separate display that is able to display information provided by the gateway device (e.g., over a wireless network).

[0054] The computer system may be in communication with a server, referred to as a management server (e.g., a cloud server, as described below). The management server may be located on the Internet, or may be in direct communication with the gateway device (e.g., without accessing the Internet), and/or both, as desired. The gateway device may be in direct communication with the management server to report information gathered by the devices or perform control actions, among other possibilities. A user may be able to configure the gateway device directly (e.g., via a web site hosted by the gateway device or via an application executing on the computer system) or may be able to configure the gateway device over the Internet, via the management server.

[0055] As shown in FIG. **1**A (and as discussed above), the gateway device may be in communication with one or more servers (e.g., which may be a single server or a server farm, as desired) over the Internet connection. The one or more servers (which may be a "cloud server" or "cloud servers") may provide various functionality for the convenience store, the gateway device, and/or other devices or users, as desired.

[0056] For example, the management server may be configured to receive and store information regarding the premise devices managed by the gateway device and possibly status information of the gateway device or the convenience store. For example, the management server may store configuration information of the various premise devices, network status information of the connection between the management server and the gateway device, etc. In one embodiment, the management server may be configured to receive messages comprising information gathered from each of (or at least a subset of) the plurality of premise devices at the convenience store and store reported information for the premise devices, e.g., in a database. The reported information may indicate a current condition of the premise device, a condition of the environment of the location near the premise device, any data measured by the premise device, etc. For example, the management server may store the current temperature (and/or a history of temperatures) for a first refrigeration device at the convenience store. The management server may also receive messages regarding various transactions performed by a first POS device at the convenience store and store information regarding those transactions. The management server may also receive messages regarding the fuel level in a first fuel tank at the convenience store and store the current level (or a history of fuel levels) in response to the message. Other types of reported information or messages may be received by the management server.

[0057] In some embodiments, the management server may receive messages indicating specific alarms or events associated with the premise devices managed by the gateway device. For example, the gateway device may send information indicating when a temperature of the refrigeration device exceeds a threshold (e.g., based on food safety requirements or otherwise); when the fuel level in the fuel tank is below a threshold level; when the liquid dispensing device is low on a respective liquid, out of ice, etc.; when the POS performs or attempts to perform a transaction above a certain level, etc. The gateway device may also send information when a security alarm has been detected (e.g., when the convenience store is broken in to), when an employee triggers an alarm (e.g., when the convenience store is being robbed), etc.

[0058] In one embodiment, rather than the management server receiving indications of a specific alarm, the management server may be configured to automatically evaluate the incoming reported information to determine whether such conditions have arisen. For example, rather than receiving a message indicating that the fuel tank is below a certain level, the management server may be configured to compare the current fuel tank level (e.g., as reported in a status message) with a threshold to determine if the fuel tank has reached a refueling level. Similarly, the management server may be configured to compare a current temperature of a refrigerator

with a threshold temperature to determine if the refrigerator is operating adequately and/or if the temperature is above food safety levels.

[0059] Note that while the reported information is provided to the management server by the gateway device, other entities or devices may provide reported information related to the convenience store (or generally, location of the gateway device) to the management server. For example, a third party server may provide information related to a device located at the location of the gateway device to the management server, and the management server may receive that data and report it for the location of the gateway device (e.g., as if it were provided from the gateway device). As one specific example, an energy company may store information related to the energy consumption of the convenience store in FIG. **1**A, but that information may not be available to the gateway device (e.g., by being coupled to a smart sensor that monitors energy use of the convenience store). Accordingly, a server storing that energy information may be configured to provide the information to the management server, which can then include that information in an interface to a user. Thus, a user may be able to view the energy use information from the management server even though it was not provided via the gateway device. Accordingly, the management server may be configured to aggregate information concerning a location from a plurality of sources, including the gateway device and/or other entities that gather or store such data. As another example, a manufacturer of a device may send reported information (or alerts) based on schedules. For example, an ice machine manufacturer may send information to the management server to update the status to "replace filter" every on a periodic basis (e.g., every six months). This may be extended to any party or particular information related to the location (e.g., a device at the location). Further, weather information related to the location may be provided to the management server (e.g., from a weather service).

[0060] The management server may also be configured to provide information related to the convenience store to various users, entities, and/or devices (e.g., user devices). For example, the management server may receive information gathered by the various premise devices and may provide the information for display on a website that may be visited by authorized users (e.g., users that are associated with the specific convenience store). Alternatively, or additionally, these users (or other devices) may execute programs on computer systems or devices designed to interface with the management server to retrieve and provide the information for presentation for the user. In one embodiment, a manager of the convenience store may be able to execute such a program, or browse to a web site provided by the management server, to view the current status of the convenience store, e.g., as reported by the various premise devices. For example, the manager may be able to quickly determine the fuel level in the fuel tanks of the convenience store, determine whether the food sold at the convenience store are currently at proper temperatures (and/or if they have been over a period of time, such as the last day, last week, last month, or any period of time specified by the user), determine the current prices of the gas sold by the convenience store, determine the number or amount of sales performed by the convenience store over various time periods (such as those described above, although they can be specified manually by the user), etc.

[0061] Note that the computer system inside of the convenience store may similarly access the management server (or the gateway device) to determine the status of the convenience store (e.g., based on the reported information), as desired. Thus, the gateway device may monitor the status or condition of various premise devices in the convenience store (or generally various conditions within the convenience store, e.g., as reported by premise devices), provide them to the management server, and the management server may provide that information to various other entities or users, which may be local or remote to the location of the convenience store, as desired. Alternatively, the gateway device may include web server functionality, and thus the gateway device may be accessible by client browsers located on the computer system or the management server.

[0062] In some embodiments, the gateway device may actively push the information related to the various premise devices to the management server (e.g., rather than the management server requesting the information). Accordingly, the management server may receive the reported information from one or more gateway devices. The management server may report an issue or actively try to contact a gateway device (e.g., for pulling the data) if it has not "checked in" or reported information on its normal schedule (e.g., once a minute, once every 5 minutes, 10 minutes, 15 minutes, 30 minutes, 60 minutes, etc.). In alternate embodiments, the management server may send requests for information, e.g., on a periodic basis.

[0063] Similarly, the management server may actively push data (e.g., reported information or alerts) to various entities or users. For example, the management server may be configured to provide information to various users when alert or alarm conditions occur. As one example, the management server may be configured to notify any managers or employees of the convenience store when any general alarm condition is reached (e.g., when the fuel tank is below the threshold, after a fuel spill or other environmental hazard, when the refrigeration temperature is above a threshold, when the door of a refrigerator has been left open, etc.). The management server may then send messages to these users in various manners, e.g., through email, text messaging, instant messaging, phone calls, etc. Note that the management server may store more than one contact information or contact method for each user or device (e.g., including one or more email addresses, one or more phone numbers, one or more associated devices, etc.).

[0064] The management server may store contact information and conditions for a plurality of different users. For example, the management server may be aware of the work schedule of various employees and may only notify such employees when they are currently working. Thus, as one example, the cloud server may notify the day manager when alarm conditions are reached during the day, but may notify the night manager when alarm conditions are reached during the night. However, some users or entities may always be notified when such conditions are reached, e.g., including the computer system of the convenience store (to alert any nearby employees), a store owner, etc.

[0065] The management server may be configured to alert specific users or entities for specific premise devices at the convenience store. For example, the management server may be configured to request a fuel refill (e.g., by providing a delivery request to a gas delivery company) when the fuel tank level is below a threshold. Similarly, the management server may be configured to contact an IT support person when the local area network at the convenience store is no

longer functioning, when the Internet connection from the gateway device is not functioning, when the computer system is malfunctioning, etc. Further, manufacturers of individual devices may be notified of the existence of and/or condition of various premise devices at the location. For example, the management server may notify a manufacturer when one of its devices is detected at a convenience store and may provide status or condition information of the device to the manufacturer (e.g., for information gathering, maintenance purposes, warranty purposes, etc.). Thus, rather than notifying users or entities in response to any event or status of the convenience store, some entities may only be notified based on reported information or alerts of a particular device at the convenience store. The management server may store rules in order to perform the notifications described above, e.g., in a database, as desired. Additionally, an administrator may be able to configure the notification rules, using an administrative interface for the convenience store. As one example, an administrator may configure various settings related to the gateway device, particular premise devices of the convenience store, and/or the convenience store via a website provided by the management server.

[0066] The management server may also allow for configuration of the gateway device or premise devices managed by the gateway device in the convenience store. For example, a manager may wish to update prices (e.g., of an electronic sign, a fuel dispenser, various food items, etc.) at the convenience store. For gas prices in particular, the manager may access the management server via an application executing on a computer system or device (such as a web browser or custom application) and set the gas prices displayed on electronic signs at the convenience store, the prices of gas at the fuel pumps (e.g., on the POS devices), etc. Thus, instead of the manager calling an employee of the convenience store to change all of these settings, the manager may simply change them remotely without interrupting any work at the convenience store. The manager (or any other authorized user) may be able to modify other settings as well. For example, the manager may change the current temperature of the convenience store by changing the desired temperature of heaters or coolers at the convenience store. The manager may be able to modify the temperatures of various refrigerator units, heating units, or the settings of any controllable device at the convenience store. The manager may be able to configure any device or setting at the convenience store using the management server.

[0067] In one embodiment, the management server may achieve various features described above by executing one or more "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. **1**A. Embodiments of the invention provide an improved system for developers to create and sell such applications and an improved system for users to browse and purchase such applications.

[0068] As described herein, an application server may store and/or provide a plurality of "applications" or "packages" (described below as "applications") for use for the convenience store, in the example of FIG. **1**A. The application server may be the same server as the management server, or may be a different server. Each of the management server and the application server may be cloud servers. In the following description, the terms "server" and "cloud server" are used to refer generically either or both of the management server and the application server (and/or other associated servers)

[0069] For example, a manager may browse the application server and select from a plurality of different applications that may be installed and executed by the gateway device and/or the management server (and possibly by devices associated with the convenience store). There are many different types of applications that are possible and may be desired by a manager or other user of the convenience store. While the following descriptions provide some examples of such applications, these are in no means limiting, and further applications are envisioned.

[0070] In one embodiment, the management server and the gateway device may be initially configured with a framework or default configuration, before a manager or other user installs such applications. For example, the gateway device may be initially configured to report information gathered by any devices with which it can communicate. The management server, in turn, may be configured to provide such information to one or more users, as described above. The framework may also allow for easy installation of applications in response to user selection of applications to install.

[0071] In one embodiment, the gateway device may not be aware of, or may not be able to manage, various ones of the premise devices of the convenience store. A user may accordingly install one or more applications in order to allow the gateway device to manage the convenience store devices. For example, the user may choose to install an application specific to the type of fuel tank device (or sensor) that is present at the convenience store. Accordingly, the server may provide at least a portion of the application to the gateway device. The gateway device may then install and execute the provided application on the gateway device, and the installed application may allow the gateway device to manage the fuel tank device (e.g., receive reported information from the fuel tank device and/or control the fuel tank device, as desired). In alternate embodiments, rather than a user specifying installation of the application, the server may automatically install the application without requiring user input selecting the application. For example, the gateway device may detect the presence of the fuel tank device and provide identification information to the server. The server may then automatically provide a corresponding application to allow the gateway device to communicate with the fuel tank device.

[0072] In some embodiments, a portion of the application may be installed on the management server. For example, the application may include a first portion (provided to the gateway device) to adequately perform management of the fuel tank device, and the application may include a second portion for user interaction. For example, the second portion of the application may be executed by the server to provide the received reported information to various other devices for presentation, provide a user interface (e.g., a graphical user interface or "GUI") to control or modify settings associated with the fuel tank device, provide a user interface to set up fuel level thresholds (for when the fuel tank should be refilled), set up contact information associated with the fuel tank (e.g., whom to contact when the fuel level is below a threshold), purchase order information for requisitioning more fuel when the fuel tank is depleted, etc. The second portion may be a pluggable module into a management application (e.g., within a website) provided by the server. In one embodiment, the management server may include a database of available interfaces (e.g., associated with each application that is configured for execution on the management server) and, in response to an application being installed, associate the inter-

face (or application portion) with the gateway device or user for which the application was installed. Thus, the application portion may already be present on the management server and may simply be activated upon installation (e.g., by associating a user or location with the application in a database). Thus, the second portion may "add on" to a management interface for the convenience store provided by the server.

[0073] As indicated above, another portion (e.g., a third portion) of the application may also execute on other devices. For example, various users associated with the convenience store may use various types of devices (e.g., devices associated with users or "user devices") to monitor or control premise devices of the convenience store. For example, the devices may include various computer systems or devices, such as a desktop computer, a laptop computer, a tablet computer, a netbook, a personal digital assistant (PDA), mobile phone, etc. The third portion of the application may be installed on one or more of these devices to allow respective users to manage the devices of the convenience store. In some embodiments, the third portion of the application may be a stand alone application for the devices. Alternatively, the portion of the application may be a modular portion that may "plug in" or operate within a larger program, e.g., for monitoring the convenience store, or a plurality of stores or locations, as desired. Thus, following the fuel tank device application above, a stand alone fuel tank device application may be installed on a user's device for the user to monitor or control the fuel tank device. Alternatively, the user may have an application for managing the convenience store and the fuel tank device application may be installed as a modular addition to that program, in addition to other modular additions for other devices or functions associated with the store. Note that this embodiment may be extended to devices that are not associated with users, as desired.

[0074] Similarly, applications specific to various POS devices, liquid dispensing machines, refrigeration devices, or any devices of the convenience store may be installed on the gateway device, the server, and/or other devices. These applications may be preferably developed by the manufacturers of the respective convenience store devices (e.g., where interaction with the devices uses a proprietary protocol); however, in alternate embodiments, the applications may be generated by any party (e.g., 3rd party developers). In one embodiment, developers may use an API and/or SDK in order to develop the applications stored on the application server. The API or SDK may be an open API or SDK or may only be provided to authorized users or companies.

[0075] Additionally, while the above described applications are specific to individual types (or even models) of premise devices in the convenience store, there may be applications that are not device specific. For example, a web hosting application may be deployed on the gateway device that allows the gateway device to become a web server, e.g., for providing a web site of the convenience store. In one embodiment, there may be an administrative application that relates to handling employee schedules and proper notification rules for various alarms or devices, e.g., allowing implementation of various embodiments described above. Additionally, there may be an application that provides a specific type of interface for viewing the reported information of the various devices of the convenience store, alerts of the convenience store, use of resources of the convenience store, etc. For example, an application may provide a graph indicating fuel consumption across all of the fuel tanks of the convenience store. Another application may monitor or keep track of transactions performed by the POS devices of the convenience store, e.g., to provide bookkeeping functionality, to analyze transactions (e.g., to determine trends of the data, such as peak business or profitability over various time periods), to visualize the transaction data, etc. Further applications are envisioned.

[0076] Thus, FIG. **1**A illustrates an exemplary system where the gateway device manages devices at a convenience store.

[0077] FIG. **1**B illustrates an alternative system where the gateway device manages premise devices at a user's home. In this example, the gateway device may provide an interface for and/or manage the user's solar panel(s), fish tank, audio/video equipment (such as a TV, DVR, receiver, DVD player), refrigerator, and/or personal devices (e.g., computers, cell phones, tablet computers, home theatre computers, etc.). Further types of premise devices may be managed by the gateway device, such as lights of the home, the garage door (or other devices associated with the garage), oven(s) (such as microwave, gas, and/or electric), heating and/or cooling systems, attic fans, fireplace(s), pool systems, security systems, gaming consoles, modems, routers, web servers (or other software executing on various systems), phone systems, answering machines, videoconferencing systems, health monitoring devices (such as a pacemaker), smoke detectors, carbon monoxide detectors, washing machines, drying machines, etc.

[0078] FIG. **1**B may operate similar to FIG. **1**A except that the gateway device is placed in a home and controls premise devices of the home. Additionally, in the embodiment of FIG. **1**B, the separate management server and application server of FIG. **1**A have been combined as a "cloud server". Note that these two servers may still be physically separate, or may be implemented by the same one or more servers with separate software functionality, as desired. Thus, the cloud server may be implemented as two separate sets of servers (one for management and one for applications) or may be implemented together, among other possibilities.

[0079] In the embodiment of FIG. **1**B, a head of family (or any subset of the family members, among other possible authorized users) may be the party that is notified for most reported or alert information. In further embodiments, the gateway device or the server may notify various third parties for different alerts or reported information. For example, in the event that a communication line ceases functioning (e.g., an Internet connection, phone line, etc.), a utility company may be automatically notified in order to troubleshoot the problem. In another embodiment, the electric company may receive reported information of the amount of power the solar panels have provided back into the electricity grid. Further, the solar panel manufacturer (or a company associated with the solar panels) may be notified if they cease to function properly. Further, police may be notified when a security breach is detected, firefighters when a fire is detected, paramedics for a health emergency, etc.

[0080] Additionally, individual family members may be associated with particular premise devices. For example, if the fish tank is located in a child's room (or is taken care of by the child), the child may be notified when the fish need to be fed, if the tank temperature is too high or too low, if a dead fish has been identified, etc.). Similarly, reported information associated with a game console may be provided to the resident gamer of the household rather than all of the family members.

[0081] Thus, FIG. 1B illustrates an exemplary home which includes a gateway device that manages various premise devices of the home. The gateway device may be deployed in a variety of other systems or locations. Essentially, the gateway device may be deployed in any environment for managing one or more devices. For example, the gateway device may be a merchant gateway device that is deployed at a specific merchant location of a merchant. The merchant may be a restaurant (e.g., fast food, family dining, etc.), department store, clothing store, retail store, or any type of store, as desired. The gateway device may also be appropriate at various companies (e.g., for branch offices or main offices, as desired).

[0082] In further embodiments, the monitoring may be extended to delivery trucks (e.g., for the delivery truck itself or for the store(s) which receives the delivery). As an example, a food delivery truck may include a gateway device that reports on the status of the various food items in the delivery truck (e.g., the temperature inside, whether it has exceeded a threshold temperature, etc.). The store receiving the goods (or the truck's company itself) may monitor the condition of the food/truck, e.g., to ensure food safety measures are properly followed. Thus, the gateway device may be used in a variety of different locations or systems.

FIG. 2—Block Diagram of Exemplary System

[0083] FIG. 2 illustrates a block diagram of one embodiment of an exemplary system. As shown, the gateway device may be coupled to various local or premise devices, e.g., of a convenience store, merchant location, or other location. As shown, the gateway device may include a router for providing a local area network at the location (e.g., for the local devices). The gateway device may also include a firewall between the Internet and the local area network.

[0084] In the embodiment, of FIG. 2, the gateway device may have a separable (or simply, separate) display, which the gateway device may communicate with in a wired or wireless fashion (e.g., over the local area network), as desired. For example, the gateway device may be stored in a server closet or room, in which case, a display collocated with the gateway device may have limited use for employees wishing to interact with the gateway device or view reported information. Accordingly, the display may be separated from the gateway device and placed in a more convenient location, e.g., where one or more employees can view reported information of the various local devices and/or control the devices (e.g., by using input device(s) that may also be separated from the gateway device). In alternate embodiments, similar to discussions above, other users may simply use other devices to monitor and control the devices, and a separable display may not be necessary or desired. Thus, the gateway device may be a "headless" or display free device. The gateway device may be coupled to the Internet as well as local devices at the merchant location.

[0085] In some embodiments, such as shown in the embodiment shown in FIG. 2, the gateway device may have a more direct connection to the server than through the Internet. For example, the gateway device may be coupled to the server via a private network or alternate WAN, e.g., that may provide more security than over the Internet. Alternatively, or additionally, the connection to the server may be via a virtual private network (VPN). Regardless of the type or method of connection, the gateway device may be configured to provide reported information to the server, which may store the information (and/or derived information) in the cloud based storage that is coupled to the server. Additionally, one or more clients may be coupled to the Internet and may receive various reported information or notifications from the gateway device and/or the server. Further, as indicated above, the server and/or clients may control various devices via the gateway device. Note that the clients may perform this control by communicating with the server or by directly communicating with the gateway device, as desired.

FIG. 3—Exemplary System with Multiple Gateway Devices and Clients

[0086] FIG. 3 illustrates a block diagram of an exemplary system including a plurality of gateway devices and clients, according to one embodiment. As shown, each of the gateway devices A-D, each of the clients A-D, the server, and the storage are coupled to the Internet. The cloud server, gateway devices, and clients may all generally operate according to embodiments described above. As noted above, the server may be a cloud server and the storage may be cloud-based storage.

[0087] In some embodiments, each of the gateway devices may be associated with a common company (e.g., merchant) or user. For example, gateway device A may correspond to a first convenience store of a convenience store chain, gateway device B may correspond to a second convenience store of the convenience store chain, gateway device C may correspond to a third convenience store of the convenience store chain, and gateway device D may correspond to a fourth convenience store of the convenience store chain. Following this particular example, each of the clients may be associated with individual ones of the convenience stores, premise devices within individual convenience store(s), with the convenience store chain, etc., as desired.

[0088] For example, client A may correspond to a device used by an owner of the convenience store chain. Accordingly, the owner of the chain may receive reported information from and be able to control any of the devices at the various convenience stores. For example, the owner may be able to view data or control devices of the first convenience store, the second convenience store, the third convenience store, and/or the fourth convenience store, e.g., since they are all associated with the same convenience store chain. Thus, client A may have access to a view (e.g., within a browser or dedicated application) of any or all of the stores within the chain.

[0089] Client B, on the other hand, may correspond to a manager of the first convenience store. Accordingly, the manager may only be able to (or care to) view data associated with or control devices managed by the gateway device A. Thus, client B may have access to a view of only the first convenience store (e.g., allowing client B to view any or all of the premise devices or associated data of the first convenience store), but may be restricted from viewing any data for other convenience stores.

[0090] As another possibility, client C may be a user associated with only a particular device (or set of devices) from a particular store. For example, client C may correspond to a diesel fuel company that delivers diesel fuel to the first convenience store when the diesel fuel (e.g., of individual fuel tanks or overall) at the first convenience store is below a certain level. Thus, client C may only have a view corresponding to premise devices that are associated with that user (e.g., the job of that user).

[0091] Client D may correspond to a financial analyst employee of the convenience store chain. Accordingly, the financial analyst may only receive reports of the sales for each of the different convenience stores, but may not receive other information that is not relevant to financials (e.g., the temperature of the refrigerators in the various stores). For example, the financial analyst may receive sales data for each of the POS devices at the different stores, current gas prices, etc., but may not receive information unrelated to financial data of the convenience stores. Thus, client D may have a view for this restricted set of data for each store and/or all of the stores (e.g., within a same view).

[0092] Thus, in one embodiment, a plurality of gateway devices may be associated with a same entity (e.g., a merchant) and various clients may access data of that entity. In another embodiment, the different gateway devices may not be associated with one another or a common entity. For example, gateway device A may correspond to a user's home, and client A may correspond to the person owning the home. Gateway device B may correspond to a convenience store and client B may correspond to an employee of the convenience store. Gateway device C may correspond to a department store, and client C may correspond to a manager of the department store. Gateway device D may correspond to an exhibition hall, and client D may correspond to a scheduler or manager of the exhibition hall. In these examples, each of the clients may not be aware of the gateway devices other than the one that is associated with them. Thus, the cloud server may manage a plurality of distinct clients and gateway devices that are not inter-related.

[0093] In further embodiments, clients A and B may each be associated with gateway devices A and B and clients B and C may each be associated with gateway devices C and D. Any subset of clients may be associated with any subset of gateway devices, as desired. Thus, the server may be configured to maintain any desired relationship between a plurality of gateway devices and clients.

FIGS. 4A and 4B—Exemplary Block Diagram of Gateway Device

[0094] FIGS. 4A and 4B illustrate exemplary block diagrams of the gateway device, according to various embodiments. As shown in FIG. 4A, the gateway device may include an integrated broadband modem or other circuitry for establishing a connection to the Internet or another network (e.g., a DSL modem, cable modem, and/or fiber modem, among other possibilities). The gateway device may also include a router for providing a local area network (e.g., a wired and/or wireless LAN) to various devices. Alternatively, the gateway device may be connected to a modem and/or router, e.g., when these are not integrated into the gateway device.

[0095] The gateway device may also provide a firewall between the external connection (e.g., the Internet) and the resident portion of applications. Additionally, the gateway device may provide a firewall between the resident portion of applications and other applications, or connected devices. The firewall may be specially adapted to protect credit card information that is transmitted via the gateway device (e.g., from POS devices coupled to the gateway device).

[0096] The gateway device may further include logic for performing the functionality described herein. For example, the gateway device may include one or more field programmable gate arrays (FPGAs), integrated circuits (e.g., ASICs), and processors and memory to perform the functionality. In some embodiments, the processor and memory may be used to store and execute applications provided by the server.

[0097] Finally, the gateway device may include a device interface for coupling to various devices, e.g., managed by the gateway device. For example, the interface may include various wired connections for coupling with the premise devices (e.g., USB, Rs232, Rs485, etc., LONtalk, Ethernet ports, etc.). The interface may also couple to one or more switches, hubs, external routers, or external networks (wireless or wired, as desired) to couple to (and potentially mange) further devices. For example, as shown, the interface may include an Ethernet connection to a device (e.g., a switch) that connects to both a WiFi network and another device. The interface may also include video ports for coupling to a display for the gateway device, although a display may not be necessary.

[0098] FIG. 4B illustrates a different block diagram of the gateway device. In the embodiment of FIG. 4B, the gateway device may include communication logic and applications logic, and possibly secure applications logic. The communication logic may include any processors and memories and/or circuitry for communicating with the Internet, communicating with the premise devices at the location, providing routing functionality, providing modem functionality, etc. For example, the communication logic may include routing logic, wireless communication logic, wired communication logic, broadband modem logic, firewall logic, etc. In some embodiments, the communication logic may include a processor and memory, and the memory may store program instructions that are executable to perform the routing functionality or other communication functionality.

[0099] The application logic may include a processor and memory, and the memory may store applications (e.g., provided by the applications server) that may be executed by the processor. The application logic may include other circuitry, as desired.

[0100] The gateway device may also include secure logic, which may be configured to store and handle any information that requires security, such as isolation from the application logic where other applications are executed. For example, the secure logic may include a different processor and memory than the applications logic, and the memory may store applications that handle information (e.g., credit card information) in a secure manner.

[0101] In some embodiments, the secure logic may use a dedicated portion of the communication logic for receiving the secure information. For example, the gateway device may have a dedicated port (or wireless network) for receiving credit card information, which is isolated from the rest of the communication circuitry (e.g., the communication logic may have two isolated portions, one for the secure logic and one for the application logic or other functionality of the gateway device). Accordingly, the credit card information may be provided to the secure logic without any applications of the application logic being able to access the credit card information. This separation of the secure logic may allow the gateway device to be certified to perform credit card transactions.

[0102] In some embodiments, one or more of the logics of FIG. 4B may be separate integrated circuits or devices that are comprised in the same housing (of the gateway device). Thus, in one embodiment, the secure logic may be a separate integrated circuit within the gateway device, e.g., to ensure isolation from the other applications executing on the gateway device. Alternatively, the applications logic and the secure

logic may be implemented by the same circuitry and the secure logic may be separated from the application logic via software. In further embodiments, one or more of the logics may be separated into different housings (e.g., the secure logic may be a separate device from the gateway device).

[0103] Thus, FIGS. **4**A and **4**B illustrates an exemplary block diagram of the gateway device. Note however that other variations and embodiments are envisioned. For example, in further embodiments, the gateway device may perform further functionality, such as providing a femtocell for local cell phones, providing DVR or audio/visual functionality for a user (e.g., in a user's all-in-one device), providing a security system, controlling HVAC at the location, etc.

FIG. **5**—Developing an Application for a Gateway Device

[0104] FIG. **5** is a flowchart diagram illustrating one embodiment of a method for developing an application for a gateway device. The method shown in FIG. **5** may be used in conjunction with any of the computer systems or devices shown in the above Figures, among other devices. In various embodiments, some of the method elements shown may be performed concurrently, performed in a different order than shown, or omitted. Additional method elements may also be performed as desired. As shown, this method may operate as follows.

[0105] In **502**, a software development kit (SDK) and/or application programming interface (API) may be downloaded, and, in **504**, the SDK and/or API may be used to develop an application. For example, a user or developer that wishes to develop an application for installation on a gateway device (and/or the management server or other devices) may download the SDK and/or the API in order to develop the application. In one embodiment, the developer may need to register with the server (or entity managing the server) in order to receive the SDK or API. For example, the developer may need to pay a licensing or developer's fee, sign a non disclosure agreement, sign a terms of user policy, etc. in order to receive the SDK or API.

[0106] The SDK and/or API may allow the developer to develop an application that is compatible with the gateway device, the management server, and/or other devices that may execute portions of the application. In one embodiment, the SDK or API may be downloaded as executable program instructions (e.g., when downloaded as a development environment) and/or as documentation. For example, the SDK may include a compiler and/or simulator for compiling and testing the application, e.g., as it is developed. For example, the simulator may allow the developer to test execution on a simulated gateway device, simulated management server, simulated other device, or any combination (e.g., in order to test distributed execution of the application).

[0107] Additionally, the SDK may include demo or template applications (or they may be downloaded separately) that a user may modify to create further applications or may simply study for education purposes.

[0108] In **506**, after the application has been developed, the application may be submitted to an application server (or "server"). For example, the developer may upload the application (e.g., in a compiled form and/or in source code form) to the application server. An approval process may be required before the application is available for installation on gateway devices, the management server, other devices, etc. For example, an administrator of the application server (or other authorized user) may review the source code, test execution of the application, etc. in order to ensure that the application behaves as it is intended and/or will not cause any issues for users installing the application (e.g., on gateway devices, the management server, or other devices). The testing may occur on each corresponding device that the application is intended to execute on.

[0109] This review may be particularly important since many of the gateway devices may handle sensitive information, such as credit card information. Accordingly, it may be important to ensure that the submitted application does not present any security issues for such sensitive information. However, in some embodiments, each application (or specifically, each third party application) may be "sand boxed" or otherwise secured such that the application is not able to interact with other data or information of the gateway device, management server, etc. Alternatively, any sensitive applications or information may be sand boxed away or firewalled from other applications (e.g., third party applications) to ensure the security of the sensitive information.

[0110] In **508**, the application may be installed on one or more gateway devices. As described further below, users may browse the applications stored on the application server and select ones for installation on the gateway device (or other devices). Alternatively, applications on the application server may be automatically determined as described herein. **508** may be performed in the manner described below.

[0111] FIG. **6** illustrates an exemplary use of APIs for a developed application. In the illustration of FIG. **6**, the application is a tank application that is configured to determine the fuel level in the fuel tank and present the fuel level information in a user interface. In the embodiment of FIG. **6**, the tank application is executed in a distributed fashion, by both the gateway device (on the capture side) and the management server (on the presentation side).

[0112] As shown, the tank application, executing on the gateway device, may receive information from the fuel tank using a communication protocol supported by the fuel tank device (e.g., the device monitoring the fuel level in the fuel tank). The tank application is also configured to provide the reported information for storage, e.g., in cloud storage on the Internet. In order to perform this functionality, the tank application interfaces with the gateway device using an external API, e.g., which may be provided in the SDK. For example, the tank application may use a set of library functions to communicate and retrieve information from the fuel tank. Similarly, the tank application may use a set of library functions to provide the reported data for storage. Accordingly, the gateway device may be configured to receive the reported information from the tank application and provide that information for storage via an internal API. The internal API, unlike the external API, may not be exposed or known to developers. Instead, the internal API may be the set of functions that the gateway device uses to perform the storage actions requested by the tank application. Thus, the capture portion of the tank application may receive information from the fuel tank and provide information for storage using an external API. As shown, after provision to the cloud storage by the gateway device (e.g., using the internal API), time series data (e.g., representing the fuel levels of the fuel tank over time), device status, and net status information may be stored in the cloud storage, among other possibilities.

[0113] On the presentation side, a second portion of the tank application may execute on the management server. This portion of the tank application may use a presentation exter-

nal API to request information stored in the cloud storage. In response, the management server may use an internal API to retrieve the desired information. The tank application may also use the external API to display the retrieved information in a user interface provided by the management server. For example, the tank application may use the external API to generate a graph to provide the time series data of the fuel tank within a user interface framework provided by the management server (e.g., on a web page hosted by the management server). Thus, FIG. **6** illustrates an exemplary embodiment of a distributed application using internal and external APIs. Note that the fuel tank application is exemplary only and further types of applications, configurations, uses of APIs, etc. are envisioned. Additionally, further APIs and application portions may be used to display information on other devices (e.g., devices of users associated with the gateway device or location).

FIG. **7**—Storing and Providing a Plurality of Applications

[0114] FIG. **7** is a flowchart diagram illustrating one embodiment of a method for storing and providing a plurality of applications to gateway devices. The method shown in FIG. **7** may be used in conjunction with any of the computer systems or devices shown in the above Figures, among other devices. In various embodiments, some of the method elements shown may be performed concurrently, performed in a different order than shown, or omitted. Additional method elements may also be performed as desired. As shown, this method may operate as follows.

[0115] In **702**, a plurality of applications may be stored. The applications may be developed and/or received in the manner described above, with respect to FIG. **5**, although other embodiments are envisioned. The plurality of applications may be hosted by a server (e.g., in cloud storage accessible by the cloud server).

[0116] In **704**, one or more applications may be selected for installation. The one or more applications may be selected for installation manually or automatically as desired. For example, a user, such as an administrator of a location (e.g., a merchant location, such as a convenience store or a home, among other possibilities) or a plurality of locations (e.g., a chain of stores of a merchant), may wish to install an application from the plurality of available applications. Accordingly, the user may access the application server to browse or search the plurality of applications. For example, in one embodiment, the application server may host a web site and the user may log on to the web site to choose from the plurality of applications for installation. Alternatively, or additionally, the user may use a locally installed program that is in communication with the application server to select one of the applications for installation. For example, the locally installed program may retrieve the list of applications available from the application server and provide them for selection on a display of a client used by the user. In one embodiment, the user may access the application server (via webpage, application, or other mechanism) to visit a "store" of applications, which may be referred to as an application store, app store, PACKAGE WAREHOUSE™, PACKAGE STORE™, etc.

[0117] The user may be able to search for or browse through the different applications. Additionally, each application may provide a description (which may be textual and/or graphical, e.g., including screen shots) of the functionality and requirements of the application. After selecting the desired applications, the user may then specify the desired devices for installation (which may include the gateway device, the server, and/or other devices, such as user devices). Alternatively, or additionally, the user may specify a desired device and then select from applications that are available for that device. Other implementations for selection of applications and target devices are envisioned. For example, the server may be configured to automatically choose the target devices based on the selected applications, the corresponding user or entity, the corresponding location of the gateway device, the gateway device itself, etc. As one specific example, a user's device may already execute a management application to manage the gateway device(s) and corresponding coupled devices and the application portion that is intended for the user's device may be provided as an update to the management application (e.g., as a modular addition to the management application).

[0118] As indicated above, the applications may be any of a variety of applications. For example, a first one or more of the applications may be associated with particular devices that may be managed by a gateway device. These type of applications may be, for example, for communicating with or controlling POS devices, electronic sign devices, sensors, liquid dispensing devices, or any type of device that may be coupled to the gateway device. A second one or more applications may be associated with presentation or management of data received from the gateway device (e.g., for individual ones of the devices managed by the gateway device, for groups of the devices managed by the gateway device, or for any type of data associated with the gateway device's location). A third one or more applications may be associated with management of a plurality of different gateway devices, e.g., for aggregating or presenting the information gathered from a plurality of different locations. For example, applications of the third type may present financial data for all of the stores managed by a manager, fuel consumption of a plurality of different convenience stores, rate of device malfunction for liquid dispensing devices, etc. A fourth one or more applications may be associated with retrieving and managing the gateway devices (or devices managed by the gateway devices) on a user's device, such as on a mobile phone, desktop computer, laptop, netbook, tablet computer, etc.

[0119] Note that there may be applications that overlap with any of the above types of applications described herein. Additionally, various ones of the applications may be executed only by the gateway device, the server, or user's devices, and various others of the applications may be executed by more than one of the above. For example, a fuel tank monitoring application may be designed for execution by the gateway device (e.g., to receive reported information from one or more fuel tank devices and provide control messages to the one or more fuel tank devices), by the server (e.g., to receive the reported information from the gateway device, provide control messages to the gateway device for controlling the fuel tank devices, provide the reported information to client devices, and/or receive control information from client devices), and by user devices (e.g., to provide an interface for a user to view the reported information and control the fuel tank devices). However, in simpler distributed execution embodiments, an application may only be executed by the gateway device (e.g., for communication purposes) and by the server (e.g., for providing an interface to users) or simply just by the gateway device or server, as desired. The execution

requirement or possibilities may be listed in the description of each application, for the user's consideration.

[0120] Thus, the user may browse through these different applications and select them for installation on the appropriate systems. Following the exemplary system of FIG. **1**A, the user may select a fuel tank device application for the fuel tanks, a liquid dispensing device application for the liquid dispensing device, a refrigeration device application for the refrigeration device, a POS device application for the POS device(s), etc. Following the exemplary system of FIG. **1**B, the user may select a TV application for the TV, a refrigerator application for the refrigerator, a fish tank application for the fish tank, a solar panel application for the solar panel(s), etc.

[0121] The user may also select other applications for installation that are not specifically associated with the devices managed by the gateway device. For example, the user may select an application for installation on one or more devices that are associated with user(s) associated with the location (e.g., employees, managers, owners, etc.) which allow for monitoring and/or control of the devices at the location. The user may also select an application for managing notifications of the various devices, although such features may already be present in the individual applications themselves. The user may also select an application for aggregating data from a plurality of different locations and/or gateway devices.

[0122] The user (or in embodiments below, the method) may be able to specify installation for a plurality of locations or devices. For example, an owner of a convenience store chain may select an application and specify installation of the application on a plurality of gateway devices or other devices. The owner may also specify that the application be associated with a plurality of locations on the management server (e.g., allowing the application to be added to the user interface for a specified plurality of convenience stores in his chain). In one embodiment, the user may specify a default application configuration that may be used for installation on a plurality of different gateway devices (e.g., as described below). Thus, the user may be able to specify mass installation of one or more applications for a plurality of different locations or gateway devices.

[0123] Thus, in one embodiment, a user may select applications for installation by browsing or searching through a plurality of applications in a user interface (e.g., a GUI) and selecting the applications for installation. However, it should be noted that the applications may also be selected in an automatic fashion. Specific embodiments describing automatic installation of applications are provided below in describing FIGS. **8** and **9**.

[0124] In some embodiments, payment may be required for installation of the applications, e.g., a one time purchase fee and/or a subscription fee, among other possibilities. Accordingly, financial information may be required (e.g., pre-stored or provided by the user) in order to install the applications. In one embodiment, the payment may be reduced for bulk purchases. For example, a user may receive a discounted price per application when installing multiple instances of the application (rather than paying the same higher price for each installation of the application). For example, the user may be able to enter a discount coupon or code in order to receive a negotiated price. In some embodiments, the user may use a user interface to request a negotiated price and may receive a negotiated price based on the request (e.g., in response to user review). For example, a user may specify an amount of a purchase (e.g., a number of applications for installation, a number of products, a number of contracts, etc.) and may request a negotiated price (e.g., by specifying a desired price or simply requesting a lower price). Accordingly, the seller (e.g., the application developer) may determine a negotiated price for the user's request. Accordingly, the negotiated price may be provided to the user and the user may make a purchase at the negotiated price. In another embodiment, the buyer and seller may negotiate over a different medium (e.g., telephone, email, etc.) and a price may be determined. The seller may establish a price for the user, and that price may be associated with the user on the cloud server. Alternatively, the seller may provide a discount or coupon code for the user to make the purchase. Thus, in some embodiments, prices may be negotiated (e.g., for bulk purchases/installations of application (s)).

[0125] In **706**, the one or more applications may be installed based on the selection in **704**. For example, the server may receive a request to install the selected applications, e.g., from a client of the user. The server may determine (e.g., automatically) the proper location for installation of each of the applications. In some embodiments, the location of the installation may be specified by the application itself, or may be specified by the user. For example, if the application is to be installed on the server, the server may install the selected application on the server (e.g., as part of an interface associated with the user) without requiring any specification of installation location. Similarly, if the application is to be installed on the gateway device, the server may provide the application to the gateway device for installation. However, where there are multiple gateway devices, the request from the user may specify the gateway devices for application installation. Additionally, if the application is also capable of being installed on user devices (e.g., mobile phones), the user request may specify those devices.

[0126] Note that where the application is distributed across a plurality of different devices, the server may provide the appropriate portions to each of the different devices for installation. For example, for an application that is executed by the server and the gateway device, the server may install its portion and provide the gateway device portion to the gateway device for installation. Thus, the selected applications may be installed at their proper locations.

[0127] In **708**, the one or more applications may be executed by the corresponding systems to perform the functionality specified by the one or more applications. For example, the applications may execute in the manner describe above, with respect to FIG. **6**.

FIG. **8**—Automatically Installing a Plurality of Applications

[0128] FIG. **8** is a flowchart diagram illustrating one embodiment of a method for automatically installing a plurality of applications on a gateway device based on detected devices. The method shown in FIG. **8** may be used in conjunction with any of the computer systems or devices shown in the above Figures, among other devices. In various embodiments, some of the method elements shown may be performed concurrently, performed in a different order than shown, or omitted. Additional method elements may also be performed as desired. As shown, this method may operate as follows.

[0129] In **802**, a gateway device may be identified or detected. For example, in one embodiment, upon being installed at a location and having an operating network con-

nection, the gateway device may automatically attempt to communicate with the server. The gateway device may provide message including a unique identifier identifying the gateway device to the server. Additionally, in some embodiments, the gateway device may report a location of the gateway device (e.g., using GPS circuitry or other methods for determining location). Similarly, the server may use various techniques for determining the location of the gateway device, as desired.

[0130] Alternatively, or additionally, a user may register the gateway device with the server. The user may optionally provide an IP address or other address to allow the server to communicate with the gateway device. Thus, in **802**, the gateway device may be detected or otherwise identified to the server.

[0131] In **804**, a plurality of devices associated with the gateway device may be identified. For example, the gateway device may be configured to automatically discover any premise devices on the same network as the gateway device. In various embodiments, the gateway device may actively seek out the other devices or the other devices may be configured to automatically notify the gateway device of their presence. In one embodiment, the gateway device may use various discovery protocols for performing the discovery, e.g., Bonjour, Plug-N-Play, ARP, or other protocols. Alternatively, or additionally, the gateway device may attempt to communicate with each IP address of the network at the location to determine if there is a device on the network at that IP address. For example, temperature sensors coupled to the refrigerator, a liquid dispensing device, a POS device, a fuel tank device, etc. may automatically be discovered by the gateway device. In automatic discovery, the devices are discovered without user input manually specifying the network address and the identification of the devices. During discovery, the type of the device, the model of the device, the serial number of the device, the location of the device, etc. may be determined.

[0132] In alternate embodiments, a user may assign or provide identification of some of the premise devices on the local area network. For example, in one embodiment, the gateway device may discover a premise device on the network, but may be unable to identify it. Accordingly, a user may provide input indicating the type of device, model of device, and/or other information used to identify the premise device. The user may also be able to specify undiscovered premise devices or premise devices that will be installed at the location at a later time.

[0133] In **806**, a plurality of applications may be automatically selected based on the identified devices. For example, information specifying the plurality of identified premise devices (e.g., identification information) may be provided from the gateway device to the server. The identification information may include the type of the premise device, the model of the premise device, and/or a device identification number (e.g., the serial number of the premise device), among other types of identification information. In general, the identification information may be any information that is usable (e.g., by the server) to appropriately determine an application for managing the premise device. For example, the gateway device may determine that a premise device has a particular device identification number (e.g., by attempting to communicate with the premise device using different protocols, or via other means) and provide that device identification to the server. In one embodiment, the device identification number may be a numbering system used by the server and not specifically associated with the device (e.g., the server may have a database of device identification numbers, and these numbers may not be specifically associated with the device, as is a serial number). In another embodiment, the type of the device or manufacturer of the device may be identified from a generic identifier (e.g., using the Organizationally Unique Identifier" (OUI) portion of a MAC address). For example, the OUI may specify a manufacturer of a device. The server may then use the identification of the premise devices in order to automatically select a plurality of applications for installation. More specifically, in one embodiment, the server may use each identification information to select a corresponding application, if available.

[0134] For example, the identification information may specify a fuel tank device having a specific manufacturer (e.g., using the OUI) and/or model (e.g., using the serial number). The server may then determine if there are any applications associated with fuel tank devices that are compatible with the manufacturer and/or model of the fuel tank device (e.g., using a database). For example, there may be an application that supports all fuel tank devices made by the manufacturer or there may be an application for each model of the fuel tank device. Similar selection procedures may be performed for each of the different identified devices.

[0135] In **808**, the plurality of applications may be automatically installed. The applications may be installed similar to **706** above. Note that at least a subset of the applications may be at least partially installed on the gateway device. In some embodiments, at least a portion of all of the applications may be installed on the gateway device. Similar to above, various ones of the devices may be designed for distributed execution among the gateway device, the server, and/or other devices, as desired.

FIG. **9**—Automatically Installing a Plurality of Applications

[0136] FIG. **9** is a flowchart diagram illustrating one embodiment of a method for automatically installing a plurality of applications on a gateway device based on a application configuration. The method shown in FIG. **9** may be used in conjunction with any of the computer systems or devices shown in the above Figures, among other devices. In various embodiments, some of the method elements shown may be performed concurrently, performed in a different order than shown, or omitted. Additional method elements may also be performed as desired. As shown, this method may operate as follows.

[0137] In **902**, a gateway device may be identified or detected, similar to **802** above. In some embodiments, the gateway device may be associated with a merchant or entity having a plurality of associated gateway devices. For example, the gateway device may be associated with a fist convenience store of a convenience store chain that includes other convenience stores, each having one or more respective gateway devices. The association of the gateway device with the merchant or entity may be performed in response to user input (e.g., specifying that it is associated with the entity) or automatically, as desired. For example, the server may be configured to automatically identify that the gateway device is associated with the entity using the identification information provided by the gateway device, the location of the gateway device, the network that the gateway device is on (e.g., a network associated with the entity), the IP address of the gateway device, or via any other means that would associated

the gateway device with the entity. However, in alternate embodiments, the gateway device may not be associated with a merchant or entity having one or more other associated gateway devices.

[0138] In **904**, an application configuration may be automatically determined for the gateway device. The application configuration may specify a plurality of applications for installation for operation with the gateway device. Similar to above, the applications may be installed on the gateway device and/or the server.

[0139] In one embodiment, a default application configuration may be determined for the gateway device. For example, the default program configuration may include a management application that may be installed and executed by the gateway device and/or the server to allow users to interact with the gateway device. However, such functionality may already be installed and may not be included in a default program configuration provided after installation of the gateway device at the gateway device location. Other applications may be included in the default application configuration.

[0140] In some embodiments, the default application configuration may be associated with a type of location, e.g., but may be agnostic with respect to the specific merchant or entity associated with the gateway device. For example, there may be a default application configuration for convenience stores, for department stores, for homes, etc. Accordingly, depending on the type of location of the gateway device, a different default application configuration may be used.

[0141] Additionally, or alternatively, the application configuration may be based on the entity with which the gateway device is associated (assuming that it is associated with such an entity). In some embodiments, the default application configuration may be specific to the merchant or entity associated with the gateway device. For example, a convenience store chain may have a default application configuration for each gateway device installed in each convenience store of the chain. Alternatively, or additionally, the application configuration may be based on the plurality of applications installed on other gateway devices that are also associated with the entity, e.g., where each location generally includes the same types of devices and/or management prefers the same types of management or presentation applications for each location. In one embodiment, an administrator or other authorized user of the entity may specify the application configuration to be used for each new gateway device associated with the entity (e.g., thereby specifying a default application configuration). However, as indicated above, the application configuration may also be generated automatically by the server (e.g., based on gateway devices in similar locations or having similar attached devices) without receiving such input.

[0142] Thus, an application configuration may be determined, e.g., automatically, for the detected gateway device.

[0143] In **906**, the plurality of applications may be automatically installed. The applications may be installed similar to **706** above. Note that at least a subset of the applications may be at least partially installed on the gateway device. In some embodiments, at least a portion of all of the applications may be installed on the gateway device.

FIG. **10**—Notification of Users

[0144] FIG. **10** is a flowchart diagram illustrating one embodiment of a method for notifying users of reported information of managed devices. The method shown in FIG. **10** may be used in conjunction with any of the computer systems or devices shown in the above Figures, among other devices. In various embodiments, some of the method elements shown may be performed concurrently, performed in a different order than shown, or omitted. Additional method elements may also be performed as desired. As shown, this method may operate as follows.

[0145] In **1002**, reported information may be received, e.g., by the server. For example, the reported information may be received from a first gateway device for a specific premise device managed by the first gateway device. The reported information may include information gathered by the specific premise device (e.g., concerning a condition of the premise device, a condition of the environment near the premise device, any measurements performed by the premise device, etc.). For example, where the premise device is a fuel tank device, the reported information may indicate the current fuel level of the corresponding fuel tank. Alternatively, or additionally, the reported information may indicate an alarm condition, e.g., following the example above, where the fuel level is below a threshold level of fuel. Note that instead of receiving the alarm condition, the server may be configured to analyze the reported information to determine if there is an alarm condition. Such functionality may be implemented as part of an application, e.g., for the specific premise device or otherwise.

[0146] In **1004**, one or more devices may be determined for receiving the reported information. The determination may be performed manually (where a user specifies which other users receives the reported information) and/or automatically, as desired. In preferred embodiments, the one or more devices may be determined automatically. For example, the server may store a plurality of rules for providing reported information to users and/or devices (e.g., user devices). The set of rules may specify any of the conditions described herein.

[0147] In one embodiment, there may be a set of users that should be notified of any reported information (or more particularly, any reported information when there is an alarm condition). Such users may be managers or owners of the location (e.g., the convenience store) or users associated with the location (e.g., managers or owners of a chain of convenience stores, including the store having the specific device). Each user may have a set of devices associated with the user. For example, a manager may be associated with a particular cell phone, tablet computer, desktop computer, and/or other device. Accordingly, the rules may indicate that all or some subset of those devices should be notified based on the reported information. These rules may indicate a general preference (e.g., manager A should only receive reported information via email whereas manager B should receive reported information via email and via text message), may be specific to the priority of reported information (e.g., high priority alarms, such as a break in, should be sent to all devices associated with the manager, and/or low priority alarms should be sent to the default device), may be specific to the type of reported information or managed device (e.g., financial information should be sent via email whereas gas pricing information should be sent via text message), etc. The selection of the devices may be selected based on other conditions as well, e.g., including, but not limited to a current time, a schedule of the user (e.g., a work schedule, vacation schedule, etc.), a location of the user (or associated devices), the gateway device providing the reported information, etc. Note that these rules may be fully customizable for each different user, e.g., where user A can set all of the above rules

(among other possibilities) for himself and user B can set the above rules in a different manner than user A for himself.

[0148] Although the above descriptions relate to devices that are associated with users, there may be particular devices (e.g., not associated with users) that should also be notified or provided reported information. For example, a computer operated at the location of the first gateway device may also be notified of reported information for the devices at that location, e.g., at all times.

[0149] Other users or devices may only be notified conditionally. For example, the users or devices may be notified based on the current time. In one embodiment, the rules may specify time periods to receive reported information (or conversely, time periods to not receive reported information, or both). Accordingly, **904** may be performed by comparing a current time against these specified time periods. As one specific example, employees may only be notified when they are on shift. Similarly, a night manager may only be notified at night and a day manager may only be notified during the day. In one embodiment, rather than always being notified, the computer at the location of the first gateway device may only be notified when the location is open (e.g., during business hours of a convenience store) or when other users or employees are present.

[0150] Further, some users or devices may only be notified for particular managed devices. For example, a financial manager may only receive information associated with financials of the location (e.g., POS device transactions, gas prices, etc.) and not reported information of all premise devices. As another example, a fuel company or employee in charge of fuel refills may only receive reported information for fuel tank devices (e.g., of a single location or of multiple different locations) and may not receive reported information of other managed devices. Further, manufacturers of the various premise devices may be only notified for their devices (e.g., regardless of the location). Thus, the manufacturers of equipment be able to keep track of use, condition, etc. of installed premise devices (e.g., for developing use statistics, monitoring whether the devices have been properly maintained, ensuring proper warranty use, performing maintenance, etc.

[0151] Thus, in one embodiment, rules may specify that a particular user should receive information for a plurality of devices, and **1004** may be performed by determining whether reported information is received for one of the plurality of devices and, if so, determining one or more devices associated with the user for delivery of the reported information. Alternatively, or additionally, the rules may simply specify a type of device that the user should receive information for and **1004** may be performed by determining whether the reported information is received from a device that falls within the specified type and, if so, determining one or more devices associated with the user for delivery of the reported information.

[0152] Additionally, not only may the reported information be provided to specific users or devices based on the rules described above, these users or devices may only receive the information under certain conditions (which may also be specified by rules). For example, the fuel level of the fuel tank may only be provided to the fuel tank employee when the fuel level is below a certain threshold, since the employee may not be interested when the tank does not need to be refilled. Other conditions may include the location of the user (or the device associated with the user, e.g., when the user is out of the country or on vacation), whether the reported information has already been sent to a threshold number of users (e.g., where a user is only to be contacted as a last resort), etc.

[0153] Note that the above features may be implemented as core functionality provided by the server and/or gateway device or may be integrated within applications that can be installed. For example, there may be an application to manage notifications (e.g., to manage the rules described above) of the various devices to users. This application may be integrated with or separate from an application that stores employee work schedules or shift information (e.g., where the rules are based on schedules or specified time periods of the user(s)). Alternatively, or additionally, the fuel tank application (as an example) may include notification features for notifying employees or other entities (e.g., fuel companies) based on reported information of the fuel tank devices. Other types of implementations and applications are envisioned.

[0154] In **1006**, one or more messages may be provided to the one or more devices based on **1004**. The messages may be provided based on rules as well. For example, the rules may specify provision of messages depending on the types of devices. For example, SMS messages may be sent to cell phones, emails may be sent to devices having email addresses, IP messages may be sent to devices having an IP address, etc. Alternatively, or additionally, the rules may specify the default or desired types of messages for each device (e.g., as specified by users).

[0155] It should be noted that in the method described above, the received reported information may be aggregated to include reported information of a plurality of devices. The method may be performed for the aggregated reported information and/or may be performed for each specific device's reported information, as desired. Thus, it may be determined where the whole of the reported information is sent or it may be determined where each individual device's reported information is sent on a per device basis.

FIG. **11**—Determining and Providing a User Interface for Display

[0156] FIG. **11** is a flowchart diagram illustrating one embodiment of a method for determining and providing advertising for display. The method shown in FIG. **11** may be used in conjunction with any of the computer systems or devices shown in the above Figures, among other devices. In various embodiments, some of the method elements shown may be performed concurrently, performed in a different order than shown, or omitted. Additional method elements may also be performed as desired. As shown, this method may operate as follows.

[0157] In **1102**, a user may provide user identification, e.g., to the cloud server. For example, the user may provide the user identification (e.g., user name and password) to a web site hosted by the cloud server. Alternatively, the user identification may be provided through an application executing on a device (e.g., such as on a smart phone, tablet computer, netbook, computer, etc.).

[0158] In **1104**, the cloud server may identify the user based on the user identification. Additionally, the cloud server (or the application) may determine a user interface associated with the user. For example, the cloud server may include a database of users and associated user interface information. Accordingly, the cloud server may use the user identification to determine the interface information associated with the user. The user interface information may indicate an interface layout and/or which applications are installed for the user.

[0159] For example, the database may indicate that the user interface of a manager of a convenience store includes all the interfaces of applications associated with the convenience store (e.g., that the interface should include all the installed applications). Thus, for example, when the manager of the convenience store logs in, the manager is able to view all of the reported information (or any information of the installed applications) within his associated user interface.

[0160] As another example, an owner of a convenience store chain may log in and the determined user interface may include interfaces for a plurality of convenience stores in the convenience store chain. In another example, a user associated with a refueling company may log in and the user interface may allow the user to view only the fuel tank information of convenience stores that use the refueling company for refueling their fuel tanks.

[0161] In one embodiment, the determination of the user interface may be associated with the entity that owns the corresponding location. For example, a fast food restaurant chain may have a user interface that is the same for all users associated with the fast food restaurant. For example, all restaurant managers may use the same interface for viewing reported information for their respective restaurant. In some embodiments, the user interface may also (or alternatively) be determined based on the type of user. For example, a manager may have a first interface while an employee may have a second interface, even when each user is associated with the same location or chain. Thus, similar to notifications described above, different users may have different interfaces depending on the type of the user, the types of devices associated with the user, the entity associated with the user, the security level of the user, etc.

[0162] In some embodiments, each user may be able to customize the user interface. For example, a user may be able to rearrange the location of various interfaces and, accordingly, the layout of the user interface may stored, e.g., in a database. Thus, each user may have a customized interface that is determined based on the user identification. Each user may also be able to install applications that are specific to the user. For example, each user may be able to install various social networking applications, games, desired user applications, etc., which may be outside of the purview of the location. Thus, the user interface determined for the user may include elements associated with the location(s) to which the user is associated (e.g., applications installed for a location, such as a convenience store) as well as elements that are associated with the user (e.g., separate applications or interfaces that the user has selected for that user's profile).

[0163] In one embodiment, the user interface may include various social networking applications or features, such as allowing the user to post status messages, interact with other users (e.g., that are also associated with the location, a chain or entity associated with the location, etc.). As described below, the user interface may include a ticketing system or forum that allows the user to post and/or respond to troubleshooting items (e.g., associated with the location), discuss various topics, etc.

[0164] Accordingly, in **1106**, the determined user interface may be presented to the user, e.g., on the website.

FIG. **12**—Determining and Providing Advertising for Display

[0165] FIG. **12** is a flowchart diagram illustrating one embodiment of a method for determining and providing advertising for display. The method shown in FIG. **12** may be used in conjunction with any of the computer systems or devices shown in the above Figures, among other devices. In various embodiments, some of the method elements shown may be performed concurrently, performed in a different order than shown, or omitted. Additional method elements may also be performed as desired. As shown, this method may operate as follows.

[0166] In **1202**, information regarding a location may be determined, and in **1204**, that information may be used to determine advertising. In each section below, a respective type of information and corresponding uses for that information is described.

[0167] In one embodiment, the information may specify the devices (or types of devices) that are operated at the location. For example, the devices and/or types of devices may be determined by analyzing stored information regarding the location, such as information gathered via a gateway device installed at the location, similar to embodiments described above, or information provided by users, among other possibilities. For example, the information may specify that a convenience store includes or operates electronic signs, fuel tanks, liquid dispensing devices, cooling devices, etc.

[0168] This information may be used to determine advertising in a variety of manners. For example, in one embodiment, the method may determine other devices that may be desirable or compatible with the location based on the premise devices that are already present at the location. As one specific example, the location may include an ice machine that generates ice, but may not have an ice bagging device. Accordingly, the method may determine an advertisement that advertises an ice bagging device that is compatible with the ice machine. As another example, the location may include one or more cooling machines or freezers that store frozen food, and the method may determine that a microwave that could be used to thaw or reheat the food by customers may be desirable. Accordingly, an advertisement for a reheating device (such as a microwave) may be determined based on the cooling devices present at the location. In another instance, the location may include a home theater system, and the method may determine an advertisement for a DVR that may be compatible with the home theater system. Thus, the method may determine advertisements for other devices (e.g., compatible devices) to those already present at the location.

[0169] In further embodiments, the method may analyze the devices to determine what devices are not present that may be desirable (rather than determining other devices based on the devices already present, as in the examples above). For example, the method may store a list of premise devices that are commonly present in various types of locations (e.g., convenience stores, department stores, homes, etc.) and compare the list of current premise devices to that list to determine whether to advertise a device that is not present. For example, a convenience store may not have an electronic sign to advertise gas prices and the method may identify that the electronic sign is not present at the convenience store and determine an advertisement for an electronic sign. In one embodiment, the list of possible devices may be determined automatically. For example, the server may analyze other locations (e.g., of the same type, within the same area, associated by the same chain, etc.) and determine which premise devices are not present at the location being analyzed in relation to the other locations. Accordingly, in one embodiment, devices that are

usually present in such locations, but are not present at the current stores may be suggested in an advertisement.

[0170] The method may also determine products or parts (e.g., replacement parts) associated with the determined premise devices. For example, the method may determine advertisements for coffee filters, coffee dispensers, coffee cups, etc. associated with a coffee device; fry baskets, frying oil, etc. for frying machines; light bulbs for lighting devices; chlorine tablets for swimming pools; saline or cleaning solutions for fish tanks; air filters for air conditioning units; coolant for cooling units; etc. Thus, advertisements for products or parts associated with the identified devices may be determined.

[0171] In some embodiments, the method may also determine advertisements for services that may be associated with the determined premise devices. For example, the method may determine advertisements for performing fuel delivery based on a fuel tank device; maintenance services for maintaining various devices, such as cooling devices, liquid dispensing devices, etc.; food delivery services for providing food items associated with devices (e.g., for delivering coffee for a coffee device); etc. Thus, the advertisements may include advertisements for services associated with premise devices at the location.

[0172] The information may also specify reported information of various premise devices operated at the location. For example, the information may specify a level of refrigerant for a cooling device, an amount of syrup for a liquid dispensing device, a level of ice in an ice device, a level of fuel for fuel tanks, etc. Accordingly, the method may determine advertisements based on the reported information of the device. For example, if the fuel tanks are low (e.g., less than a threshold), the method may determine fuel advertisements. Similarly, if the syrup in the fountain drink machine is low, the method may determine advertisements for fountain drink syrups. In another example, if the coolant of an air conditioning unit is low, the method may determine coolant advertisements or advertisements for a repair service to service the air conditioning unit.

[0173] Various other types of reported information may also be gathered, such as the condition of premise devices (e.g., indicating whether a premise device is operating poorly, is in bad condition, or operating sufficiently, etc.), age of the premise devices, etc. For example, if an air conditioning unit has not had its air filters replaced (or otherwise is indicates that air quality is suffering), the method may determine advertisements for air filters or a maintenance service. Alternatively, for lower priced items, if the condition of the device is sufficiently low, the advertisements may suggest replacement devices or upgrades to the devices. For example, when a microwave is no longer functioning properly, advertisements for new microwaves may be determined. Thus, advertisements may be generated based on the reported information of devices.

[0174] The information may also specify a type of location, which may be usable to determine advertisements. For example, the information may specify that the location is a convenience store and advertisements (e.g., for products or services) that are appropriate for convenience stores may be determined. For example, the method may determine advertisements related to pest services (e.g., insects or other pests), cleaning services (e.g., for the parking lot), trash services, fuel delivery services, bulk food services, etc. Similarly, for different locations, such as a home, the advertisements may suggest maid services (e.g., for cleaning the house), lawn services, grocery services, repair services, etc. that are appropriate for a home. In addition to services, products or other items may be suggested. For example, for a home, various electronic devices or appliances may be suggested; for a convenience store, various food items, vending machines, etc. may be suggested; for a department store, various apparel items that are appropriate for that type of department store may be suggested; etc. Thus according to some embodiments, advertisements may be determined based on the type of the location.

[0175] The information may also specify inventory levels, e.g., of a convenience store. In one embodiment, the server may receive information (e.g., or otherwise store information) regarding the inventory of various items for purchase or use at the convenience store. For example, an employee or other entity may use an inventory management application to keep track of inventory supplies of the convenience store, which may be stored or otherwise accessed by the server. Accordingly, advertisements may be presented for items based on the inventory level. For example, if certain inventory items are low, advertisements for those items may be determined. As another example, the inventory may be used to suggest other products or items that the location may wish to sell. More specifically, the method may determine that the store is not selling a particular type of soft drink and then provide advertisements to buy the soft drink for resale.

[0176] The information may also specify a geographic location of the location, demographic information of the location, socioeconomic information of the location, etc. For example, the method may use the geographic location, demographic information, and/or socioeconomic information to determine services or products that may be popular in that location, and then determine advertisements for those services or products. As one example, based on the geographic location of convenience store (e.g., in Texas) and demographic information (e.g., rural population and migrant workers), the method may determine a food product that is popular or typical, such as tex-mex food (e.g., burritos) and determine an advertisement for that food (e.g., for stocking the convenience store with the advertised food). The geographic location could also be used to identify weather patterns and provide advertisements based on those weather patterns (e.g., air conditioning units for warmer areas, heating units for cooler areas, hail resistant roofs for areas prone to hailing, snow shovels for areas that receive a lot of snow, etc.). Socioeconomic information could be used to determine advertisements as well. For example, where the location is a home of a relatively wealthy person, the advertisements may concern high end electronic devices such as laptops or smart phones.

[0177] The information may specify the applications installed with respect to the location (e.g., on the server, on the gateway device of the application, on devices associated with the location, etc.). Accordingly, advertisements for other applications (or different application installation locations) may be determined based on the list of advertisements. For example, the method may determine advertisements for applications which are compatible with the devices operated at the location, applications for managing data associated with the devices, etc. The advertisements regarding applications may be performed in a similar manner to the installation of applications described above in FIGS. **7-9**.

[0178] Thus, any of various information regarding a location may be used to determine advertisements.

[0179] In **1206**, the advertising may be provided for display. The advertising may be provided for display in any number of locations via any number of devices. For example, the determined advertising may be provided for display at one or more computer systems at the location, in interface(s) provided by the server (e.g., within web pages used to display reported information or alerts), in interface(s) used to interact with the devices or view reported information (e.g., user devices which execute an application for interacting with the devices or viewing the reported information), within the application store (or package warehouse, such as when the applications are being selected or purchased), within an application interface (e.g., for in-application purchases, whether they are on the website or on a user's device), etc.

[0180] In some embodiments, the advertisements may be provided to specific devices or users based on the type of the determined advertisements. For example, advertisements for larger purchases may only be provided to users (e.g., devices associated with users) with the ability to authorize or make such purchases, whereas advertisements for smaller purchases may be provided to all users. In some embodiments, the advertisements may be provided based on the role of the user. For example, a user associated with fuel systems may receive fuel related advertisements (or have a preference towards those advertisements), a user associated with finance may receive finance related advertisements, a user associated with a particular location may only receive location specific advertisements, a user associated with a chain of locations may receive advertisements that can apply to one or a plurality of locations (e.g., bulk purchase advertisements or chain wide advertisements), etc. Thus, the advertisements may be targeted to individual users based on the types of advertisements.

[0181] Additionally, the advertisements may be provided to specific interfaces based on the type of the determined advertisements. For example, advertisements related to applications may be provided within the applications store (or application selection interface). As another example, advertisements related to fuel may be provided when a user is viewing reported information related to fuel devices. Advertisements for maintenance or repair services may be provided for particular types of premise devices when reported information of the premise device is being viewed (e.g., and when then condition indicates that a repair or maintenance should be performed). Thus, the advertisements may be automatically determined and provided based on the context of the current interface.

[0182] After the advertisements are provided, a user may be able to purchase the advertised products. As indicated above, the advertised products may be physical products, service agreements, or software products (e.g., further applications that may be installed on gateway device(s), user device(s), other devices, the interface provided by the management server, etc.).

[0183] Thus, FIG. **12** illustrates various embodiments of providing advertisements based on information regarding a location, such as a merchant location, although others are envisioned.

FIGS. **13**A-**15**C—Exemplary Screen Shots

[0184] FIGS. **13**A-**13**L, **14**A-**14**D, and **15**A-**15**C illustrate exemplary screen shots corresponding to embodiments of the methods described above. Additional features shown in these Figures are applicable to embodiments described above, and the specific implementation shown is not limiting; instead, any variation or appropriate modifications for other applications are envisioned.

[0185] FIG. **13**A illustrates an exemplary interface (e.g., a webpage) for monitoring a plurality of different convenience stores in a convenience store chain. As shown, the convenience store chain may be "Gas Plus". This interface illustrates a plurality of Alon Brands stores in or around Abilene, Tex. In some embodiments, each icon for each store may have a corresponding status, which may be illustrated using a color or in other methods. In FIG. **13**A, the status of each store is illustrated using a condition marker (black circle for critical, hashed circle for warning, white circle for ok).

[0186] As shown in the toolbar, the user (in this case, "administrator") may access various features, e.g., tools, help, feedback, logout, search functionality, etc.). The user may be able to add a new location by pressing the "+New Location" button or perform other tasks. The user may also be able to select one of the stores to view further information about that store. FIG. **13**B illustrates an alternate view where the user may view the various different locations in a list view. As shown, each different store may have a status icon, location, carrier (e.g., for broadband connection), serial number (e.g., of the gateway device), etc. FIG. **13**C illustrates a map view where the various locations can be displayed on a map. As also shown in the interface of FIG. **13**C, recent activity of the displayed locations may be displayed, e.g., where users can post messages concerning the various locations.

[0187] FIG. **13**D illustrates the resulting screen when the user selects "Gas Plus 045", which as shown in FIG. **13**A (dashed) is in critical condition. FIG. **13**D illustrates an exemplary status screen where the user can quickly view reported information for each of the devices managed at the selected store. As shown, various ones of the premise devices are in critical condition, including a freezer, an ice cream machine, the open deli, a mid temp refrigerator, a fountain drink dispenser, and the three fuel tanks (regular, super, and diesel). The status screen also indicates that slurpee machine **1** and one of the refrigerators is in a warning condition, and slurpee machine **2**, the coffee island, and the air conditioning units are in "OK" condition. The status screen further indicates that the coffee island and the slurpee machine **2** should be restocked. As also shown, the user may be able to view the floor plan (shown in FIGS. **13**E and **13**F), 24 hour temperature screens, configuration screens, etc.

[0188] FIG. **13**E illustrates a floor plan of the convenience store, with indications for the location of each of the premise devices inside of the convenience stores. In some embodiments, the floor plan may also indicate the status and condition of the various premise devices (e.g., via color coding or other methods, as desired). FIG. **13**F illustrates an alternative user interface, similar to the floor plan of FIG. **13**E. As shown in FIG. **13**F, there may be a floor plan section, which may be similar to that shown in FIG. **13**E. As also shown in FIG. **13**F, on the left side, there may be an application dashboard, e.g., where a user can select each of the icons to select and view an interface of an installed application. As shown on the right side, the interface may include an alerts section where any critical or warning conditions (e.g., as reported by installed applications) can be viewed. As also shown, the user may select a "email store manager" to alert the store manager to one or more of the alert conditions. In the recent activity section, the interface may display any recent messages posted regarding the selected store. In this section, a user may also be

able to post various messages to the recent activity portion. The photos section illustrates photos of the location. In some embodiments, the photos may correspond to installation of the premise devices shown in the floor plan section. The site info section illustrates various contact information and relevant location information. As shown, the site info section shows the location information of the location, the contact information of the manager, and the network that provides connectivity for the location.

[0189] FIG. **13**G illustrates a status screen for each of the fuel tanks at the convenience store. FIG. **13**G may be accessed via selection of one or more of the fuel tanks or the configuration in FIG. **13**D, among other possible methods. As shown, the volume of gas and status of each fuel tank may be visually and/or textually indicated in this interface. As shown, the regular unleaded tank is 9.7% full, the super unleaded tank is 10.9% full, and the diesel tank is 11.5% full. The interface also indicates the capacity for each tank, the height for each tank, and the temperature for each tank. Additionally, as shown, the regular unleaded tank is in condition "warning" because the volume is less than 10%, although other thresholds are envisioned. FIG. **13**G also illustrates a variety of other screens the user can access, such as "network", "clients", "telapoint", "DDNS", "VPN", "ADSL", and "Tank Log", among other possibilities.

[0190] FIG. **13**H illustrates an exemplary configuration interface. In this interface, the user may specify various ranges for conditions of each of the premise devices (or the ranges may be predetermined by the manufacturer of the devices or the developer of the application, as desired). For example, in this exemplary interface, the condition for the gas tanks are "OK" when the percent of gas is above 10, in a warning condition between 5 and 10 percent, and critical when below 5%. Similarly, for the ambient temperature (e.g., the A/C or heating units), there condition is ok between 50 and 70 degrees Fahrenheit and warning when the temperature is outside of that range. Other ranges are shown for the various other devices. As also shown, the user may save the configuration and add new configurations for other devices.

[0191] FIG. **13**I illustrates an exemplary temperature screen where a user may be able to view the temperature over time of various premise devices. As shown, the first mid temp refrigerator has largely been within the ok range (indicated via colors or, in the case of FIG. **13**I, via horizontal lines in the graph and condition indicators) but has been in the warning range several times (e.g., when a door to the refrigerator is left open or during other times). The second mid temp refrigerator is much more consistent and illustrates temperature changes due to the turning on and off of the compressor. FIG. **13**J provides an alternate interface that displays similar temperature graphs of FIG. **13**I.

[0192] FIG. **13**K illustrates an exemplary interface for showing the current status of electronic signs. As shown, the E85 fuel on both sides of the sign is at a price of $4.19 and the biodiesel fuel on both sides of the sign is at a price of $4.24. As also shown in the interface, the status of one of the signs is "L.E.D. outage on panel **2**" indicating there is a problem with the sign that should be fixed. In one embodiment, the user may select the "order parts" or "contact FutureMedia" to address the problem. In further embodiments, the parts may be automatically ordered and/or the manufacturer may be automatically contacted.

[0193] FIG. **13**L illustrates an exemplary interface for a Manitowoc™ Ice Machine. The interface of FIG. **13**L may be similar to the physical interface of the physical machine located at the location. Additionally, the interface illustrates the current status (washing, add chemical, 20 minutes to end) as well as the current settings.

[0194] Thus, FIGS. **13**A-**13**L illustrate exemplary screen shots where a user can view reported information from any of a plurality of different stores and modify configurations of the devices and/or interface.

[0195] FIGS. **14**A-**14**D illustrate exemplary interfaces for managing users and support tickets for a location. As shown in FIG. **14**A, users associated with a location are displayed. More specifically, FIG. **14**A shows the six users associated with the Gas Plus location: Beth Coldwater, Robert Frank, Peter Smith, Carol Hauser, Gas Plus, and Financial Systems. The interface also shows the last time each user has accessed the location, links for contacting each user (e.g., via email, SMS, or other means), and the ability to edit or remove users.

[0196] FIG. **14**B illustrates open tickets for Gas Plus. As shown, there is a ticket that indicates that the gateway device of Gas Plus location **233** is not reporting. As also shown, this ticket is assigned to Mark Wagner and the ticket was last updated by Chris Michaels. As also shown, there is an open ticket to increase the range and bandwidth of pico stations at Gas Plus locations.

[0197] FIG. **14**C illustrates an interface for viewing the non-responding gateway device ticket of FIG. **14**B. As shown in this interface, the ticket was opened by Doug Henning at 10:52 on April 16. The interface also indicates each of the updates to the ticket, the participants who are emailed regarding the ticket (e.g., whenever updates occur for the ticket), a method for responding or updating the ticket (e.g., by typing and submitting a message in the form), and the ability to add more tickets.

[0198] FIG. **14**D is an interface to add notifications, e.g., for a support ticket. As shown, the current interface shows that the email address "Support@gasplus.com" receives emails whenever tickets are opened or updated. As also shown, a custom subject may be provided (e.g., for automatic email sorting or filing) and additional email addresses may provided according to different conditions (e.g., when notes are created).

[0199] Thus, FIGS. **14**A-**14**D illustrate exemplary interfaces that may be used in the described method.

[0200] FIGS. **15**A-**15**C illustrate an exemplary interface for purchasing applications or packages, e.g., from an application store or package warehouse.

[0201] As shown in FIG. **15**A, the user may view current packages, e.g., that are purchased or installed. In this case, the user has a "fuel monitoring" package which is installed on 76 sites, a "Bunn Coffee-O-Matic" package installed on 64 sites, a "FutureMedia Sign" package installed on 4 sites, an "Energy Monitor" package installed on 22 sites, and a "Slurpee Monitor" package installed on 101 sites. In some embodiments, a user can select a package name to see locations using the package.

[0202] The exemplary interface of FIG. **15**A also has a "Find More Packages" portion where a user may browse through different aisles (e.g., categories) of the package warehouse or browse different vendors, although other browsing techniques are envisioned. FIG. **15**A further includes an indication of various detected devices (PointSix People Counter, Hoshizaki Ice Machine, Bunn PostMix Machine) that may require packages. In one embodiment, the user may click on

these detected devices to browse for compatible or suggested packages for those devices, e.g., for installation.

[0203] FIG. **15**B illustrates another interface where the user may view various temperature monitoring packages (e.g., after selecting the "temperature" aisle in FIG. **15**A. In this case, the user may select to install the "PointSix Wireless Temperature Monitoring" package (described as "Easy install sensors. Real-time temps and food safety reports. $12/month per store), the "Aginova Wireless Temperature Monitoring" package (described as "Streamlined sensors. Real-time status and simple graphs. FREE (AGinova sensors required)"), and/or the "Compressor Monkey" package (described as "Device pre-installed on major brand new compressors. Real-time warning alerts. Prevent inventory loss. $8/month per store"). The user may install these packages by selecting the install button for each of the packages. As also shown, the user may browse more packages using a browsing interface similar to that of FIG. **15**A, although other browsing interfaces are envisioned.

[0204] FIG. **15**C illustrates a more detailed description of the "PointSix Wireless Temperature Monitor" package, which may be viewed upon selection of that package. As shown in FIG. **15**C, the package is described as:

[0205] PointSix Wireless Temperature Sensor makes it easy to monitor temperature at any location in your store. Simply attach the sensor to the wall and you are ready to go. AcuVigil seamlessly detects the sensors and this package provides you with real-time data, both current status and history. Set your customized temperature ranges so you can quickly see if the monitored area is within target range. Place inside coolers, deli cases, and other refrigerated zones for automatic food safety reports. FIG. **12**C also shows an exemplary graph that may be provided by the application once installed.

[0206] Thus, FIGS. **15**A-**15**C illustrate an exemplary package installation interface. Further interfaces are envisioned, e.g., where the user specifies an installation location for already purchased applications, or during the purchase process. In some embodiments, the user may browse compatible packages for a specific device (e.g., a managed device, a user device, a gateway device, etc.) or the user may browse packages and then select installation target(s) for the selected packages. Thus, the above interfaces are exemplary only and further interfaces for browsing and installing packages are envisioned.

FIG. **16**—Installing a Gateway Device

[0207] FIG. **16** is a flowchart diagram illustrating one embodiment of a method for installing a gateway device. The method shown in FIG. **16** may be used in conjunction with any of the computer systems or devices shown in the above Figures, among other devices. In various embodiments, some of the method elements shown may be performed concurrently, performed in a different order than shown, or omitted. Additional method elements may also be performed as desired. As shown, this method may operate as follows.

[0208] In **1602**, a picture of a gateway device may be taken, e.g., at the time of installation of the gateway device or upon reaching the target location for installing the gateway device. For example, a technician that is about to install the gateway device at the target location may initially take this picture of the gateway device, e.g., to confirm that he is at the correct location.

[0209] In one embodiment, the picture of the gateway device may be of an image on the gateway device that uniquely identifies the gateway device. For example, the image may be a scan code, such as a bar code, a quick response (QR) code, etc. The scan code may encode a unique identification code for the gateway device (e.g., a serial number or other unique identifier). The picture may be taken by any of various devices, such as a camera, smart phone, or any device with image capturing capabilities.

[0210] In **1604**, location information may be determined at the same location as where the picture is taken in **1602** above. For example, in one embodiment, the device that captured the image in **1602** may also be configured to capture location information, e.g., automatically. More specifically, in one embodiment, the device may include GPS circuitry and may determine latitude and longitude coordinates of the device at the time the picture was taken. Other methods for gathering location information are envisioned, e.g., cell tower triangulation, based on nearby access points or hotspots with known locations, etc. Further, the location information may be any information that identifies the location, e.g., including lat/long coordinates discussed above, addresses, etc.

[0211] In **1606**, the image and the location information may be provided to a server. In one embodiment, the location information may be included in the image (e.g., within metadata of the image). The image and location information may be provided at approximately the same time as when the picture was taken (e.g., using a data connection, such as using a smart phone). Alternatively, the image and location information may be provided at a later time, although this may be less desirable for installations (e.g., when ensuring that the installation is at the correct location).

[0212] The server may be any of the servers discussed above, e.g., which keep track of the various gateway devices. In one embodiment, the server may store a database which includes various locations (e.g., for merchants, convenience stores, homes, etc.) as well as associated location information (e.g., lat/long coordinates). The database may also include a field for associated gateway device(s), e.g., storing unique information associated with each gateway device, such as the information indicated by the image of **1602** above.

[0213] In **1608**, there may be at least two conditions to evaluate. First, where location information of the target location is already known, the method may continue to **1610**. Alternatively, where the location information of the target location is not known (e.g., the specific lat/long coordinates are not known), the method may continue to **1620**.

[0214] In **1610**, the method may determine if the determined location information from **1604** matches the target location information, e.g., stored in the database. For example, the method may match the unique identifier of the gateway device (from **1602** above, e.g., after decoding the image to determine the unique identifier) with a target location and then determine location information of the target location for matching. Alternatively, the method may simply attempt to match the determined location information with any location information, e.g., stored in the database. In this embodiment, where there is a match and the gateway device was not previously associated with the target location, the unique identifier of the gateway device may then be associated with the target location, e.g., in the database.

[0215] The determination of the match may be based on thresholds, e.g., where the specific lat/longs do not match exactly (which is likely). For example, the threshold may be

if the two sets of location information are within 10, 20, 30, 50 feet of each other. In one embodiment, the threshold may vary depending on the particular location. For example, for embodiments where the locations correspond to convenience stores, if the urban density is very low or if there are no other convenience stores nearby, then the threshold may be relatively lax (e.g., within a mile of the target location); however, where there are several convenience stores in a close proximity (e.g., at the same intersection), the threshold may be much smaller (e.g., less than 25 feet).

[0216] In **1612**, an indication of whether the location information matches the target location information may be provided, e.g., to the user who took the picture in **1602** above. For example, where the image was taken via a smart phone (or other mobile communication device), the notification may be provided to the smart phone, or even another device associated with the user, as desired.

[0217] This branch of the method may be particularly useful for ensuring that technicians are installing the gateway device at the proper target location. For example, following the embodiment discussed above where there are multiple convenience stores within a small radius, technicians may often attempt to install the gateway device at the wrong location, which is an inefficient use of time. By following the **1610**-**1612** branch, it may be ensured that the gateway device is installed at the proper location.

[0218] Turning now to **1620**, where the location information of the target location is not known, the method may determine a target location associated with the unique information indicated by the image (e.g., by decoding the image to determine the unique information).

[0219] After determining the target location in **1620**, the location information determined in **1604** may be associated with the target location and/or gateway device in **1622**. For example, the database may be updated to include the location information in an entry associated with the target location and/or gateway device.

[0220] The various steps discussed above (e.g., **1608**-**1622**) may be performed by the server. Alternatively, various information usable for performing those steps may be provided from the server to a device (e.g., the device that captured the image) which may be configured to implement the steps above. For example, the target location information may be provided by the server and the matching may be performed by the device itself. In this embodiment, the location information determined in **1604** may not be necessarily provided to the server, e.g., the server may provide the target location information based on the image of the gateway device and the device may perform the matching using the target location information.

[0221] While the method of FIG. **16** is described above with respect to gateway device installation, it may also apply to database clean-up. For example, a technician may service a particular location (e.g., a gateway device at the particular location). Accordingly, the steps described above may be performed after installation. In this case, previous location information may have already been stored, but may be incorrect or inaccurate. Accordingly, the new location information may be used to overwrite the old location information associated with the target location, e.g., in the database. Thus, the method may be performed similar to **1620** and **1622** above, except for overwriting previous location information rather than providing new location information.

Further Embodiments

[0222] In some embodiments, the user may be able to configure automatic purchases of products, e.g., as the devices indicate the need or desirability to make the purchases. For example, an application may be able to determine that a filter should be replaced for an A/C unit. Accordingly, the application (or an associated application) may be able to order a new filter or request service of the A/C unit automatically without an employee or manager having to deal with the issue. Similarly, when inventory of a product is low, more of the product may be automatically ordered (e.g., coffee filters, grocery products, fuel for a fuel tank, refrigerant for a refrigerator, etc.). Thus, applications may be configured to automatically purchase items as the need arises, as desired.

[0223] Some applications may be default or "irreplaceable" applications that are always installed on gateway devices (or always installed for specific types of gateway devices). For example, one such application may be a device reporting application that maintains a list of premise devices that are detected and/or controlled by the gateway device. A user may be able to use the application to view or manage premise devices connected to the gateway device. Accordingly, such an application may be installed on every gateway device. Such applications may not be removable from the device (e.g., a user may not be able to remove the application from the gateway device or management server). However, third parties may develop an application with a similar purpose as one of these default or irreplaceable applications. In one embodiment, the third party and/or the application server may mark the application as a replacement (or candidate for replacement) for the default or irreplaceable applications. Accordingly, when such an application is selected for installation, the original application may be removed and replaced with the new application (e.g., in response to user input confirming the replacement or automatically, as desired). In some embodiments, the new application may be associated with a newer device (or piece of hardware). For example, a user may purchase a new switch which may have nicer features with respect to device management. Accordingly, the user may install a corresponding application that is able to use these nicer features. However, the replacement application may be purely functional (e.g., not associated with new hardware or a specific hardware device) and the user may simply wish to replace the original default software application with a different third party application (e.g., which is better or that the user prefers).

[0224] In some embodiments, manufacturers (or other entities) may provide services or products in response to detected activity. For example, a manufacturer may specify a maintenance schedule for a premise device that helps ensure that the device lasts for its operation lifetime. As one example, an HVAC unit may have a maintenance schedule where Freon (or a corresponding substitute) is replaced every six months (or before a certain level is reached) and air filters are replaced every 6 months. Since the gateway device may be able to monitor the condition of the premise device, the method may be able to determine whether or not the prescribed maintenance schedule has been followed. For example, if a manufacturer (or other entity) determines that the maintenance schedule has been followed (e.g., using reported information), the manufacturer may extend the warranty on the

premise device, lower future maintenance costs, lower the cost a service contract, lower costs of future purchases or services, provide a free product (e.g., a free air filter for an HVAC), etc. Accordingly, the customer may be rewarded for properly maintaining the equipment, and the manufacturer can maintain its reliability standards. Conversely, a manufacturer (or other entity) may be able to monitor the condition or maintenance of a premise device and determine that the maintenance schedule is not being followed and/or the premise device is not being operated as intended. Accordingly, the manufacturer (or a warranty provider) may revoke or shorten the warranty since the user is not meeting the terms of the warranty agreement.

[0225] In one embodiment, one of the applications may assist in performing external security scans of the gateway device. For example, in order to handle sensitive information, such as credit card information, the gateway device may need to be periodically scanned for vulnerabilities (e.g., a security company may attempt to penetrate the gateway device in order to steal the sensitive matter). Since the gateway devices will typically have a dynamically assigned IP address, performing these scans are somewhat difficult for a third party. Accordingly, one of the applications may determine the IP address of the gateway device and submit a scanning request to a security company to perform the scan (e.g., a PCI scan). That application may be developed and/or associated with security company. Additionally, the result of the security scans or attempts to hack the gateway device may be provided via the management server using the application.

[0226] In some embodiments, one of the applications may take photographs in response to various conditions at the location. For example, since refrigerators typically have increased temperatures when a door is opened (e.g., above a food safety level), a corresponding camera may be directed at the refrigerator door. The camera may be configured to take a photograph whenever the door is opened and/or whenever the temperature exceeds a warning or critical level. Accordingly, a manager of the store may be able to monitor and determine which employee is routinely causing the temperature to increase above the threshold level (e.g., by leaving the door open). The camera may also be used to keep track of security issues (e.g., employees stealing food from the refrigerator, etc. This concept may be performed by an application (e.g., an application that control the camera to take the picture in response to detected stimuli or alerts). The concept may be extended to performing any action (e.g., controlling action) based on stimuli occurring at the location.

[0227] In some embodiments, each application and/or device may have a security policy. For example, applications that handle sensitive information (e.g., credit card information) may be executed in a different portion of the gateway device (e.g., by secure logic of the gateway device) in order to securely handle the information. In some embodiments, security policies may dictate how applications, devices, and/or third parties (e.g., communicating with the gateway device over the Internet) are able to communicate. For example, an application for monitoring fuel tank devices may have a security policy that only allows communication with the fuel tank devices, and not, for example, devices that handle financial information. Similarly, the fuel tank devices (or applications associated therewith) may only provide the reported information to specific servers or third parties over the Internet. The security policy may be implemented by the gateway device and/or management server. For example, the gateway device may control how an application communicates with devices controlled by the gateway device as well as how the application provides information to third parties, e.g., over the Internet.

[0228] In one embodiment, an application may request to communicate with one of the premise devices controlled by the gateway device, and the gateway device may determine whether or not that application is allowed, based on the security policy. In this embodiment, the gateway device may use rules to determine that a electronic sign application is only able to control or retrieve reported information from electronic signs, and not any other device. Accordingly, the request may be accepted if it meets the security policy and rejected if it does not. Similarly, an application may request to communicate with a server or other entity over a network, such as the Internet. The gateway device may determine whether the communication is allowed using a security policy. For example, the application may only be allowed to communicate with specific IP addresses, domain names, etc., and may not be allowed to access other entities on the Internet (in order to protect the reported information).

[0229] Similarly, third party communications may be controlled as well. For example, in one embodiment, all outside communication may be prohibited. In other embodiments, a server associated with a device or application may be allowed to provide incoming communications, but the communications may be restricted to certain applications or devices. For example, communications from the energy company may be restricted to applications or devices associated with that energy company. Similarly, any device communications may be controlled via security policies. Thus, the security policies may be used to control communications between third parties, applications, and/or devices. The security policies may also be extended to providing information to or receiving information from third parties on the cloud server (e.g., the management server), devices associated with users, etc.

[0230] Alternatively, or additionally, some applications may communicate over a VPN based on security policies. Further types of policies may be assigned to each application, e.g., within a user interface provided by the management server.

Exemplary Use Cases and Applications

[0231] The following provides exemplary locations and corresponding examples of use that may apply to the methods described herein.

[0232] The owner of an automatic (coin-op) car wash wants to be notified via text message when there is a fault in the carwash. So, he downloads a free application from the car wash manufacturer through the server. Later, the owner decides he wants to be able to view consumable levels remotely and/or be notified when consumable fluids (soap, wax, etc.) are low so he upgrades and purchases the premium application. He later decides he would like to automatically purchase consumables and makes a configuration change in the application allowing in-application purchases. In one embodiment, the managing company of the gateway device or server may collect a fee for the initial premium application purchase and for each in-application purchase.

[0233] A busy homeowner wants to be able to adjust her thermostat remotely because she frequently forgets to turn it up when she goes out of town for a race. So, she downloads the free thermostat application from the appropriate manufacturer from the server. Later, when she finds the air filter to

be long-overdue for a change, she upgrades the application to a premium application that monitors air flow and knows when it is time to change the filter. In-application purchases allow her to purchase and have a new filter sent to her house. In some embodiments, the filter may be purchased automatically without her requesting the purchase. Similar to above, the managing company of the gateway device or server may collect a fee for the initial premium application purchase and for each in-application purchase.

[0234] An owner of a convenience store wants to ensure that the information controlled by his gateway device is secure. Accordingly, the owner downloads a "freemium" application for performing security scans. The application may allow for an unlimited number of local scans (e.g., within the network, possibly against an internal firewall of the gateway device). However, once set up properly, the owner may then make an in-application purchase (e.g., a signing a monthly contract) for performing periodic vulnerability scans of the gateway device from an external server. The application may be configured to determine the IP address before each scan (e.g., where the IP address is dynamically assigned), report the IP address to the external server, and the external server may perform the external scan for the owner. The owner then may receive an email or may otherwise view the results of the scan (e.g., the report may be provided on the management server or within an application that is executed by the owner's smart phone).

[0235] Although the embodiments above have been described in considerable detail, numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. For example, while many of the embodiments above are described with respect to a cloud server, simpler embodiments involving a single server are also envisioned. It is intended that the following claims be interpreted to embrace all such variations and modifications.

1. A non-transitory computer readable memory medium comprising program instructions for automatically configuring a gateway device, wherein the program instructions are executable to:

detect a gateway device that is deployed at a location;

receive an indication of a plurality of premise devices coupled to the gateway device at the location;

automatically determine a plurality of applications to be deployed on the gateway device, wherein said automatically determining is based on the indication of the plurality of premise devices coupled to the gateway device;

provide the plurality of applications to the gateway device over a wide area network, wherein the gateway device is configured to execute the plurality of applications, wherein at least one of the applications is executable by the gateway device to manage a first premise device of the plurality of devices at the location.

2. The non-transitory computer readable memory medium of claim **1**, wherein the indication of the plurality of premise devices comprises identification information of each premise device coupled to the gateway device.

3. The non-transitory computer readable memory medium of claim **2**, wherein the identification information identifies a type of each premise device.

4. The non-transitory computer readable memory medium of claim **2**, wherein the identification information identifies a model of each premise device.

5. The non-transitory computer readable memory medium of claim **1**, wherein the identification information identifies a serial number of each premise device.

6. The non-transitory computer readable memory medium of claim **1**, wherein the indication comprises identification information of the first premise device coupled to the gateway device, wherein said automatically determining the plurality of applications comprises determining the at least one of the applications for managing the first premise device coupled to the gateway device using the identification information of the first premise device.

7. The non-transitory computer readable memory medium of claim **1**, wherein the indication comprises identification information for each of the plurality of premise devices, wherein said automatically determining the plurality of applications comprises determining one or more applications associated with each identification information.

8. The non-transitory computer readable memory medium of claim **1**, wherein at least a subset of the applications are designed for distributed execution by the gateway device and a server, wherein the program instructions are further executable to:

automatically install the at least a subset of the applications on the server.

9. The non-transitory computer readable memory medium of claim **8**, wherein the program instructions are executed by a cloud server, wherein said automatically installing the at least a subset of the applications is performed on the cloud server.

10. The non-transitory computer readable memory medium of claim **1**, wherein the gateway device is configured to execute a first application of the at least one of the applications to control the first premise device at the location.

11. The non-transitory computer readable memory medium of claim **1**, wherein the gateway device is a merchant gateway device of a merchant located at a merchant location.

12. The non-transitory computer readable memory medium of claim **11**, wherein the merchant is a convenience store.

13. The non-transitory computer readable memory medium of claim **1**, wherein the gateway device is located at a home of a user, wherein the plurality of premise devices are located in the home of the user.

14. A method for automatically configuring a gateway device, comprising:

at least one computer system detecting a gateway device that is deployed at a location;

the at least one computer system receiving an indication of a plurality of premise devices coupled to the gateway device at the location;

the at least one computer system automatically determining a plurality of applications to be deployed on the gateway device, wherein said automatically determining is based on the indication of the plurality of premise devices coupled to the gateway device;

the at least one computer system providing the plurality of applications to the gateway device over a wide area network, wherein the gateway device is configured to execute the plurality of applications, wherein at least one of the applications is executable by the gateway device to manage a first premise device of the plurality of devices at the location.

15. The method of claim 14, wherein the indication of the plurality of premise devices comprises identification information of each premise device coupled to the gateway device.

16. The method of claim 14, wherein the indication comprises identification information of the first premise device coupled to the gateway device, wherein said automatically determining the plurality of applications comprises determining the at least one of the applications for managing the first premise device coupled to the gateway device using the identification information of the first premise device.

17. The method of claim 14, wherein at least a subset of the applications are designed for distributed execution by the gateway device and the at least one computer system, wherein the method further comprises:

the at least one computer system automatically installing the at least a subset of the applications.

18. The method of claim 14, wherein the gateway device is a merchant gateway device of a merchant located at a merchant location.

19. The method of claim 18, wherein the merchant is a convenience store.

20. The method of claim 18, wherein the gateway device is located at a home of a user, wherein the plurality of premise devices are located in the home of the user.

21. A non-transitory computer readable memory medium comprising program instructions for automatically configuring a gateway device at a location, wherein the program instructions are executable to:

determine a plurality of premise devices coupled to the gateway device at the location;

provide an indication of the plurality of premise devices to a server over a wide area network;

receive a plurality of applications from the server over the wide area network, wherein the plurality of applications are automatically determined based on the provided indication;

execute the plurality of application programs, wherein a first application is executable to manage a first premise device of the plurality of premise devices at the location.

22. The non-transitory computer readable memory medium of claim 21, wherein said determining the plurality of premise devices is performed automatically by the gateway device without receiving user input specifying the plurality of devices.

23. The non-transitory computer readable memory medium of claim 21, wherein said determining the plurality of premise devices comprises receiving user input identifying a premise device of the plurality of premise devices.

24. The non-transitory computer readable memory medium of claim 21, wherein the indication of the plurality of premise devices comprises identification information of each premise device coupled to the gateway device.

25. The non-transitory computer readable memory medium of claim 24, wherein the identification information identifies a type of each premise device.

26. The non-transitory computer readable memory medium of claim 24, wherein the identification information identifies a model of each premise device.

27. The non-transitory computer readable memory medium of claim 24, wherein the identification information identifies a serial number of each premise device.

28. The non-transitory computer readable memory medium of claim 21, wherein the indication comprises identification information of the first premise device coupled to the gateway device, wherein the first application is automatically determined by the server for managing the first premise device coupled to the gateway device using the identification information of the first premise device.

29. The non-transitory computer readable memory medium of claim 21, wherein at least a subset of the applications are designed for distributed execution by the gateway device and the server, wherein the at least a subset of the applications are also executed by the server.

30. The non-transitory computer readable memory medium of claim 21, wherein the gateway device is a merchant gateway device of a merchant located at a merchant location.

31. The non-transitory computer readable memory medium of claim 30, wherein the merchant is a convenience store.

32. The non-transitory computer readable memory medium of claim 21, wherein the gateway device is located at a home of a user, wherein the plurality of premise devices are located in the home of the user.

33. The non-transitory computer readable memory medium of claim 21, wherein the program instructions are further executable to:

upon establishing a network connection to the wide area network, automatically provide a message to the server.

34. The non-transitory computer readable memory medium of claim 34, wherein the message comprises identification information of the gateway device.

35. A method for automatically configuring a gateway device at a location, comprising:

determining a plurality of premise devices coupled to the gateway device at the location;

providing an indication of the plurality of premise devices to a server over a wide area network;

receiving a plurality of applications from the server over the wide area network, wherein the plurality of applications are automatically determined based on the provided indication;

executing the plurality of application programs, wherein a first application is executable to manage a first premise device of the plurality of premise devices at the location.

36. The method of claim 35, wherein said determining the plurality of premise devices is performed automatically by the gateway device without receiving user input specifying the plurality of devices.

37. The method of claim 35, wherein said determining the plurality of premise devices comprises receiving user input identifying a premise device of the plurality of premise devices.

38. The method of claim 35, wherein the indication of the plurality of premise devices comprises identification information of each premise device coupled to the gateway device.

39. The method of claim 35, wherein at least a subset of the applications are designed for distributed execution by the gateway device and the server, wherein the at least a subset of the applications are also executed by the server.

40. The method of claim 35, wherein the gateway device is a merchant gateway device of a merchant located at a merchant location.

41. The method of claim 35, wherein the gateway device is located at a home of a user, wherein the plurality of premise devices are located in the home of the user.

42. The method of claim **35**, further comprising:

upon establishing a network connection to the wide area network, automatically providing a message to the server.

43. The method of claim **42**, wherein the message comprises identification information of the gateway device.

44. A gateway device, comprising:

a first one or more interfaces for communicating with a plurality of premise devices collocated with the gateway device;

a second one or more interfaces for communicating with a server over a wide area network (WAN);

one or more processors;

one or more memory mediums coupled to the one or more processors, wherein the one or more memory mediums are configured to store applications that are executable by the one or more processors to manage the plurality of premise devices;

wherein the gateway device is configured to:

detect the plurality of premise devices collocated with the gateway device;

provide an indication of the plurality of premise devices to a server over the WAN;

receive a plurality of applications from the server over the WAN, wherein the plurality of applications are automatically determined based on the provided indication;

store the plurality of applications in the one or more memory mediums based on said receiving; and

execute the plurality of application programs to manage at least a subset of the plurality of premise devices.

45. The gateway device of claim **44**, further comprising:

a modem coupled to the second one or more interfaces, wherein the modem is configured to provide communication capabilities over the WAN.

46. The gateway device of claim **44**, wherein the first one or more interfaces comprise one or more of:

one or more local area network interfaces;

one or more serial interfaces; or

a wireless local area network interface.

47. The gateway device of claim **44**, further comprising:

a router, wherein the router is configured to provide a local area network (LAN) for the plurality of premise devices.

* * * * *

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

U.S. Patent No. 11,695,823	)	Attorney Docket No. UNP-16002
	)	
Issued: July 4, 2023	)	
	)	
Filed: January 6, 2021	)	
	)	
Title: DISTRIBUTED SOFTWARE DEFINED NETWORKING	)	
	)	
	)	
	)	

REQUEST FOR *EX PARTE* REEXAMINATION OF U.S. PATENT NO. 11,695,823

Mail Stop *Ex Parte* Reexam
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Dear Commissioner:

Pursuant to the provisions of 35 U.S.C. §§ 301-307, Unified Patents, LLC ("Requester") hereby requests an *ex parte* reexamination of claims 1, 2, 3, 4, 8, 15 and 19 (the "Challenged Claims") of U.S. Patent No. 11,695,823 ("the '823 Patent") (Ex. 1001), which issued on July 4, 2023 from U.S. Patent Application 17/142983 ("the '983 Application"), filed on January 6, 2021. The '823 Patent is a continuation of U.S. Patent Application No. 16/900,963, which was filed on June 14, 2020 and which has issued as U.S. Patent No. 10,893,095, which is a continuation of U.S. Patent Application No. 15/836,824, which was filed on December 09, 2017 and has issued as U.S. Patent No. 10,686,871, which is a continuation of U.S. Patent Application No. 14/295,331, which was filed on June 4, 2014 and has issued as U.S. Patent No. 9,843,624, which claims the benefit of U.S. Provisional Patent Application No. 61/834807, which was filed on June 13, 2013[1]. The '823 Patent was filed with Big Data Federation, Inc. as the Applicant ("Applicant"), but is currently assigned to Edge Networking Systems, LLC (the "Patent Owner"). An assignment from Big Data Federation, Inc. to the first named inventor, Pouya Taaghol, is recorded in the U.S. Patent and Trademark Office ("USPTO") at reel/frame 063290/0800, and an assignment from

[1] At this time, Requester assumes June 13, 2013 to be the priority date of the '823 Patent. Requester reserves the right to challenge the priority date in a different proceeding.

Pouya Taaghol to Edge Networking Systems, LLC is recorded in the U.S. Patent and Trademark Office ("USPTO") at reel/frame 063454/0352.

Requester submits that this Request presents prior art references and analyses that are noncumulative of the prior art that was before the examiner during the original prosecution of the '823 Patent and that the Challenged Claims are invalid over these references. Requester therefore requests an order for reexamination and an Office Action rejecting claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent.

***Ex Parte* Patent Reexamination Filing Requirements**

This Request is being submitted during the period of enforceability of the '823 Patent pursuant to 37 C.F.R. § 1.510(a), and the fee set forth in 37 C.F.R. § 1.20(c)(2) is being submitted herewith.

Pursuant to 37 C.F.R. § 1.510(b)(1), statements pointing out at least one substantial new question of patentability based on material, non-cumulative prior art patents for the Challenged Claims are provided in Section IV of this Request. None of the prior art cited herein was considered during the original examination of the '823 Patent.

Pursuant to 37 C.F.R. § 1.510(b)(2), reexamination of the Challenged Claims is requested, and a detailed explanation of the pertinency and manner of applying the cited prior art to the Challenged Claims is provided in Section V of this Request.

Pursuant to 37 C.F.R. § 1.510(b)(3), copies of every patent relied upon or referred to in the statement pointing out each substantial new question of patentability or in the detailed explanation of the pertinency and manner of applying the cited prior art are provided as Exhibits 1001-1005 and Appendices A and B of this Request.

Pursuant to 37 C.F.R. § 1.510(b)(4), a copy of the '823 Patent is provided as Ex. 1001 of this Request.

Pursuant to 37 C.F.R. § 1.510(b)(5), the Certificate of Service attached hereto indicates that a copy of this Request, in its entirety, has been served on Patent Owner at the following the correspondence address in the patent file, in accordance with 37 C.F.R. § 1.33(c):

Roark IP
Attention: Christopher Horgan
1438 Dahlia Loop
San Jose, CA 95126

UNITED STATES

Pursuant to 37 C.F.R. § 1.510(b)(6), Requester hereby certifies that neither the statutory estoppel provisions of 35 U.S.C. § 315(e)(1) nor 35 U.S.C. § 325(e)(1) prohibit Requester from filing this Request.

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
447

TABLE OF CONTENTS

	SECTION	PAGE
I.	OVERVIEW OF THE '823 PATENT	1
	A. Summary of the '823 Patent	1
	B. Prosecution History of the '823 Patent	4
II.	CLAIM CONSTRUCTION	5
	A. General Principles	5
	B. Construction of "Unified Capabilities"	6
	C. Construction of "Enabling…"	6
	D. Construction of Functional Language	8
III.	LEVEL OF SKILL OF A PERSON OF ORDINARY SKILL IN THE ART	10
IV.	STATEMENT POINTING OUT EACH SUBSTANTIAL NEW QUESTION OF PATENTABILITY FOR THE CHALLENGED CLAIMS	10
	A. Listing of Prior Art Patents and Printed Publications	10
	B. Ground 1: Stewart Anticipates the Challenged Claims	11
	1. Overview of Stewart	12
	C. Grounds 2: Stewart Renders Obvious the Challenged Claims	16
	1. Overview of the Knowledge of a POSITA at the time of the '823 Patent	17
	2. Stewart in View of the Knowledge of a POSITA	17
	D. Ground 3: Malik Anticipates Claims 1, 2, 3, 4, 15 and 19	18
	1. Overview of Malik	19
	E. Ground 4: Malik Renders Obvious Claims 1, 2, 3, 4, 15 and 19	23
V.	DETAILED APPLICATION OF THE PRIOR ART TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED	24
	A. Proposed Rejections of Clams	24
	B. Secondary Considerations	25
VI.	DISCLOSURE OF CONCURRENT LITIGATION, REEXAMINATION, AND RELATED PROCEEDINGS	25
VII.	CONCLUSION	26

TABLE OF APPENDICES AND EXHIBITS

Appendix	Description
Appendix A	Claim Chart for Grounds 1 and 1 of Unpatentability of the '823 Patent (Ground 1: Anticipation in view of Stewart; Ground 2: Obviousness in view of Stewart)
Appendix B	Claim Chart for Grounds 3 and 4 of Unpatentability of the '823 Patent (Ground 3: Anticipation in view of Malik; Ground 4: Obviousness in view of Malik)

Exhibit	Description
Ex. 1001	U.S. Patent No. 11,695,823 ("the '823 Patent")
Ex. 1002	File History of the '823 Patent
Ex. 1003	Declaration of James T. Geier
Ex. 1004	U.S. Pub. No. 20120278454 to Stewart et al. (Stewart)
Ex. 1005	U.S. Pub. No. 20130054682 to Malik et al. (Malik)

Exhibit 1084
Microsoft Corporation v. Edge Networking Systems, LLC
449

I. OVERVIEW OF THE '823 PATENT

A. Summary of the '823 Patent

The '823 Patent generally relates to "a network architecture that facilitates secure and flexible programmability between a user device and across a network," '823 Patent (Ex. 1001) at 1:14-16, and more specifically to a "Distributed Software Defined Network (dSDN)" that purports to be "an architecture that enables secure and flexible programmability across a network with full lifecycle management of services and infrastructure applications (fxDeviceApp)" that "also harmonizes application deployment across the network independent of the hardware vendor." *Id.* at Abstract and 2:31-39. The disclosure of the '823 Patent is applicable to "many different types of networks," although is specifically disclosed with reference to a "cellular network." *Id.* at 8:62-67.

Claims 1 and 19 are the independent claims of the '823 Patent, while claims 2-18 each depend directly or indirectly from independent claim 1. Claims 1 and 19 are reproduced below:

> 1. A system comprising:
> a programmable network device adapted to host a plurality of network device applications;
> a programmable cloud device adapted to host a plurality of cloud applications, wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications; and
> wherein the plurality of network device applications and plurality of cloud applications device form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware.

> 19. A system comprising:
> a programmable network device adapted to host a plurality of network device applications;
> a programmable cloud device adapted to host a plurality of cloud applications, wherein the plurality of network device applications and the plurality of cloud applications are in secure communication with each other to form distributed applications;
> wherein the plurality of network device applications and plurality of cloud applications form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware;

an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications; and

an infrastructure application marketplace in communication with the application management portal, said infrastructure marketplace capable of providing tested distributed applications to the application management portal.

Independent claims 1 and 19 are both directed to systems. The system of claim 19 includes all of the elements of the system of claim 1, as well as two additional elements, namely the application management portal and the infrastructure application marketplace. The Specification of the '823 Patent describes examples of systems as recited in claims 1 and 19. For example, a system that includes the elements recited in both claims 1 and 19 is illustrated in Fig. 3 of the '823 Patent, as shown below:

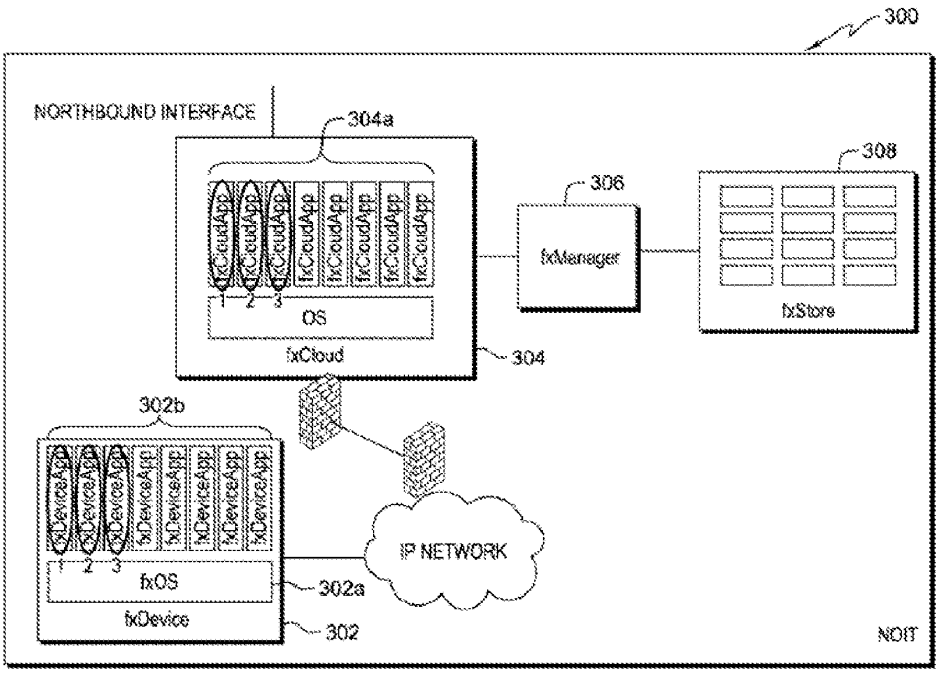


FIG. 3

As shown in Fig. 3 of the '823 Patent, "[t]he dSDN system 300 consists of a flexible network device (fxDevice) 302, a flexible cloud platform (fxCloud) 304, an application management portal (fxManager) 306, and an infrastructure application market place (fxStore) 308." '823 Patent (Ex. 1001) at 10:9-13. With respect to Fig. 3, the '823 Patent explains that "[t]he fxDevice 302 may host several independent and securely isolated applications (on top of

fxOS 302 a) named here fxDeviceApp 302 b" and that "fxDeviceApp 302 b may have a sister app in the backend cloud infrastructure (i.e., flexible cloud platform 304) referenced here as fxCloudApp 304 a." *Id.* at 10:21-27. The '823 Patent also states that "[t]he fxCloudApp 304 a in the cloud is paired with its fxDeviceApp 302 b in the fxDevice 302" and that they "collectively form a distributed application (dApp or fxApp)." *Id.* at 10:27-31. Thus, in the '823 Patent, "when fxDeviceApp is referenced it shall mean any application that may have software components in the fxDevice, fxCloud, or both." *Id.* at 10:31-33.

Further, the specification of the '823 Patent states that "[A]s used in this disclosure in describing the dSDN, a database may be either a single unified system or a distributed system wherein certain database elements are located upon different systems or servers which may be in different physical locations, acting in harmony to appear as one unified database." *Id.* at 4:43-47.

With reference to Fig. 4, reproduced below, which illustrates an examplary view of the software layers in the operating system (fxOS 302a) of the network device 302, the '823 Patent explains that "[a]bove the kernel, there are several tools 416, native daemons 418, native libraries 420 and Hardware Abstraction Layers (HAL) 422" and that "these tools abstract out the hardware dependencies for the upper layers and programmers." *Id.* at 16:38-51.

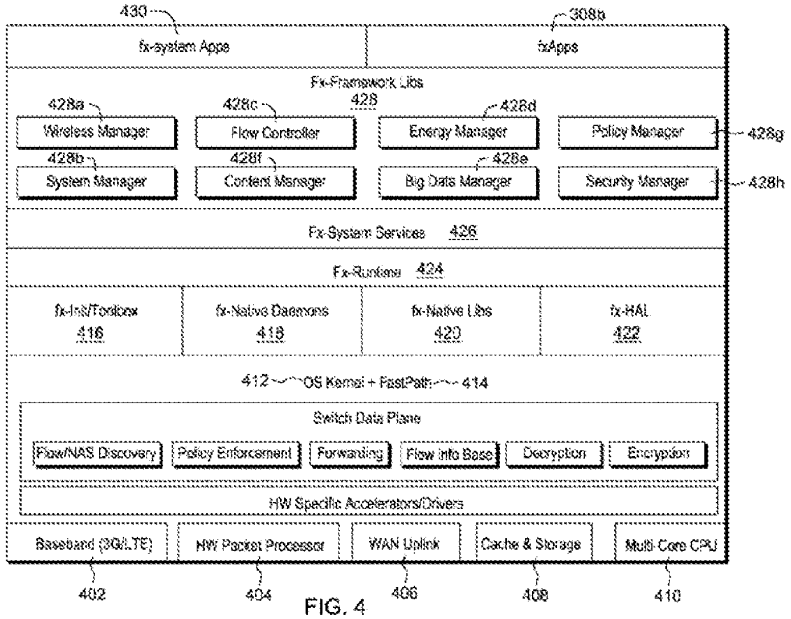


FIG. 4

With respect to application programming interfaces (APIs), the Specification of the '823 Patent discloses a flexible Software Development Kit (fxSDK) and explains that:

> The fxSDK is unique in several aspects exemplified as follows. First, since the dSDN applications may have a device component (fxDevice 302) and cloud component (fxCloud 304), the fxSDK allows the developer to develop the components together to simplify the development and testing. Second, the dRMS offers various system level APIs with remote accessibility. In other words, the APIs in the fxDevice 302 could be accessed via the applications in the fxCloud (fxCloudApps) and vice versa.

Id. at 16:38-51. The '823 Patent also discloses an "Extensible API Framework" that "allows the developer to create custom APIs and make it accessible for other applications (in the fxDevice or in the fxCloud)" and that "[i]n turn, the fxCloud could present these APIs using e.g. REST technologies (via the fxCloud Northbound Interface) to the developer outside the dSND system." *Id.* at 28:49-54.

B. Prosecution History of the '823 Patent

The '823 Patent was origianlly filed as the '983 Application on January 6, 2021. '823 Patent (Ex. 1001) at cover page. The '823 Patent is a continuation of U.S. Patent Application No. 16/900,963, which was filed on June 14, 2020 and which has issued as U.S. Patent No. 10,893,095, which is a continuation of U.S. Patent Application No. 15/836,824, which was filed on December 09, 2017 and has issued as U.S. Patent No. 10,686,871, which is a continuation of U.S. Patent Application No. 14/295,331, which was filed on June 4, 2014 and has issued as U.S. Patent No. 9,843,624, which claims the benefit of U.S. Provisional Patent Application No. 61/834807, which was filed on June 13, 2013.[2] A copy of the prosecution history of the '823 Patent is provided in Exhibit 1002 filed herewith.

The application for the '823 Patent was initially filed with eighteen claims, including independent claims 1, 15, and 17. Ex. 1002 at pp. 427-432. However, all of the originally filed claims were rejected on statutory double patenting grounds in view of the immediate parent patent (U.S. Patent 10,893,095) in a first non-final Office Action dated January 5, 2022, *Id.* at pp. 168-172, and the rejection was maintained in the first final rejection that sent on April 8, 2022. *Id.* at pp. 133-137. In response, on July 19, 2022, Applicant cancelled all of the originally filed claims and filed new claims 19-38. *Id.* at pp. 123-130. In a second non-final Office Action dated September 13, 2022, claims 19-36 were rejected under 35 U.S.C §112(a) as not having proper

[2] Because the America Invents Act (AIA) went into effect March 16, 2013, the '823 Patent is subject to treatment under the rules as amended in the AIA.

written description support for "wherein the programmable network device and programmable cloud device form unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) that allow for simultaneous programming of the programmable network device and programmable cloud device independent of network device hardware and cloud device hardware." *Id.* at pp. 094-097. Only claims 37 and 38 of the new claims were rejected based on prior art, being rejected as being anticipated by U.S. Pub. No. 20130054763 to Van Der Merwe et al. *Id.* at pp. 098-099.

In a response filed on September 19, 2022, Applicant cancelled claims 37 and 38, significantly amended each of claims 19-36 and added a new independent claim 39. *Id.* at pp. 084-090. Independent claims 19 and 39 as presented were substantially the same as issued claims 1 and 19, except for the inclusion of the term "simultaneously" in the wherein clause regarding "unified capabilities enabling a plurality of upper layer application programming interfaces (APIs) to simultaneously program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware." *Id.* at pp. 086 and 090. Applicant also argued that paragraph [0030] of the Specification provided support for the use of the term "simultaneously."

In a seocnd final Office Aciton dated December 16, 2022, the Applicant's argument regarding the term "simultaneously" were rejected, and all of the claims were once again rejected as not having proper written description support. In a response filed on April 9, 2023, Applicant amended claims 19 and 39 to remove the term "simultaneously."

A Notice of Allowance was mailed on May 9, 2023. *Id.* at pp. 010-014. No reasons for allowability were given.

II. CLAIM CONSTRUCTION

A. General Principles

The patent claims in an ex parte reexamination proceeding are to be given the broadest reasonable interpretation consistent with the specification and limitations in the specification are not read into the claims. MPEP §2258(I)(G) (citing *In re Yamamoto*, 740 F.2d 1569 (Fed. Cir. 1984)). "The broadest reasonable interpretation does not mean the broadest possible interpretation." MPEP §2111. Instead, the meaning must be "consistent with the ordinary and customary meaning of the term (unless the term has been given a special definition in the

specification), and must be consistent with the use of the claim term in the specification and drawings." *Id.* The broadest reasonable interpretation of the claims must also "be consistent with the interpretation that those skilled in the art would reach." *Id.* (citations omitted). The claims and individual terms therein are construed in this Request in accordance with these requirements, and Requester does not believe any terms require any explicit construction apart from their plain and ordinary meaning. Requester reserves the right to advocate a different claim interpretation in any other forum if necessary. Nonetheless, below, Requester addresses certain claim terms found in the independent claims of the '823 Patent to provide additional context as to what the claim terms encompass.

B. Construction of "Unified Capabilities"

Requester notes that claims 1 and 19 of the '823 Patent use the phrase "wherein the plurality of network device applications and plurality of cloud applications device form **unified capabilities**. . . ." '823 Patent (Ex. 1001) at claims 1 and 19.[3] However, the specification of the '823 Patent does not use the term "unified capabilities," and the term was not addressed during the prosecution history. The only description provided in the specification of the '823 Patent involving the use of the term "unified" states that "As used in this disclosure in describing the dSDN, a database may be either a **single unified system** or a distributed system wherein certain database elements are located upon different systems or servers which may be in different physical locations, acting in harmony to appear as one unified database." *Id.* at 4:43-47. Accordingly, in the context of claims 1 and 19, which recites the combination of a plurality of network device applications and a plurality of cloud applications, the broadest reasonable interpretation of the term "unified capabilities" should at least encompass network device applications and cloud applications that, although distributed, act in a unified manner; that is, act in harmony or in concert so as to appear as unified application(s). Declaration of James T. Geier (the "Geier Decl.") (Ex. 1003) at ¶¶ 24-25.

C. Construction of "Enabling…"

Additionally, Requester notes that claims 1 and 19 of the '823 Patent recite "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device

[3] Unless otherwise noted, all bold used in quotes is emphasis added.

hardware and cloud device hardware" (the "Enabling Clause"). *Id.* at claims 1 and 19. To the extent that this limitation is given patentable weight despite being functional language (see Section II(D) below, the '823 Patent only mentions the term "upper layer" on one occasion when the '823 Patent states, with reference to FIG. 4, as follows:

> **FIG. 4 demonstrates an examplary view of the software layers** in the fxOS 302a. Each layer of this software stack is modified and redefined to optimize for embedded networking equipment of wireless and wireline nature. Starting from the bottom of the stack, the kernel also connects to the hardware resources via the device drivers. Examples of such resources are wireless baseband SoC 402, hardware packet processor 404, Wide Area Network (WAN) uplink interface 406, cache and storage 408, and multi-core CPU 410. The OS kernel 412 is enhanced with the FastPath function 414 to accelerate packet processing (by help of hardware features such as Intel's Data Plane Development Kit (DPDK) or pure software enhancements) and other related data path functions. The enhanced kernel with FastPath 412, 414 would enable visibility into the data flows, sniffing the Non-Access Stratum (NAS) signaling (which is the signaling between the UE and the Core Network and the RAN is supposed to be transparent to), policy enforcements, regular packet forwarding functions, collection of flow statistics into Flow Information Base (FIB), and encryption/decryption functions.
>
> Above the kernel, there are **several tools 416**, native daemons 418, native libraries 420 and Hardware Abstraction Layers (HAL) 422. **These tools abstract out the hardware dependencies for the upper layers and programmers**. The Runtime 424 is an embedded virtual machine capable of securely isolating and executing the applications. System Services 426 are a set of services always running and available to the developers (e.g., timing and messaging services).

Id. at 11:16-12:26 (emphasis added).

Further regarding the recitation of "independent of network device hardware and cloud device hardware," the specification of the '823 Patent also states on several occasions that "The dSDN also harmonizes application/fxApp deployment across the network independent of the hardware vendor." *Id.* at 2:35-37 and 7:61-62 (*see also id.* at Abstract).

Accordingly, the Enabling Clause as recited in claims 1 and 19 should be interpreted as encompassing that the implementation of the network device applications and cloud applications in a "unified manner" (interpreted as discussed above) allows for network device applications and cloud device applications to be programmed via APIs. Geier Decl. (Ex. 1003) at ¶¶ 26-28. Additionally, the Enabling Clause should be interpreted as encompassing that the aforementioned APIs are "upper layer" APIs in that the APIs constitute or interact with software layers that are abstracted out from (or only indirectly in contact with) a hardware-interactive layer such as a kernel

(or operating system), and therefore independent of (or removed from) any particular hardware on which the network device applications and cloud applications are implemented. Geier Decl. (Ex. 1003) at ¶28.

D. Construction of Functional Language

Finally, Requester notes that each of the claims of the '823 Patent are system claims (and thus appratus claims). However, several of the claims include functional limitations as set forth below:

- "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" in claims 1 and 19.
- "an application management portal capable of managing life cycles of the distributed applications" in claim 3.
- "said infrastructure marketplace capable of providing tested and certified distributed applications to the application management portal" in claims 3 and 19.
- "the application management portal is capable of at least one of the group consisting of: receiving new applications from the infrastructure application marketplace; testing said distributed applications prior to deployment; provisioning said distributed applications; and deprovisioning said distributed applications" in claim 4.
- "an application management portal capable of managing life cycles of the plurality of network device applications and plurality of cloud applications" in claim 19.

When interpreting the patent claims, functional language in apparatus claims should be identified, as the analysis of whether a prior art reference anticipates or renders obvious a given claim can take into account additional considerations when the claim language is functional language rather structural language. In this regard, the Section 2114 of the MPEP (in subpart II thereof, entitled "MANNER OF OPERATING THE DEVICE DOES NOT DIFFERENTIATE APPARATUS CLAIM FROM THE PRIOR ART") states as follows:

> "[A]pparatus claims cover what a device *is*, not what a device *does*." *Hewlett-Packard Co.v.Bausch & Lomb Inc.*, 909 F.2d 1464, 1469, 15 USPQ2d 1525, 1528 (Fed. Cir. 1990) (emphasis in original). A claim containing a "recitation with respect to the manner in which a claimed apparatus is intended to be employed does

> not differentiate the claimed apparatus from a prior art apparatus" if the prior art apparatus teaches all the structural limitations of the claim. Ex parte Masham, 2 USPQ2d 1647 (Bd. Pat. App. & Inter. 1987)….

MPEP §2114. Additionally, Section 2114 of the MPEP also states (in subpart I thereof, entitled "INHERENCY AND FUNCTIONAL LIMITATIONS IN APPARATUS CLAIMS") as follows:

> Features of an apparatus may be recited either structurally or functionally. *In re Schreiber*, 128 F.3d 1473, 1478, 44 USPQ2d 1429, 1432 (Fed. Cir. 1997). See also MPEP § 2173.05(g). If an examiner concludes that a functional limitation is an inherent characteristic of the prior art, then to establish a prima case of anticipation or obviousness, the examiner should explain that the prior art structure inherently possesses the functionally defined limitations of the claimed apparatus. *In re Schreiber*, 128 F.3d at 1478, 44 USPQ2d at 1432. *See also Bettcher Industries, Inc. v. Bunzl USA, Inc.*, 661 F.3d 629, 639-40, 100 USPQ2d 1433, 1440 (Fed. Cir. 2011). The burden then shifts to applicant to establish that the prior art does not possess the characteristic relied on. *In re Schreiber*, 128 F.3d at 1478, 44 USPQ2d at 1432; *In re Swinehart*, 439 F.2d 210, 213, 169 USPQ 226, 228 (CCPA 1971) ("where the Patent Office has reason to believe that a functional limitation asserted to be critical for establishing novelty in the claimed subject matter may, in fact, be an inherent characteristic of the prior art, it possesses the authority to require the applicant to prove that the subject matter shown to be in the prior art does not possess the characteristic relied on").

In view of this discussion from the Section 2114 of the MPEP, two principles are clear: (1) if a prior art reference teaches the structural limitations of an apparatus claim, functional language in that apparatus claim cannot serve as a basis for distinguishing that claim from the prior art; and (2) if a claimed functional limitation is an inherent characteristic of a prior art structure disclosed by a prior art reference, then the prior art reference can anticipate the claim even if the prior art reference does not expressly disclose that functional limitation. For purposes of this Request, Requester does not solely rely on the possibility that the limitations identified above are functional (and therefore not adding patentable weight), because the prior art presented here discloses, or renders obvious, every element of the claims regardless. *See* Appendix A and Appendix B. Nonetheless, to the extent the Office agrees that these limitations do not serve to differentiate the claims from the prior art, Requester raises this issue as potentially allowing for a quicker and simpler resolution of the proceeding.

III. LEVEL OF SKILL OF A PERSON OF ORDINARY SKILL IN THE ART

The prior art rejections proposed in this Request are based on anticipation under 35 U.S.C. §102(a) and obviousness under 35 U.S.C. §103(a) in light of the knowledge of a person of ordinary skill in the art (a "POSITA") at or around June 13, 2013, the priority date of the '823 Patent. '823 Patent (Ex. 1001) at cover sheet. Requesters expert, James T. Geier, has opined as to the level of ordinary skill in the art. Geier Decl. (Ex. 1003) at ¶¶ 31-33.

In determining the characteristics of a POSITA, several factors have been considered, including the type of problems encountered in the art, the solutions to those problems, the rapidity with which innovations are made in the field, the sophistication of the technology, and the education level of active workers in the field. Geier Decl. (Ex. 1003) at ¶ 31. The level of ordinary skill in the art that one would need in order to have the capability of understanding the scientific and engineering principles applicable to the '823 Patent is (i) a Bachelor degree (or higher degree) in electrical engineering, computer science, or computer engineering or equivalent and (ii) at least two years of experience working in network architecture and solutions, communication systems, wireless networks, or mobile wireless devices. *Id.* at ¶ 32. Additional industry experience or technical training may offset less formal education, while advanced degrees or additional formal education may offset lesser levels of industry experience. *Id.*

IV. STATEMENT POINTING OUT EACH SUBSTANTIAL NEW QUESTION OF PATENTABILITY FOR THE CHALLENGED CLAIMS

A. Listing of Prior Art Patents and Printed Publications

As explained in the introduction, the earliest possible priority date for the '823 Patent is June 13, 2013. Accordingly, reexamination of the Challenged Claims is requested in view of the following prior art references:

- **Ex. 1004** U.S. Pub. No. 2012/0278454 to Stewart et al. (Stewart) was filed as U.S. Patent Application No. 13/450,625 on April 19, 2012, published on November 1, 2012, and issued on January 22, 2019 as U.S. Patent No. 10,187,494. Stewart is prior art to the '823 Patent at least pursuant to 35 U.S.C. §102(a)(1) and (2).
- **Ex. 1005** U.S. Pub. No. 2013/0054682 to Malik et al. (Malik) was filed as U.S. Patent Application No. 13/598256 on August 29, 2012, published on February 28,

2013, and issued on May 19, 2015 as U.S. Patent No. 9,034,642. Malik is prior art to the '823 Patent at least pursuant to 35 U.S.C. §102(a)(1) and (2).

None of the references listed above were cited during prosecution of the '823 Patent. A Form SB-08 and copies of the cited references are submitted herewith.

As explained more fully below and in Appendices A and B, Stewart anticipates (Ground 1) or renders obvious (Ground 2) each of claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent, and Malik anticipates (Ground 3) or renders obvious (Ground 4) each of claims 1, 2, 3, 4, 15 and 19 of the '823 Patent.

Requester submits that the prior art references each raise a "substantial new question of patentability" because "the teaching of the (prior art) patents and printed publications is such that a reasonable examiner would consider the teaching to be important in deciding whether or not the claim is patentable." MPEP §2242. For example, the references and knowledge of a POSITA are discussed generally below and in further detail in comparison to the Challenged Claims in the attached charts (Appendix A and Appendix B). Each of the cited references, when considered alone or in combination with the knowledge of a POSITA, teaches each limitation of the Challenged Claims.

Further, none of these references were previously considered. Accordingly, the "same question of patentability as to the claim has not been decided by the Office in an earlier concluded examination or review of the patent" MPEP §2242, at least because none of the art referenced in this Request was before the Office during prosecution of the '823 Patent or during a prior post-grant proceeding challenging any claim of the '823 Patent, any of which that are known to Requester have been listed below in Section VI, infra.

B. Ground 1: Stewart Anticipates the Challenged Claims

The Stewart reference is discussed generally below and in detail in comparison to claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent in Appendix A. Requester submits that Stewart raises a "substantial new question of patentability" because, as discussed below and in Appendix A, Stewart teaches each limitation of claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent, including the concept of a system including a plurality of network device applications and the plurality of cloud applications that are in secure communication with each other to form distributed applications. For example, Stewart discloses a system that includes a gateway device (network device) in secure communication with a cloud server, where the gateway device installs (and thus hosts)

applications provided by a cloud server in order to manage various devices coupled to the gateway device. *See* Stewart (Ex. 1004) at Abstract and ¶¶ [0010], [0012], [0042], [0051], [0085], [0227]-[0230]. Stewart further discloses that the applications described therein may be distributed applications (including portions that execute on the cloud server and portions that execute on the gateway device). *See* Stewart (Ex. 1004) at ¶¶ [0016], [0111], [0126]. Because Stewart discloses each limitation of claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent, Stewart anticipates each of those claims.

1. Overview of Stewart

Stewart "generally relates to software application development and deployment on remote devices, and more specifically to systems and methods for configuring or managing gateway devices at various locations." Stewart (Ex. 1004) at ¶[0002]. FIGS. 1A and 1B of Stewart, produced below, illustrate various exemplary systems of Stewart. *Id.* at ¶[0043].

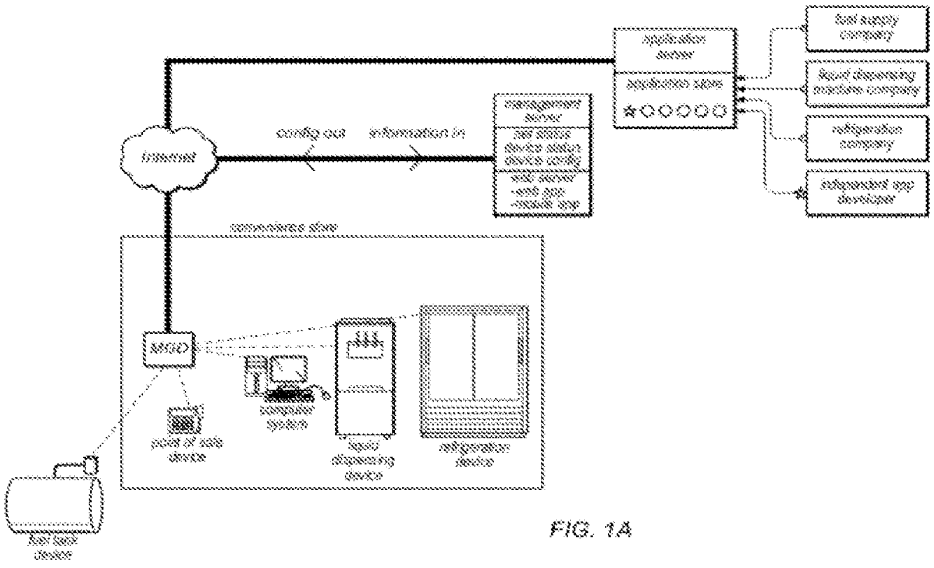


FIG. 1A

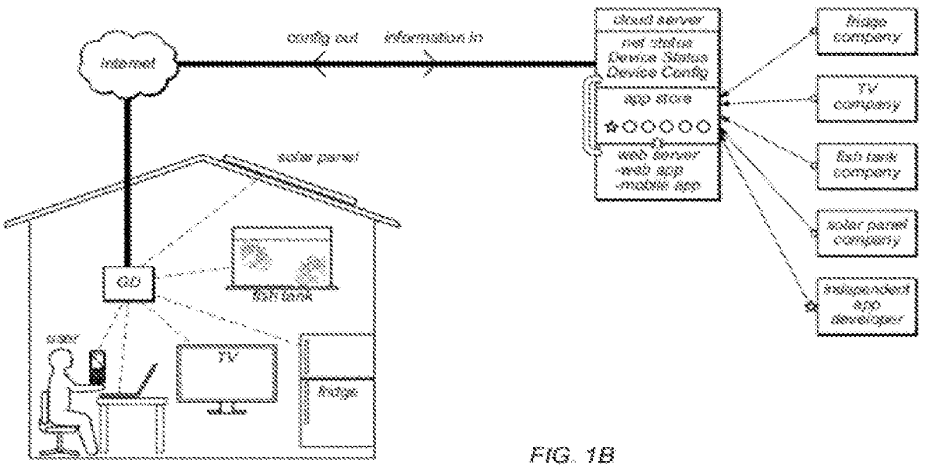


FIG. 1B

The systems of Stewart as shown in Figs. 1A and 1B are very similar, except that the system of Fig. 1A is implemented in a merchant location and the system of Fig. 1B is implemented in a home. *Id.* at ¶[0044] and [0076]-[0078].

Stewart discloses that its systems include a gateway device (MGD in Fig. 1A and GD in Fig. 1B), which is a device that is "configured to couple to a wide area network, such as the Internet, and also couple (either wired or wirelessly) to one or more devices, and which is configurable with one or more application programs to monitor and/or control the one or more devices." *Id.* at ¶ [0042]. Stewart further discloses that the gateway device may install (and thus host) applications that may be provided by a cloud server in order to manage various devices coupled to the gateway device. Id. at Abstract.

Stewart discloses that the gateway device is in communication with one or more servers, which may be a cloud server or cloud servers. *Id.* at ¶¶[0054]-[0055] and [0078]. The one or more servers include a management server and an application server, which may be the same server or different servers, and which are illustrated in Fig. 1A as being separate servers and in Fig. 1B as being provided by a single cloud server. *Id.* at ¶¶[0068] and [0078].

Various different application developers (or "package developers") may create various different applications that may be uploaded to the application server for purchase by customers (e.g., merchants) who are operating gateway devices. *Id.* at ¶[0012]. The applications may be

executed in a distributed fashion, e.g., among the gateway device, the server, and/or other devices, such as user devices. *Id.* at ¶[0016].

"A user may be able to configure the gateway device directly (e.g., via a web site hosted by the gateway device or via an application executing on the computer system) or may be able to configure the gateway device over the Internet, via the management server." *Id.* at ¶[0054]. "For example, a manager may browse the application server and select from a plurality of different applications that may be installed and executed by the gateway device and/or the management server (and possibly by devices associated with the convenience store). *Id.* at ¶[0069].

Stewart discloses that a software development kit (SDK) and/or application programming interface (API) may be used to allow a developer to develop an application to be stored on the application server that is compatible with the gateway device, the management server, and/or other devices that may execute portions of the application. *Id.* at ¶¶[0074] and [0105]-[0106].

Additionally, with reference to Fig. 6, reproduced below, Stewart discloses that APIs can also be used in relation to developed applications. Specifically, Fig. 6 of Stewart shows a tank application that is configured to determine the fuel level in the fuel tank and present the fuel level information in a user interface. *Id.* at ¶¶[0111]. The portion of the tank application executing on the gateway device receives information from the fuel tank using a communication protocol supported by the fuel tank device (e.g., the device monitoring the fuel level in the fuel tank). *Id.* at ¶¶[0112]. The gateway device is configured to receive the reported information from the tank application and provide that information for storage via an internal API. *Id.* Additionally, the tank application is configured to provide the reported information for storage, e.g., in cloud storage on the Internet, and executes an external API to perform that functionality. *Id.* at ¶¶[0112].

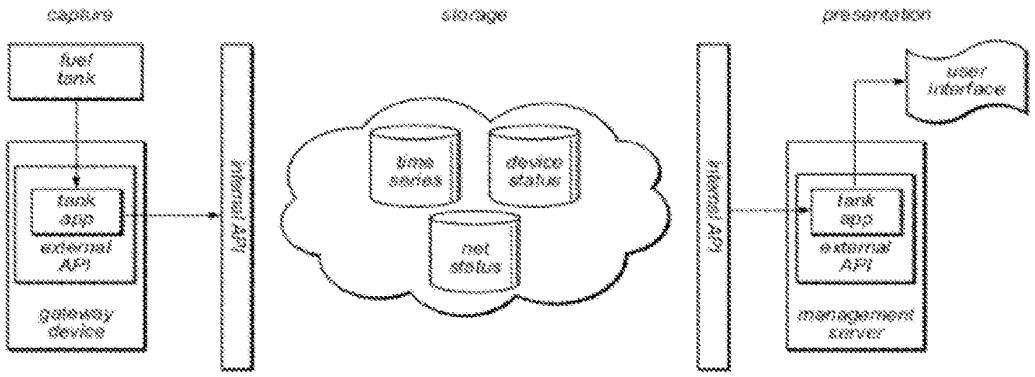


FIG. 6

Further, a second portion of the tank application may execute on the management server, and may use a presentation external API to request information stored in the cloud storage. *Id.* at ¶¶[0113]. "Thus, FIG. 6 illustrates an exemplary embodiment of a distributed application using internal and external APIs." *Id.*

Stewart is analogous art to the '823 Patent. The field of endeavor of the '823 Patent relates to a network architecture that facilitates "secure and flexible programmability between a user device and across a network…" '823 Patent (Ex. 1001) at Abstract; Geier Decl. (Ex. 1003) at ¶ 54. Further, the '823 Patent relates to a network architecture that includes one or more cloud device (e.g., a cloud server) that interact with one or more network devices. See e.g., '823 Patent (Ex. 1001) at Abstract and at 10:9-13; Geier Decl. (Ex. 1003) at ¶ 54. The '823 Patent relates to network technology that "is not limited to wireless networks embodiments" but rather "could be applied to many other network types including enterprise, wireline service providers, data centers, and Internet of Things (IoT)." '823 Patent (Ex. 1001) at 7:67-8:3; Geier Decl. (Ex. 1003) at ¶ 54.

Stewart similarly relates to networks, and for example provides disclosure regarding how a "gateway device may also be coupled to a server (e.g., a cloud server) over a wide area network…." Stewart (Ex. 1004) at ¶¶[0009]; Geier Decl. (Ex. 1003) at ¶ 55. Stewart also addresses the programming of devices on a network, stating for example that a "default program configuration may include a management application that may be installed and executed by the gateway device and/or the server to allow users to interact with the gateway device. …." Stewart (Ex. 1004) at ¶¶[0139]; Geier Decl. (Ex. 1003) at ¶ 55. Because Stewart relates to networks with

involve both cloud device(s) and other network device(s) as well as programming of such devices, Stewart is within the field of endeavor of the '823 Patent. Geier Decl. (Ex. 1003) at ¶ 55.

Further, Stewart is reasonably pertinent to at least one problem considered by the inventor of the '823 Patent. Geier Decl. (Ex. 1003) at ¶ 56. The '823 Patent concerns addressing problems associated with the inflexibility of configuring a network, for example, stating that "[e]xisting network solutions are built on proprietary hardware and software" and additionally that "[n]etwork Operators & Information Technology (NOIT) can only configure the proprietary network infrastructure provided by vendors and are unable to add new and customized features or capabilities." '823 Patent (Ex. 1001) at 1:22-26; Geier Decl. (Ex. 1003) at ¶ 56. Stewart describes a networked system that achieves enhanced configuration of devices on the network, for example, describing how a "management server may also allow for configuration of the gateway device or premise devices managed by the gateway device in the convenience store. Stewart (Ex. 1004) at ¶¶[0066]; Geier Decl. (Ex. 1003) at ¶ 56. Stewart further describes a network system having a framework that may allow "for easy installation of applications in response to user selection of applications to install." Stewart (Ex. 1004) at ¶¶[0070]; Geier Decl. (Ex. 1003) at ¶ 56. Stewart further recognizes that, in at least some embodiments, it is significant that configurations be achieved that are not tied to specific devices, stating for example that "while the above described applications are specific to individual types (or even models) of premise devices in the convenience store, there may be applications that are not device specific." Stewart (Ex. 1004) at ¶¶[0075]; Geier Decl. (Ex. 1003) at ¶ 56. Thus, Stewart closely relates to the aforementioned problem identified by the '823 Patent, in that Stewart concerns network systems that, in at least some embodiments, are designed to facilitate flexible configuration of devices associated with the network and, in at least some circumstances in a manner that is independent of specific characteristics of those devices. Geier Decl. (Ex. 1003) at ¶ 56.

C. Ground 2: Stewart Renders Obvious the Challenged Claims

While Requester's primary position regarding Stewart is that Stewart anticipates every limitation of the Challenged Claims (Ground 1), to the extent that "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" as recited in independent claims 1 and 19 is given any patentable weight and is not expressly or inherently taught by Stewart, Stewart renders this limitation obvious. The

Stewart reference is discussed generally in Section IV(B)(1) above and the knowledge of a POSITA is discussed generally in Section IV(C)(1) below. Stewart as it would be understood in view of the knowledge of a POSITA is discussed generally in Section IV(C)(2) below and in detail in comparison to claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent in Appendix A. As discussed there, Stewart renders obvious each limitation of claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent.

1. Overview of the Knowledge of a POSITA at the time of the '823 Patent

The use of application programming interfaces (APIs) to program applications independently of device hardware has been known for decades, and well before the time of the '823 Patent in June of 2013. Geier Decl. at ¶ 102. Generally, APIs provide a set of rules or protocols that enable software applications to communicate with each other to exchange data, features and functionality independently of the hardware implementation of those applications. *Id.* at ¶ 103. The term "API" was introduced at least as early as 1974, in a paper called "The Relational and Network Approaches: Comparison of the Application Programming Interface." *Id.* at ¶ 104. With the growth of the Internet in the 1990's, APIs were used as a way for applications to exchange data. *Id.* at ¶105. Prior to May 30, 1995, Microsoft had developed a software model referred to as WOSA, which was defined for use in the Windows programming environment. The WOSA model isolated "application programmers from the complexities of programming to different service providers by providing an API layer that is independent of an underlying hardware or service." *Id.* (quoting U.S. Patent No. 5,691,897 at 2:55-58).

Accordingly, the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware) was well within the knowledge of a POSITA prior to the June 13, 2023 priority date of the '823 Patent. *Id.* at ¶ 106.

2. Stewart in View of the Knowledge of a POSITA

As discussed in Section IV(B)(1) above and in detail in Appendix A, Stewart anticipates claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent. For at least the same reasons, Stewart renders obvious each and every limitation of claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent. However, to the extent that Stewart is deemed not to disclose the Enabling Clause either expressly or inherently for purposes of anticipation, and that clause is given patentable weight despite being

functional language, Stewart, as it would be understood in view of the knowledge of a POSITA as of June 13, 2023, renders that limitation obvious. Geier Decl. at ¶¶ 107-109.

As discussed in detail in Appendix A, Stewart explicitly discloses the use of APIs *to program the plurality of network device applications and plurality of cloud applications* as recited in independent claims 1 and 19 of the '823 Patent. Appendix A at pp. 10-16. However, Stewart does not explicitly state that the APIs are *upper layer* APIs or that the programming is *independent of network device hardware and cloud device hardware*. To the extent that those elements are not deemed to be inherent, they would have been obvious based on the teachings of Stewart in view of the knowledge of a POSITA as of June 13, 2023. Geier Decl. at ¶ 108. Specifically, as discussed in Section IV(C)(1) above, a POSITA as of June 13, 2023 would have known about and understood the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware). *Id.* at ¶ 109. Accordingly, a POSITA as of June 13, 2023 would have understood that the APIs of Stewart are "upper layer" APIs within the meaning of the '823 Patent, and that they would be enabled "to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" as recited in independent claims 1 and 19 of the '823 Patent. *Id.* at ¶ 109.

D. Ground 3: Malik Anticipates Claims 1, 2, 3, 4, 15 and 19

The Malik reference is discussed generally below and in detail in comparison to claims 1, 2, 3, 4, 15 and 19 of the '823 Patent in Appendix B. Requester submits that Malik raises a "substantial new question of patentability" because, as discussed below and in Appendix B, Malik teaches each limitation of claims 1, 2, 3, 4, 15 and 19 of the '823 Patent, including the concept of a system including a plurality of network device applications and the plurality of cloud applications that are in secure communication with each other to form distributed applications. For example, Malik discloses agents on endpoint devices "for downloading updates in a secure manner from any suitable server resources." Malik at ¶[0047]. Malik also discloses that its MDM system can be implemented with a plurality of servers hosting the applications or components thereof, resulting in distributed applications, to provide scalability. *Id.* at ¶¶[0038], [0057], and [0062]. Malik further discloses that software components that reside on an endpoint gather data that is then uploaded to the server components, thus disclosing applications distributed between

the endpoint and the server. *Id.* at ¶¶[0062] and [0087]. Because Malik discloses each limitation of claims 1, 2, 3, 4, 15 and 19 of the '823 Patent, Malik anticipates each of those claims.

1. Overview of Malik

Malik generally relates to "methods and systems for providing mobile device management services and more specifically providing deployment services for deploying software modules to a plurality of managed endpoints." Malik (Ex. 1005) at ¶[0002]. Malik teaches a mobile device management (MDM) platform that can be used to allow an organization to selectively add or remove applications/feature modules on-demand. Malik (Ex. 1005) at ¶[0030]. Malik teaches that "deployment of applications and data to endpoints can be facilitated by a network-based solution" using "a cloud-based approach". Malik (Ex. 1005) at ¶[0030]. With reference to Fig. 1 of Malik, reproduced below:

> FIG. 1 shows a high level architecture of the deployment and distribution system as part of an MDM platform. System 100 contains server components and endpoint components. An application server 102 provides server functionality as described herein. This application server can be a SaaS application server providing server functionality in a cloud-based approach, which include a plurality of servers hosting the application or components thereof. Agents 104 on endpoints 105 provide core MDM functionality including allowing actions to be executed on the endpoints and allowing data and status information to be returned to the server. Agents 104 can include a plurality of endpoint applications that provide agent functionality. SaaS application server 102 may transmit actions and modules across a link 106, which may traverse a network, such as the Internet. Agents 104 may receive these modules and actions and install modules or execute actions accordingly. Agents 104 can also transmit data to the application server across link 108 which may also traverse a network, such as the Internet. This allows agents to report back status of deployments and data collected or generated by the deployed feature modules to application server 102.

Malik (Ex. 1005) at ¶[0057].

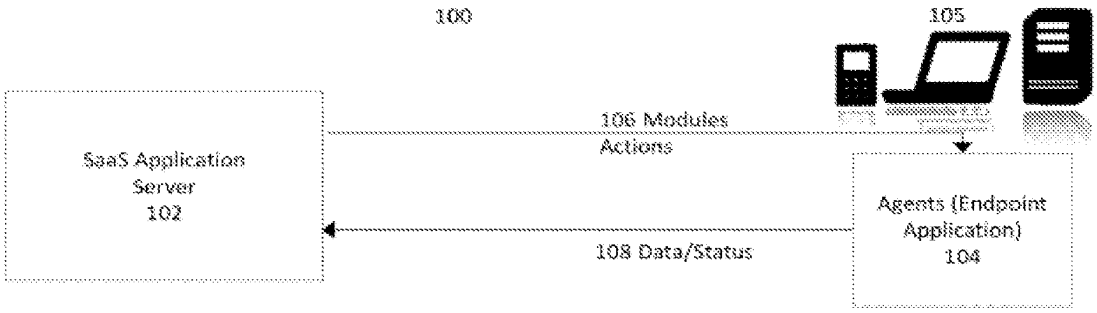


FIG. 1

The endpoint devices in Malik "can include any endpoint devices suitable for accessing or interacting with corporate resources" such as "PCs, laptops, mobile devices, smart phones, tablets, network clients, or the like." Malik (Ex. 1005) at ¶[0035]. Endpoint devices have a processor, memory, and a communication circuit to allow communication with a network, either wirelessly or via a network cable. *Id.* Modules and software components, such as agents, are stored in non-transient memory on the endpoint. *Id.*

Figures 2 and 3 of Malik, reproduced below illustrate system and flow diagrams of "exemplary server components for deploying feature modules to endpoint devices" (Fig. 2) and "exemplary client components for deploying feature modules to endpoint devices" (Fig. 3).

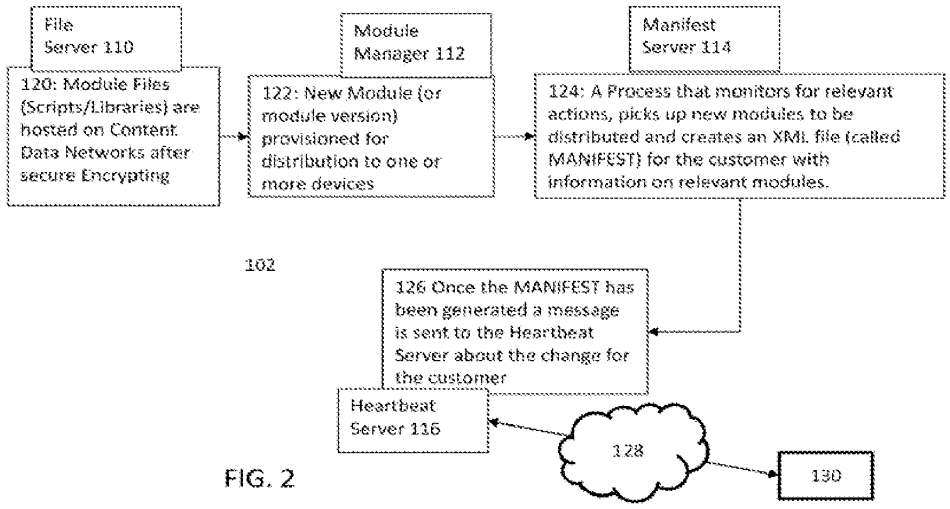


FIG. 2

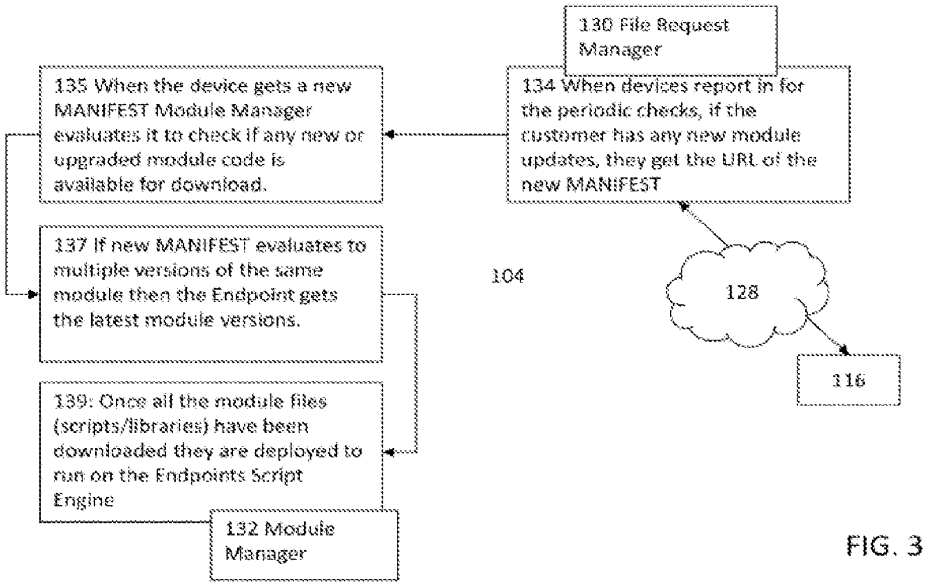


FIG. 3

As shown in Fig. 2, the server of Malik includes "a file server configured to host a plurality of feature modules, a module manager configured to maintain the plurality of feature modules, and a manifest server configured to maintain a plurality of manifests, wherein each manifest conveys the feature module components that at least one endpoint should have installed." Malik (Ex. 1005) at ¶[0012]. As shown in Figs. 2 and 3, "in response to a changed module component identified in the manifest, module manager 132 (residing on the endpoint device) can communicate with file server 110 across a network, such as Internet, to download these modules and utilize encryption keys stored in the manifest to decrypt the contents." *Id.* at ¶[0078].

Malik also discloses that its MDM system can be implemented with a plurality of servers hosting the applications or components thereof, resulting in distributed applications, to provide scalability. Malik at ¶¶[0038], [0057], and [0062]. Malik further discloses that software components that reside on and endpoint gather data that is then uploaded to the server components, thus disclosing applications distributed between the endpoint and the server. Malik at ¶¶[0062] and [0087].

Malik teaches that developers of feature modules can take advantage of APIs as part of Malik's MDM platform, to allow reporting of information gathered by feature modules. Malik at

¶¶ [0084], [0088], and Fig. 4. Malik further discloses use of a web based dashboard and also that developers can define and configure modules for deployment by use of a dashboard. Malik at ¶¶[0094] and [0099].

Malik is analogous art to the '823 Patent. The field of endeavor of the '823 Patent relates to a network architecture that facilitates "secure and flexible programmability between a user device and across a network…" '823 Patent (Ex. 1001) at Abstract; Geier Decl. (Ex. 1003) at ¶116. Further, the '823 Patent relates to a network architecture that includes one or more cloud device (e.g., a cloud server) that interact with one or more network devices. See e.g., '823 Patent (Ex. 1001) at Abstract and at 10:9-13; Geier Decl. (Ex. 1003) at ¶ 116. The '823 Patent relates to network technology that "is not limited to wireless networks embodiments" but rather "could be applied to many other network types including enterprise, wireline service providers, data centers, and Internet of Things (IoT)." '823 Patent (Ex. 1001) at 7:67-8:3; Geier Decl. (Ex. 1003) at ¶116. Malik similarly relates to networks, for example, as shown in Figure 1 thereof, which shows a system that "contains server components and endpoint components." Malik (Ex. 1005) at ¶¶[0057]; Geier Decl. (Ex. 1003) at ¶116. Malik also addresses the programming of devices on a network, stating for example that the "deployment of applications and data to endpoints can be facilitated by a network-based solution" and that, "[i]n some embodiments, a cloud-based approach is used…." Malik (Ex. 1005) at ¶¶[0030]; Geier Decl. (Ex. 1003) at ¶ 116. Because Malik relates to networks involving server components and other components and that involve a cloud-based approach, as well as relates to the programming of network devices, Malik is within the field of endeavor of the '823 Patent. Geier Decl. (Ex. 1003) at ¶ 116.

Further, Malik is reasonably pertinent to at least one problem considered by the inventor of the '823 Patent. Geier Decl. (Ex. 1003) at ¶ 117. The '823 Patent concerns addressing problems associated with the inflexibility of configuring a network, for example, stating that "[e]xisting network solutions are built on proprietary hardware and software" and additionally that "[n]etwork Operators & Information Technology (NOIT) can only configure the proprietary network infrastructure provided by vendors and are unable to add new and customized features or capabilities." '823 Patent (Ex. 1001) at 1:22-26; Geier Decl. (Ex. 1003) at ¶ 117. As already discussed above, Malik relates to a network system having "server components and endpoint components." Malik (Ex. 1005) at ¶¶[0057]; Geier Decl. (Ex. 1003) at ¶ 117. Further, Malik discloses a system that is directed toward the deployment of applications/feature modules, where

the system particularly includes agents that are "installed on endpoint devices, where the "agents can facilitate installation of applications/feature modules on-demand." Malik (Ex. 1005) at ¶¶[0030]; Geier Decl. (Ex. 1003) at ¶ 117. Malik further describes at least some embodiments the envision the implementation of genericized applications, stating for example that in some embodiments "certain feature modules may utilize the reporting mechanisms shown in FIGS. 4-5 to report back installation status, allowing feature modules include generic applications that may or may not be related to MDM functionality" and also that in some embodiments "feature modules may wrap basic reporting features around non-MDM applications to allow basic reporting of status of the nation and installation to server components." Malik (Ex. 1005) at ¶¶[0084]; Geier Decl. (Ex. 1003) at ¶ 117. Thus, Malik closely relates to the aforementioned problem identified by the '823 Patent, in that Malik concerns network systems that, in at least some embodiments, are designed to facilitate flexible configuration of devices associated with the network and, in at least some circumstances in a genericized manner. Geier Decl. (Ex. 1004) at ¶ 117.

E. Ground 4: Malik Renders Obvious Claims 1, 2, 3, 4, 15 and 19

While Requester's primary position regarding Malik is that Malik anticipates every limitation of the Challenged Claims (Ground 3), to the extent that "enabling a plurality of upper layer application programming interfaces (APIs) to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" as recited in independent claims 1 and 19 is given any patentable weight and is not expressly or inherently taught by Malik, Malik renders this limitation obvious. The Malik reference is discussed generally in Section IV(D)(1) above and the knowledge of a POSITA is discussed generally in Section IV(C)(1) above. As discussed below, Malik as it would be understood in view of the knowledge of a POSITA renders obvious each limitation of claims 1, 2, 3, 4, 15 and 19 of the '823 Patent.

As discussed in Section IV(D)(1) above and in detail in Appendix B, Malik anticipates each and every limitation of claims 1, 2, 3, 4, 15 and 19 of the '823 Patent. For at least the same reasons, Malik renders obvious each and every limitation of those claims. However, to the extent that Malik is deemed not to disclose the Enabling Clause either expressly or inherently, and that clause is given patentable weight despite being functional language, Malik, as it would be understood in view of the knowledge of a POSITA as of June 13, 2023, renders that limitation obvious. Geier Decl. at ¶¶ 155-158. As discussed in detail in Appendix B, Malik explicitly

discloses the use of APIs to *program the plurality of network device applications and plurality of cloud applications* as recited in independent claims 1 and 19 of the '823 Patent. Appendix B at pp. 10-16. However, Malik does not explicitly state that the APIs are *upper layer* APIs or that the programming is *independent of network device hardware and cloud device hardware*. To the extent that those elements are not deemed to be inherent, they would be obvious based on the teaching of Malik in view of the knowledge of a POSITA as of June 13, 2023. Geier Decl. at ¶157. Specifically, as discussed in Section IV(C)(1) above, a POSITA as of June 13, 2023 would have known about and understood the functionality of APIs and their ability to allow programmers to program applications, and components thereof, and provide for data exchange, independently of device hardware (above the hardware or as upper layer APIs relative to the hardware). *Id.* at ¶ 158. Accordingly, a POSITA as of June 13, 2023 would have understood that the APIs of Malik are "upper layer" APIs within the meaning of the '823 Patent, and that they would be enabled "to program the plurality of network device applications and plurality of cloud applications independent of network device hardware and cloud device hardware" as recited in independent claims 1 and 19 of the '823 Patent. *Id.*

V. DETAILED APPLICATION OF THE PRIOR ART TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED

A. Proposed Rejections of Claims

This request presents two proposed anticipation rejections based on Stewart and Malik, respectively, as well as two proposed obviousness rejections based on Stewart and Malik, respectively.

Requester notes that the proposed rejections below are based on anticipation under 35 U.S.C. § 102(a) and obviousness under 35 U.S.C. § 103(a) in light of the knowledge of a person of ordinary skill in the art at or around June 13, 2013 (the priority date of the '823 Patent). With respect to the proposed obviousness rejections, when compared with the prior art described below and in light of the knowledge of a POSITA, any "the differences between the claimed invention and the prior art are such that the claimed invention as a whole would have been obvious before the effective filing date of the claimed invention to a person having ordinary skill in the art to which the claimed invention pertains." 35 U.S.C. § 103.

The table below summarizes the grounds of unpatentability presented in this Request. Requester submits that each ground below raises a substantial new question (SNQ) of patentability for at least one claim of the '823 Patent.

Ground	Proposed Rejection
1.	Stewart anticipates claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent under 35 U.S.C. §102
2.	Stewart renders obvious claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent under 35 U.S.C. §103
3.	Malik anticipates claims 1, 2, 3, 4, 15 and 19 of the '823 Patent under 35 U.S.C. §102
4.	Malik renders obvious claims 1, 2, 3, 4, 15 and 19 of the '823 Patent under 35 U.S.C. §103

The proposed rejections are described in detail in the attached claim charts to this Request. *See* Appendix A (Stewart-based grounds) and Appendix B (Malik-based grounds).

B. Secondary Considerations

This Request demonstrates that the Challenged Claims of the '823 Patent are unpatentable as obvious in view of the prior art references. The Applicant did not identify any evidence of secondary considerations during prosecution. Further, the clear teachings in the prior art are not outweighed by any supposed "secondary considerations." *Graham v. John Deere Co. of Kansas City*, 383 U.S. 1, 36 (1966).

VI. DISCLOSURE OF CONCURRENT LITIGATION, REEXAMINATION, AND RELATED PROCEEDINGS

Based on information available to Requester, the '823 Patent is the subject of the following currently pending District Court litigations, as listed below:

- Edge Networking Systems, LLC v. Microsoft Corporation, Case No. 1:24cv215 (W.D. Tex. Feb. 29, 2024).
- Edge Networking Systems, LLC v. Amazon.com, Inc., et al, Case No: 2:24cv887 (E.D. Tex. Nov. 1, 2024).

Additionally, the '823 Patent is the subject of the following *Inter Partes* Review proceedings:

- IPR 2025-00618 filed by Microsoft Corporation on February 17, 2025.

Requester is not aware of any other litigation, reexaminations or other post-grant proceedings in which the '823 Patent is or has been involved.

VII. CONCLUSION

For the foregoing reasons, it is clear that the accompanying prior art references show that there exists at least one substantial new question of patentability with respect to each of claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent. Accordingly, it is respectfully requested that *ex parte* reexamination of each of claims 1, 2, 3, 4, 8, 15 and 19 of the '823 Patent be ordered.

The Commissioner is hereby authorized to charge Deposit Account No. 50-6990 the *Ex Parte* Reexamination fee of $12,600 under 37 C.F.R. § 1.20(c)(2). Requester believes no other fee is due with this submission, however the Commissioner is hereby authorized to charge any fee deficiency or credit any over-payment to Deposit Account No. 50-6990.

Please direct all correspondence in this matter to the undersigned.

Respectfully submitted,

Dated: May 6, 2025

_/John T. Pienkos/__
John T. Pienkos
Registration No. 42,997
Ph: (414) 847-6170
E-mail: jpienkos@amundsendavislaw.com

Amundsen Davis, LLC
111 East Kilbourn Avenue
Suite 1400
Milwaukee, WI 53202

_/Jennifer Lacroix/__
Jennifer Lacroix
Registration No. 46,852
Ph: 312-455-3863
E-mail: jlacroix@amundsendavislaw.com

CERTIFICATE OF SERVICE

The undersigned attorney hereby certifies that on May 6, 2025, true and correct copies of each of the following were filed electronically with the USPTO:

- Request for *ex parte* Reexamination of U.S. Patent No. 11,695,823
- Appendix A - Claim Chart for Unpatentability of the '823 Patent (Stewart)
- Appendix B - Claim Chart for Unpatentability of the '823 Patent (Malik)
- Exhibit 1001 - U.S. Patent No. 11,695,823
- Exhibit 1002 - File History of the '823 Patent
- Exhibit 1003 - Declaration of James T. Geier
- Exhibit 1004 - U.S. Pub. No. 20120278454 to Stewart et al.
- Exhibit 1005 - U.S. Pub. No. 20130054682 to Malik et al.
- Request Transmittal Form
- Information Disclosure Statement

and that true and correct copies of each of the above listed documents have been served on the patent owner at the correspondence address of record by FedEx and e-mail to:

Roark IP
Attention: Christopher Horgan
1438 Dahlia Loop
San Jose, CA 95126
UNITED STATES
chris.horgan@roarkip.com

Respectfully submitted,

_/Jennifer Lacroix/__
Jennifer Lacroix
Registration No. 46,852
Ph: 312-455-3863
E-mail: jlacroix@amundsendavislaw.com

Amundsen Davis, LLC
111 East Kilbourn Avenue
Suite 1400
Milwaukee, WI 53202