US012387573B2

(12) **United States Patent**
Au et al.

(10) **Patent No.: US 12,387,573 B2**
(45) **Date of Patent: Aug. 12, 2025**

(54) **DEPLOYING IMMUTABLE IMAGE SOFTWARE VIA MEMORY PARTITION OF BRANCH RETAIL DEVICES**

(71) Applicant: **JPMorgan Chase Bank, N.A.**, New York, NY (US)

(72) Inventors: **Kin Au**, Lewis Center, OH (US); **Thomas Skinner**, Columbus, OH (US); **Steven Burd**, Columbus, OH (US); **Russell White**, Marysville, OH (US); **Kevin Sharb**, Lancaster, OH (US); **Chad Miller**, Ontario, OH (US); **Abhishek Mahajan**, Powell, OH (US)

(73) Assignee: **JPMORGAN CHASE BANK, N.A.**, New York, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 84 days.

(21) Appl. No.: **18/370,806**

(22) Filed: **Sep. 20, 2023**

(65) **Prior Publication Data**

US 2025/0069485 A1 Feb. 27, 2025

Related U.S. Application Data

(63) Continuation-in-part of application No. 18/236,357, filed on Aug. 21, 2023.

(51) **Int. Cl.**
G07F 19/00 (2006.01)

(52) **U.S. Cl.**
CPC ***G07F 19/211*** (2013.01); ***G07F 19/204*** (2013.01)

(58) **Field of Classification Search**
CPC G06F 8/63
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

9,910,678	B1 *	3/2018	Furyaev	H04L 12/42
11,321,107	B1 *	5/2022	Eyberg	G06F 9/45545
2004/0243998	A1 *	12/2004	Collins	G06F 8/60 717/178
2021/0224074	A1 *	7/2021	O'Dell	G06F 1/24

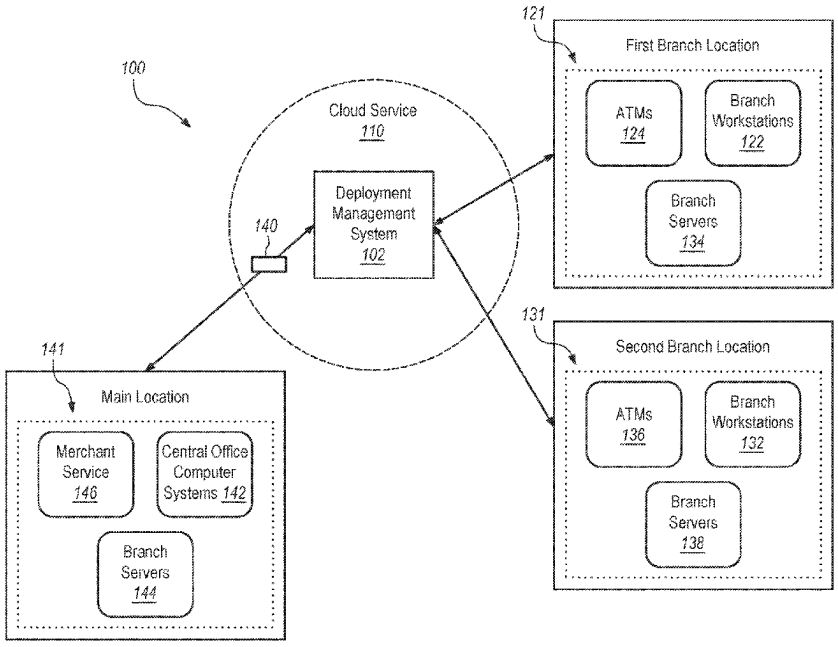
* cited by examiner

Primary Examiner — Zachary K Huson
(74) *Attorney, Agent, or Firm* — Greenblum & Bernstein, P.L.C.

(57) **ABSTRACT**

In one example, a storage medium with program instructions to deploy immutable-image software to retail branch devices is disclosed. The operations include accessing a change request from a user interface to initiate an event on a remote computer system. The event is a deployment of immutable branch-image software to alter or reimage existing software stored or executed on the remote computer system. The operations include validating that the deployment of the immutable branch-image software is required on the remote branch computer system and generating a new partition in a memory of the remote branch computer system. The operations also include retrieving the immutable branch-image software via a secure network socket in response to the event message, installing the immutable branch-image software in the new partition and transferring operation of the remote branch computer system from an original partition of the memory to the new partition.

20 Claims, 10 Drawing Sheets



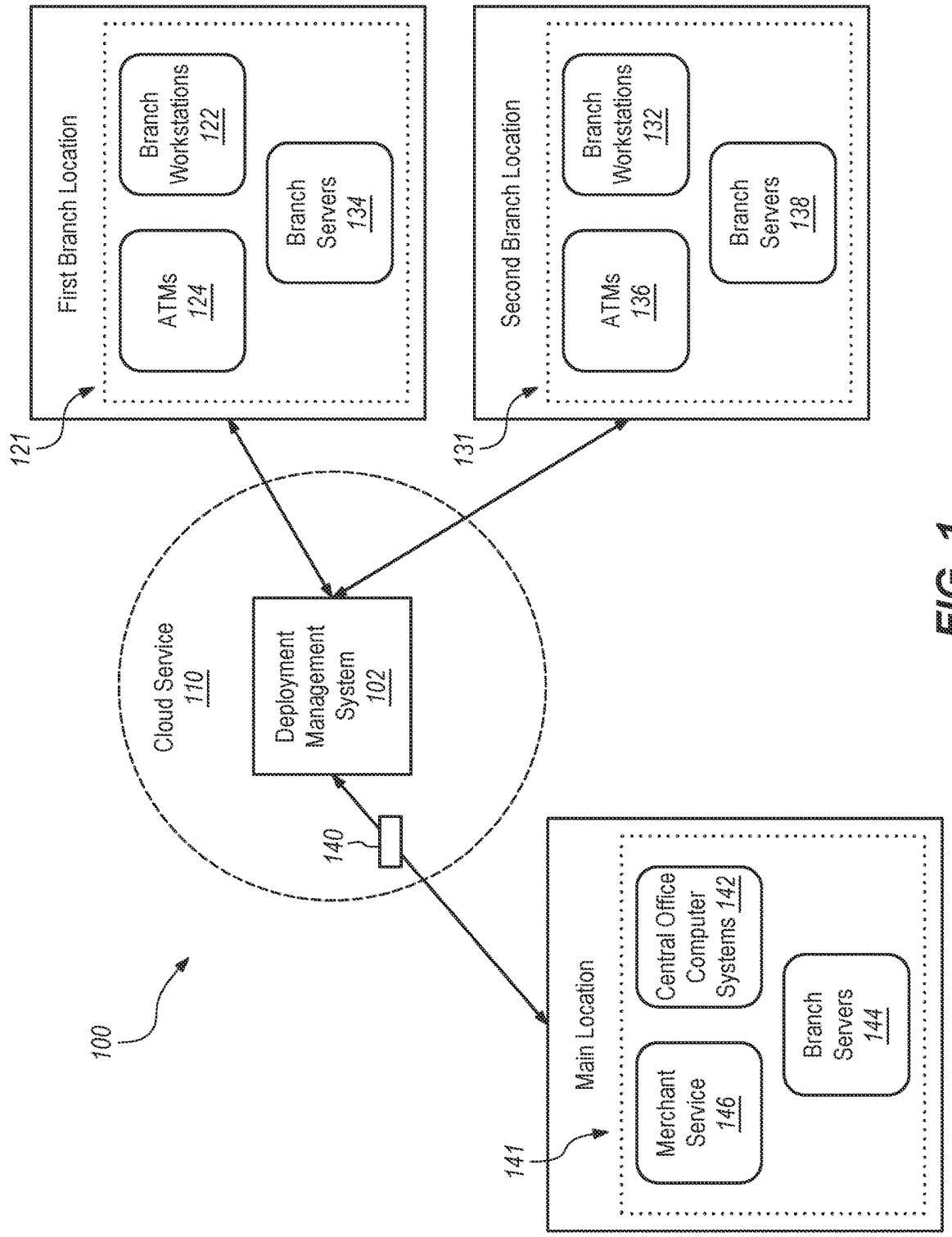

First Branch Location
ATMs 124
Branch Workstations 122
Branch Servers 134
121
Second Branch Location
ATMs 136
Branch Workstations 132
Branch Servers 138
131
Cloud Service 110
Deployment Management System 102
140
100
Main Location
Merchant Service 146
Central Office Computer Systems 142
Branch Servers 144
141

FIG. 1

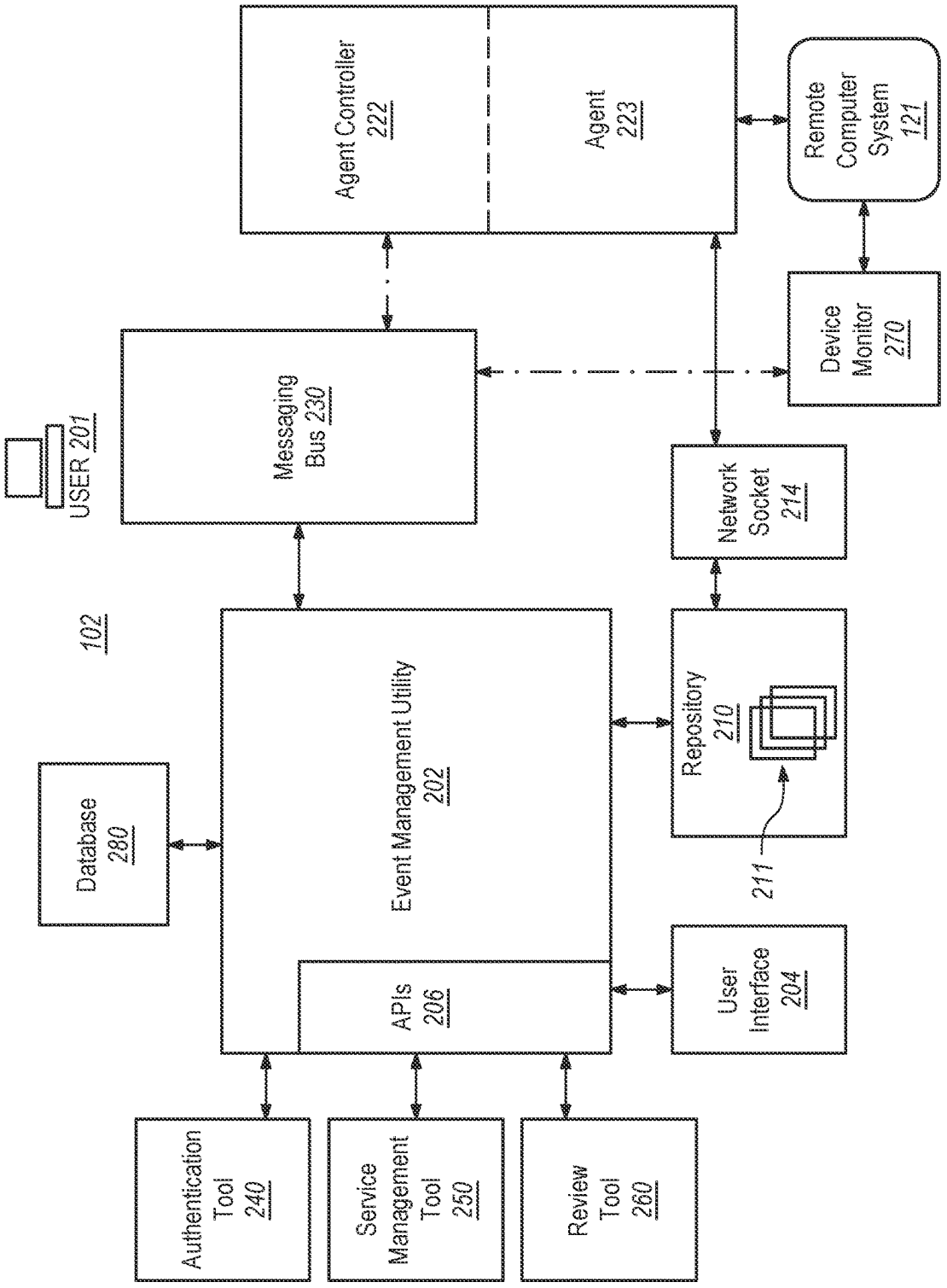

Agent Controller 222
Agent 223
Remote Computer System 121
Device Monitor 270
Messaging Bus 230
USER 201
Network Socket 214
102
Event Management Utility 202
Repository 210
211
Database 280
APIs 206
User Interface 204
Authentication Tool 240
Service Management Tool 250
Review Tool 260

FIG. 2

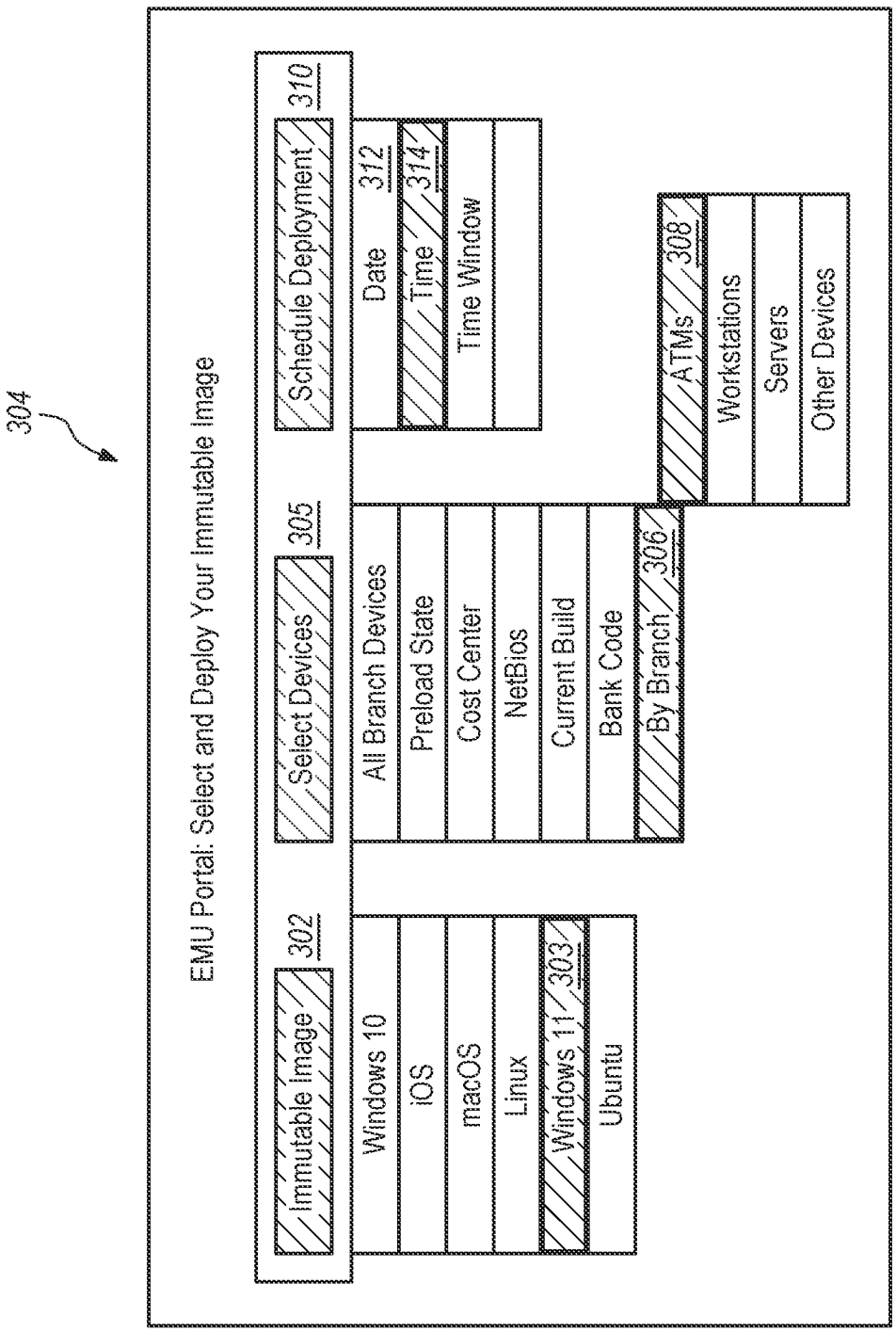

304
EMU Portal: Select and Deploy Your Immutable Image
Immutable Image 302
Windows 10
iOS
macOS
Linux
Windows 11 303
Ubuntu
Select Devices 305
All Branch Devices
Preload State
Cost Center
NetBios
Current Build
Bank Code
By Branch 306
Schedule Deployment 310
Date 312
Time 314
Time Window
ATMs 308
Workstations
Servers
Other Devices

FIG. 3

FIG. 4

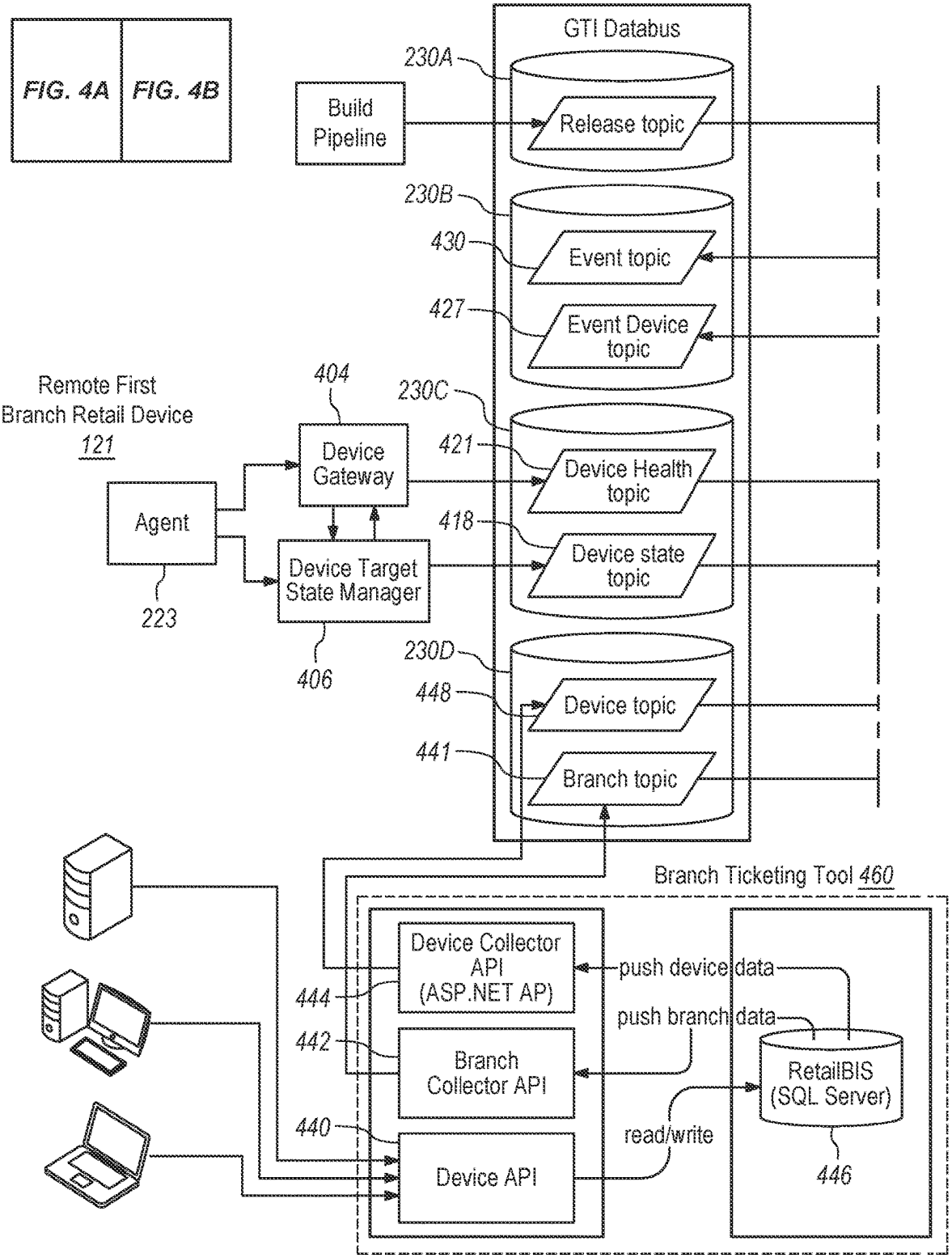


FIG. 4A
FIG. 4B
Databus 230
GTI Databus
230A
Build Pipeline
Release topic
230B
430
Event topic
427
Event Device topic
Remote First Branch Retail Device 121
404
230C
421
Device Gateway
Device Health topic
Agent
418
Device state topic
Device Target State Manager
223
406
230D
448
Device topic
441
Branch topic
Branch Ticketing Tool 460
Device Collector API (ASP.NET AP)
push device data
444
push branch data
442
Branch Collector API
RetailBIS (SQL Server)
440
read/write
Device API
446

FIG. 4A

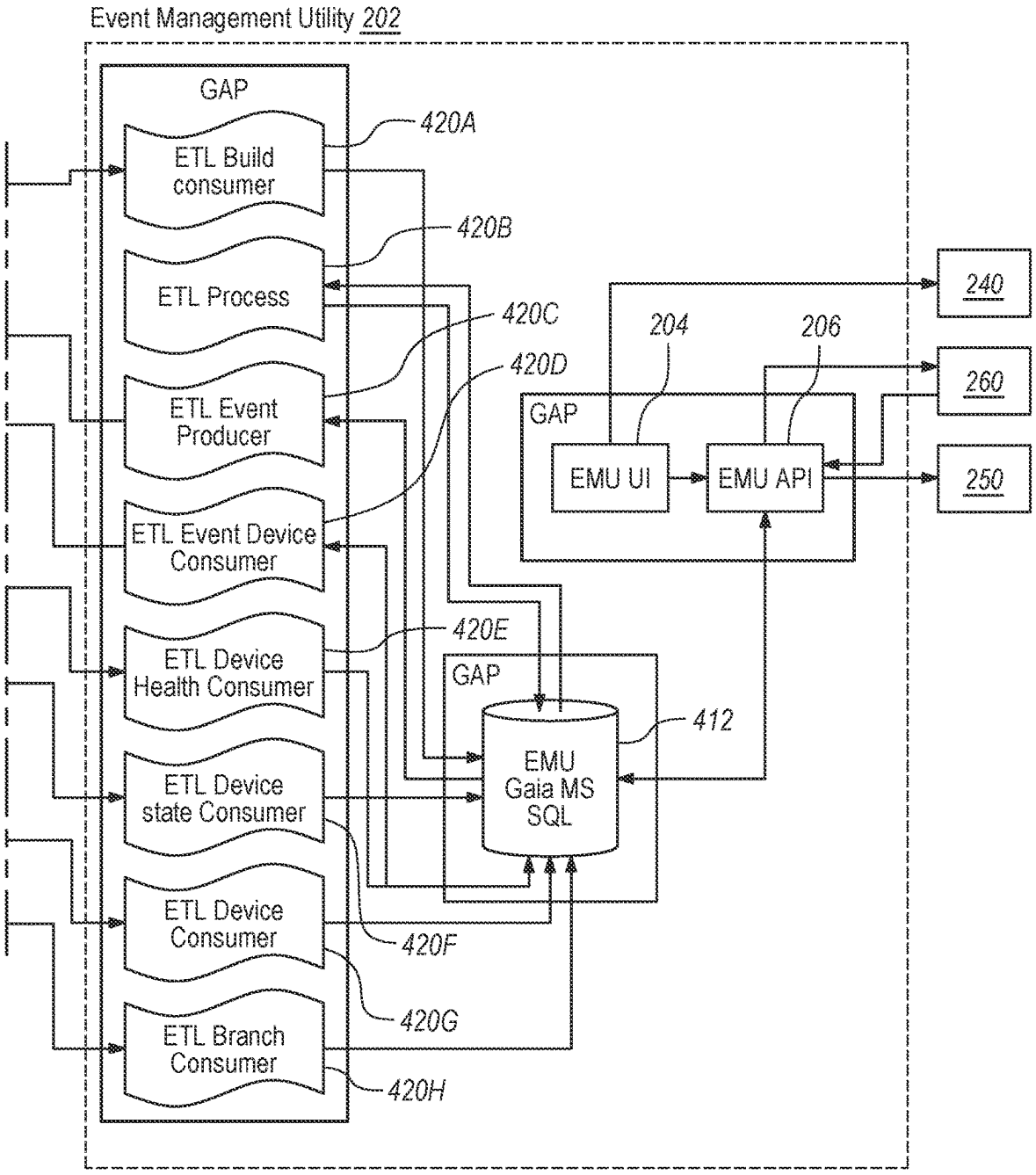

Event Management Utility 202
GAP
ETL Build consumer
420A
ETL Process
420B
ETL Event Producer
420C
ETL Event Device Consumer
420D
ETL Device Health Consumer
420E
ETL Device state Consumer
ETL Device Consumer
420F
ETL Branch Consumer
420G
420H
GAP
EMU Gaia MS SQL
412
GAP
204
206
EMU UI
EMU API
240
260
250

FIG. 4B

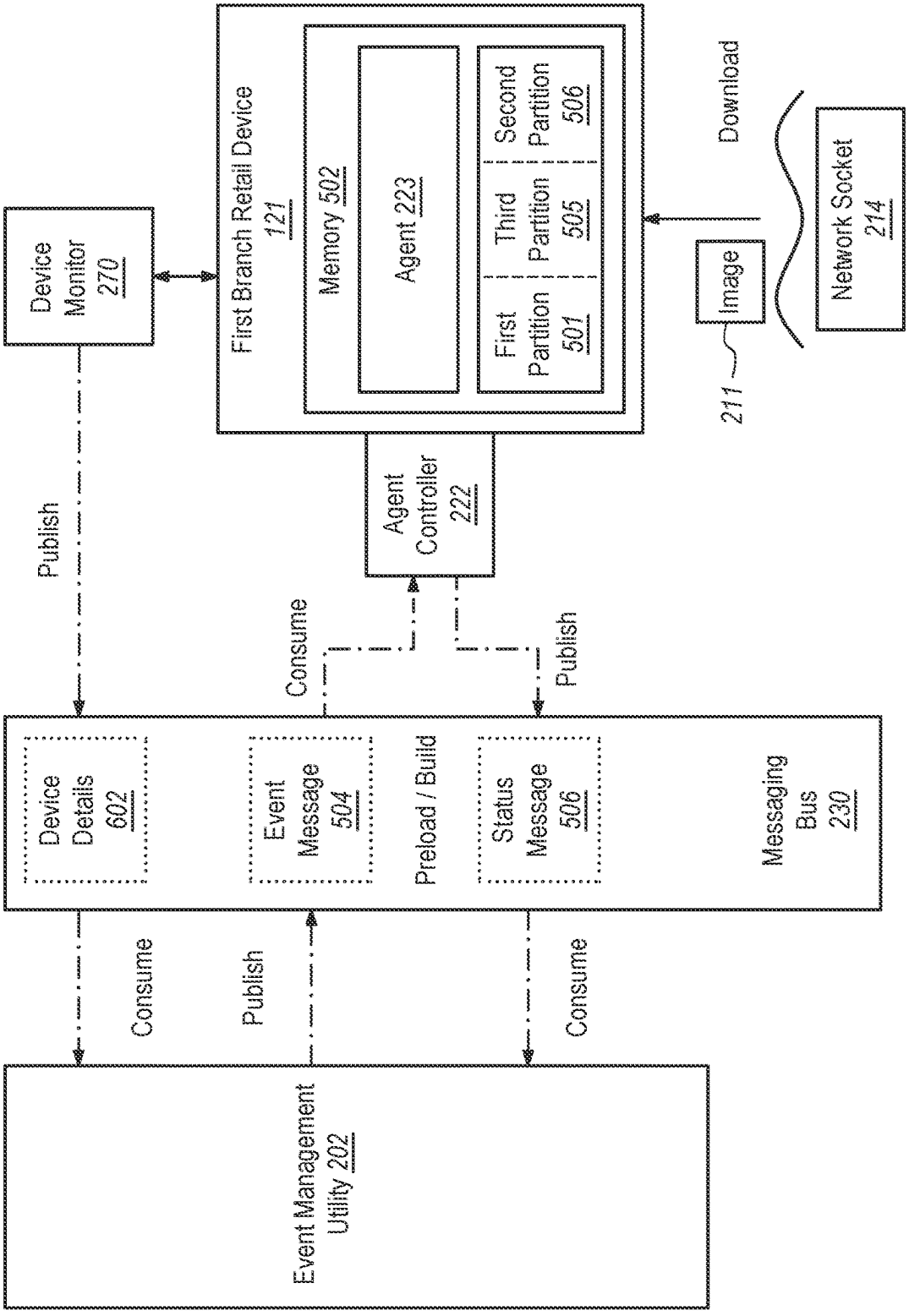

Event Management Utility 202
Consume
Publish
Consume
Messaging Bus 230
Device Details 602
Event Message 504
Preload / Build
Status Message 506
Publish
Consume
Agent Controller 222
Publish
Device Monitor 270
First Branch Retail Device 121
Memory 502
Agent 223
First Partition 501
Third Partition 505
Second Partition 506
211
Image
Download
Network Socket 214

FIG. 5

FIG. 6

FIG. 6A
FIG. 6B

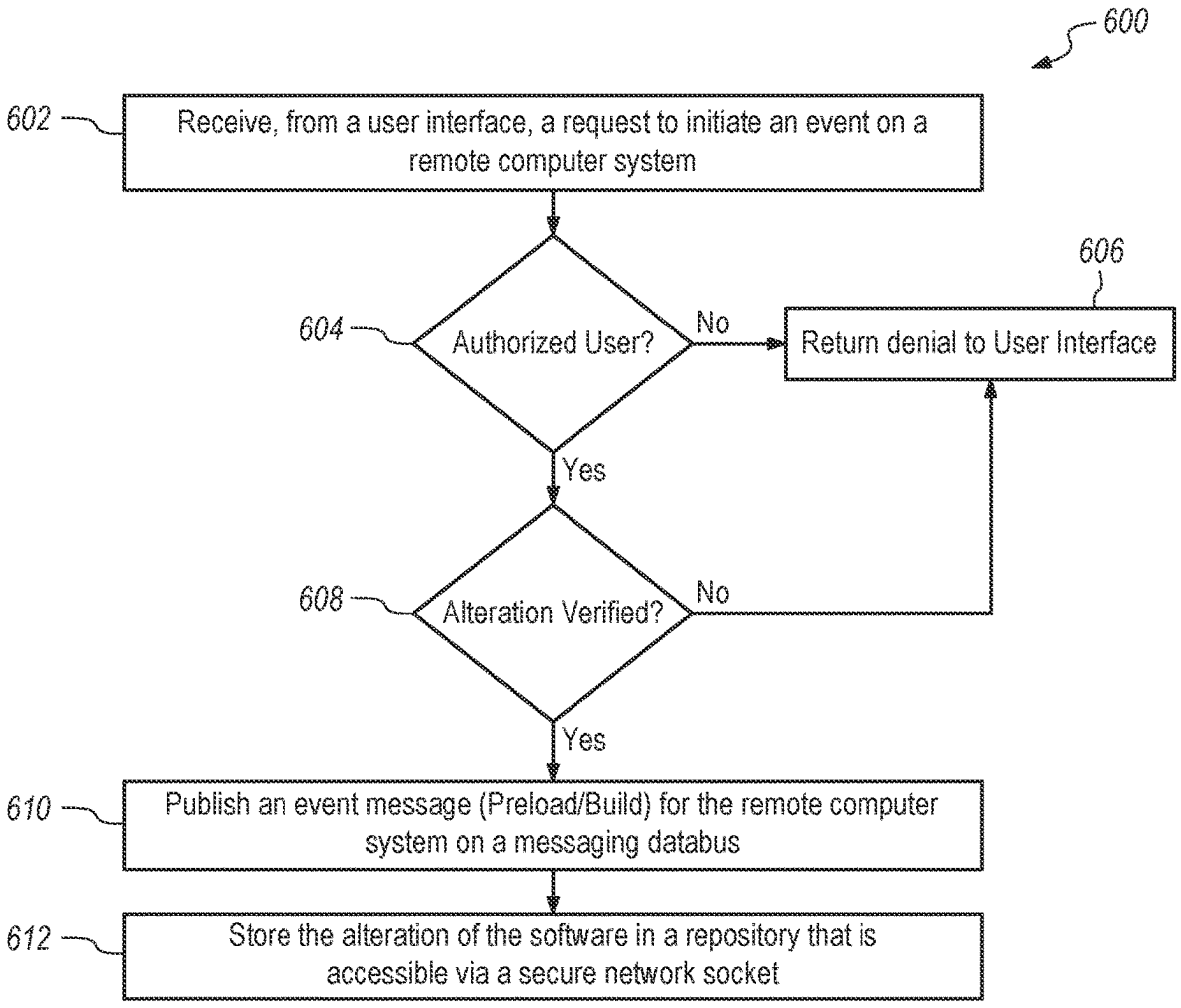


FIG. 6A

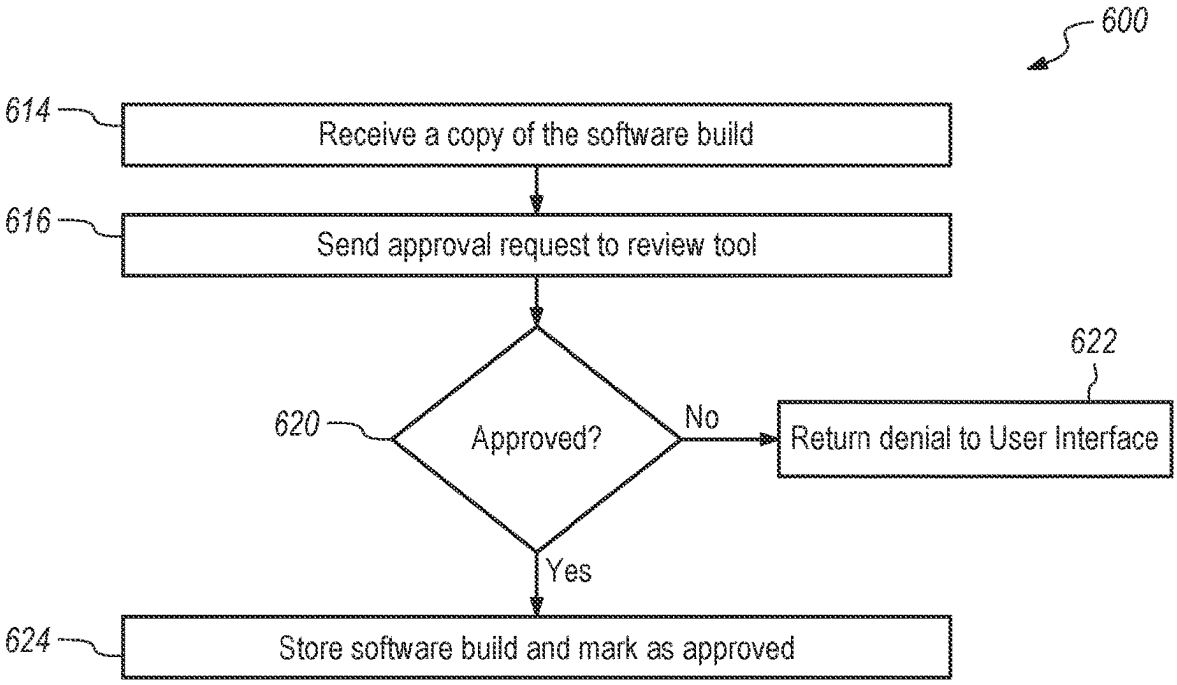

600
614
Receive a copy of the software build
616
Send approval request to review tool
620
Approved?
No
622
Return denial to User Interface
Yes
624
Store software build and mark as approved

FIG. 6B

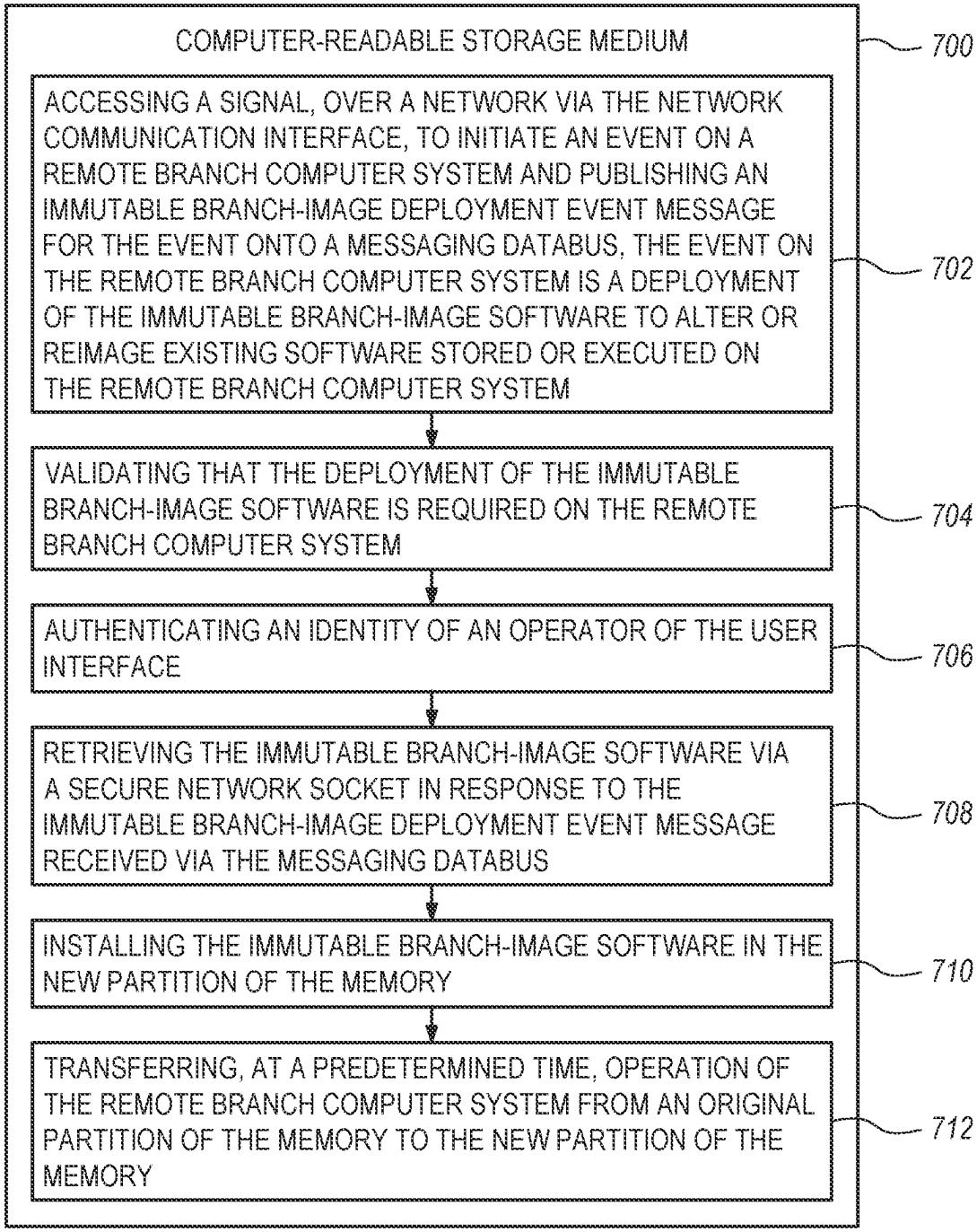

COMPUTER-READABLE STORAGE MEDIUM
700
ACCESSING A SIGNAL, OVER A NETWORK VIA THE NETWORK COMMUNICATION INTERFACE, TO INITIATE AN EVENT ON A REMOTE BRANCH COMPUTER SYSTEM AND PUBLISHING AN IMMUTABLE BRANCH-IMAGE DEPLOYMENT EVENT MESSAGE FOR THE EVENT ONTO A MESSAGING DATABUS, THE EVENT ON THE REMOTE BRANCH COMPUTER SYSTEM IS A DEPLOYMENT OF THE IMMUTABLE BRANCH-IMAGE SOFTWARE TO ALTER OR REIMAGE EXISTING SOFTWARE STORED OR EXECUTED ON THE REMOTE BRANCH COMPUTER SYSTEM
702
VALIDATING THAT THE DEPLOYMENT OF THE IMMUTABLE BRANCH-IMAGE SOFTWARE IS REQUIRED ON THE REMOTE BRANCH COMPUTER SYSTEM
704
AUTHENTICATING AN IDENTITY OF AN OPERATOR OF THE USER INTERFACE
706
RETRIEVING THE IMMUTABLE BRANCH-IMAGE SOFTWARE VIA A SECURE NETWORK SOCKET IN RESPONSE TO THE IMMUTABLE BRANCH-IMAGE DEPLOYMENT EVENT MESSAGE RECEIVED VIA THE MESSAGING DATABUS
708
INSTALLING THE IMMUTABLE BRANCH-IMAGE SOFTWARE IN THE NEW PARTITION OF THE MEMORY
710
TRANSFERRING, AT A PREDETERMINED TIME, OPERATION OF THE REMOTE BRANCH COMPUTER SYSTEM FROM AN ORIGINAL PARTITION OF THE MEMORY TO THE NEW PARTITION OF THE MEMORY
712

FIG. 7A

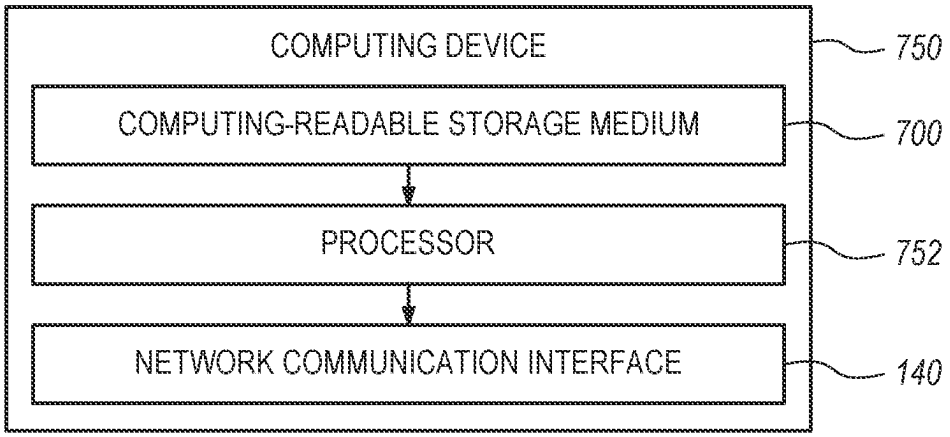

COMPUTING DEVICE
750
COMPUTING-READABLE STORAGE MEDIUM
700
PROCESSOR
752
NETWORK COMMUNICATION INTERFACE
140

FIG. 7B

DEPLOYING IMMUTABLE IMAGE SOFTWARE VIA MEMORY PARTITION OF BRANCH RETAIL DEVICES

BACKGROUND

Large enterprise networks often have hundreds if not thousands of computing devices. The computing device themselves may include or process highly sensitive data that is subject to privacy and governmental regulations. Examples of such enterprise platforms can include healthcare platforms, financial and banking enterprise platforms, information technology enterprise platforms, etc. Audits may be mandated to ensure that the enterprise platforms and any associated computing software/devices meet certain standards.

BRIEF DESCRIPTION OF THE DRAWINGS

Examples of the disclosure will be rendered by reference to specific examples which are illustrated in the appended drawings. The drawings illustrate only particular examples of the disclosure and therefore are not to be considered to be limiting of their scope. The principles here are described and explained with additional specificity and detail through the use of the accompanying drawings.

FIG. **1** illustrates an event management and immutable-images deployment platform according to an example of the present disclosure.

FIG. **2** illustrates a block diagram of an architecture of the deployment management system of FIG. **1** according to examples of the present disclosure.

FIG. **3** illustrates an example of the user interface of FIG. **2** showing an EMU (Event Management Utility) Portal.

FIG. **4**A and FIG. **4**B (both combined into FIG. **4**) illustrate the interaction between components of event management utility, the messaging databus and the remote first branch retail device in an example of the present disclosure.

FIG. **5** illustrates system event messaging between EMU and the first branch retail device to facilitate downloading of an immutable image in accordance with examples of the present invention.

FIG. **6**A and FIG. **6**B (both combined into FIG. **6**) illustrate a method for deploying immutable images according to an example of the present disclosure.

FIG. **7**A illustrates example instructions stored on a non-transitory computer-readable storage medium for deploying immutable branch-images according to one example of the present disclosure, and FIG. **7**B illustrates an example computing device according to the present disclosure.

DETAILED DESCRIPTION

From time-to-time, software that is deployed on large enterprise platforms may need to be patched to meet new security standards, for example. However, there can be difficulties associated with the patching process. On a large enterprise platform with geographically-distributed devices, the patching process can cause configuration drift. Configuration drift refers to the application of inconsistent configurations across computers or devices in a geographically distributed network.

Another difficulty associated with patching is that making changes to working systems often requires business downtime, which becomes even more difficult when the retail branch computing devices are spread out, geographically across many different time zones.

Further, it is not uncommon for software updates to fail for any number of reasons. It may be that configuration drift has caused the current version of software on a branch retail device to vary from that for which the update is compatible with. If that happens the update would fail causing the system to hang up or freeze such that settings and/or data become lost or deleted are become uncoverable.

The present invention disclosure addresses the foregoing by providing a system for deploying immutable branch-image software onto branch retail devices. The system begins by accessing a signal, over a network through a network communication interface, to initiate an event on a remote branch computer system. The signal for the event can be generated by a user via the user interface of an event management utility or engine.

Here, the user may deploy immutable branch-image software or other software alteration onto selected branch retail devices across a geographically distributed enterprise. Once the user request is received, the event is associated with the deployment of immutable branch-image software to alter or reimage existing software stored or executed on the remote branch computer system. By deploying immutable images, different and separate pieces of update patching are avoided thus eliminating configuration drift and associated gradual changes that are inconsistent with the environment.

After the event is created, the system publishes a branch-image deployment event message for the event onto a messaging databus. The system then validates that the immutable branch-image deployment is required for the remote branch computer system. In one implementation, validation can be by comparing the current status or software version of the remote branch computer system and a new status indicated by the immutable branch-image deployment event message. If the current status and the new status do not match, then alteration or reimaging of the software is required on the remote computer system.

If the update is required, the system generates a new partition in a memory of the remote branch computer system. As used herein, the term "memory" refers to any non-transitory computer-readable storage medium that can store computer-executable software instructions. The system then installs the immutable branch-image software on the new partition of the memory. Specifically, in response to the immutable branch-image deployment event message, the immutable branch-image software is retrieved from a repository that is accessible via a secure network socket for installation on the new partition. Upon completion of installation, the system then transfers, at a predetermined time, operation of the remote branch computer system from the original partition of the memory to the new partition of the memory.

If installation fails, the process is aborted and roll back occurs. In other words, if a "fail" status is assigned to the installation, operation of the remote branch computer system remains with the original partition and is not transferred to the new partition. The remote branch computer system simply boots back up on the existing base software on the original partition. In this manner, even if configuration drift has caused the current version of the base software to vary from that for which the update is compatible, any hang up or lockup that can result in settings or data loss on the remote branch computer system can be avoided. This is an enormous efficiency gain for a multi-retail branch-device enterprise network that is deploying immutable images on as many as 60,000 devices.

FIG. **1** illustrates an event management and immutable-images deployment platform **100** according to an example of the present disclosure.

In FIG. **1**, the event management deployment platform **100** can initiate, manage and coordinate events and messages to deploy immutable images. As used herein, an immutable image refers to a software object or code that cannot be changed, modified, updated or patched.

The immutable images can be deployed across any large multi-branch enterprise system with many retail computing devices. For example, event management and deployment platform **100** may be incorporated by a healthcare, government, banking institution, etc., with numerous geographically dispersed branches and locations and corresponding retail branch devices. Once initiated by a user, the system manages the entirety of the immutable image deployment onto retail branch devices based on memory partitioning without further user interaction and simplifies what can be a relatively complex deployment process.

In this example, event management and deployment platform **100** includes a cloud service **110** incorporating a deployment management system **102**. The cloud service **110** can be a public cloud service, a private cloud service, or a hybrid (public/private) cloud service. For example, the cloud service **110** can be a public cloud such as AWS™ to provide cloud services to subscribers and customers.

The deployment management system **102** is communicably coupled to multiple branch locations including a first branch location **121** and a second branch location **131** and a main location **141**. Other branch locations are not illustrated for brevity. The deployment management system **102** may manage and coordinate the deployment of immutable images, software or software alterations to numerous branch locations including the first branch location **121** and second branch location **131**.

Here, the first branch location **121** includes a number of branch retail devices including ATMs (Automatic Teller Machines) **124**, multiple branch workstations **122** and multiple branch servers **134**, some or all of which branch retail devices may receive immutable images or other software alterations at periodic intervals. The branch retail devices are collectively referred to as first branch retail devices **121** or remote computer system **121** (FIG. **2**).

Likewise, the second branch location **131** may include one or more ATMs (Automatic Teller Machines) **136**, numerous branch workstations **132** and multiple branch servers **138**, all collectively referred to as second branch retail devices **131**.

The main location **141**, which is coupled via a network communication interface **140** to the deployment management system **102**, includes a merchant service **146**, a main computer system **142** and a number of branch servers **144**. Although not shown, each of the components of main location **141**, first branch location **121** and second branch location **131** include memory devices that can be partitioned in accordance with aspects of the present disclosure. Use and operation of the event management and deployment platform **100** will now be described with reference to the FIG. **2** below.

FIG. **2** illustrates a block diagram of an architecture of the deployment management system **102** of FIG. **1** according to examples of the present disclosure.

As noted above, the deployment management system **102** can manage events related to deployment of immutable images, software and software alterations to the first branch retail devices **121**, the second branch retail devices **131** the main location **141** computing devices.

At the core of the deployment management system **102** is an event management engine or utility **202** that serves as a command control center and interfaces with an authentication tool **240**, a service management tool **250** and a review tool **260** via Application Programming Interfaces (APIs) **206**.

Here, APIs **206** facilitate, without user interaction, the exchange of information between event management utility **202** and each one of service management tool **250** and review tool **260**. The APIs facilitate information exchange to accomplish deployment of immutable images.

Event management utility **202** also interfaces with a messaging databus **230**, a repository **210** and a user interface **204** that a user **201** can employ to initiate an immutable-image deployment event. As mentioned above, immutable image deployment may be desirable for various reasons. Existing software such as an OS (Operating System) on a remote branch computer system may require patching to meet new security standards. Without patching, the network infrastructure may be exposed to malware, hackers and other bad actors that can gain unauthorized access to private personal information or worse bring down or damage the entire network infrastructure.

Within a larger network infrastructure with thousands of retail branch devices including workstations, ATMs, merchant services terminals, software alteration deployment can be difficult. When the retail branch device software is patched to meet new security stands (for example), the patching is often piece-meal. Existing base software may be followed by different and separate pieces of update patches. This can cause configuration drift or gradual changes that are inconsistent with the environment, which can trigger instability, unexpected system behavior and/or downtime. Configuration drift can also cause a major impact on security vulnerability management. Further, drift may lead cause an inability to monitor and track the health of the retail branch devices.

Many of the branch retail devices may be inactive and a large number such devices may be in different time zones such that software event coordination is necessary. Even when active, a branch retail device may lack readiness for the alteration if the device is unhealthy.

The event management utility **202** coordinates and administers dynamic or static events to cause deployment of software alterations or immutable images onto a large number of branch retail devices so that such devices can adhere to standards and report up correctly, drift can also be monitored and out-of-compliance devices can be detected. In one implementation, the event management utility **202** coordinates immutable-image deployment events by publishing branch-image deployment event messages onto the messaging databus **230**. As will be further discussed, in one implementation, the branch-image deployment event messages may include preload event messages and build event messages.

In operation, the user **201** may wish to deploy immutable images or other software alteration onto the first branch retail devices **121**, the second branch retail devices **131** and the main location **141** computing devices. The user **201** begins by interacting with the user interface **204** (e.g., a browser) to access event management utility **202**. In turn, the event management utility **202** redirects the user to the authentication tool **240**.

The authentication tool **240** authenticates user **201** to confirm that the user is an authorized user. For example, if user **201** is a release engineer, for example, authentication tool **240** can authenticate that the user is an approved user

and is authorized to implement the software deployment event. Authentication tool **204** identifies who is using system and can trace the user's activities. An example of authentication tool **204** that can be utilized with the present disclosure is IDAnywhere™. After user authentication, the user **210** is directed to the service management tool **250**.

The service management tool **250** may receive from the user a change request for the software deployment or alteration. As used herein, a "change request" is a proposal for the deployment or alteration to a product or system. Service management tool **250** manages change requests and corresponding approvals, and provides a catalog of information technology services as well as application support to the enterprise. An example of service management tool **250** us ServiceNow™. In response to the change request, the service management tool **250** generates a change ticket associated with the change request. The change ticket includes the details of user **201**'s request to deploy immutable images as entered via the user interface **204**.

One implementation of the user interface **204** is shown in FIG. **3**. Here, the user interface **204** is an EMU (Event Management Utility) Portal **304** shown in FIG. **3**. The EMU Portal **304** facilitates the deployment of immutable images in accordance with one implementation of the present disclosure. As shown by the highlighted text of FIG. **3**, the user has selected Immutable Image **302** on the menu, which then displays a number of options including Windows® 10, iOS, macOS®, Linux®, Windows® 11, Ubuntu®, etc. Here, the user **201** has selected "Windows® 11" **303** for deployment.

User **201** can also search for and select the targeted devices, first by selecting Select Devices **305**, and then By Branch **306**. Here, user **201** has selected ATMs **308** of the first branch as target devices. User **201** can also select target retail devices by Preload State, Cost Center, NetBios, Current Build, etc.

Schedule Deployment **310** is then selected to schedule deployment on a particular day and time by selecting Date **312** and Time **314**. The deployment event is submitted to and persists in a database **280** (FIG. **2**) until the scheduled time for event message publication.

System Backend

Referring to FIG. **2**, the system backend workflow continuously monitors the deployment event, and upon the scheduled event time, the workflow communicates with an agent controller **222** via preload and build event messages sent to databus **230**. The agent controller **222** may reside either external of or on the first and second branch retail devices **121**, **131** and the main location **141** computing devices.

Agent controller **222** is a control plane that facilitates the preload and build install immutable images by leveraging branch-image deployment event messages published by messaging databus **230**. The branch-image deployment event messages may include both preload event messages and build event message. The preload event message triggers the preload of immutable images for staging on the local storage of the remote branch retail device. The preloading of an event procures all impacted devices to begin downloading the associated image. The preload event can only occur after the event start time and before event end time.

Here, preloading facilitates a quicker identification of a problem during the deployment process; not every device may be connected to High-Speed Internet and immutable image download may not always be successful, thus preloading of immutable images allows the system to distinguish between a download and an install issue.

The build events assume the same as a preload event with the additional check on the preloaded event. The build event is the primary driver that sends data to the messaging databus **230** to begin installation of the image. This installation does not occur until the activation time. This is local to the machine and enables updates during off peak hours for reduced impact.

Agent controller **222** communicates with its corresponding agent **223** also residing on the branch retail devices. That is, agent controller **22** can issue commands to the agent **223** for execution on the device. Specifically, based on the published event messages, agent controller **222** communicates with agent **223** to provide instructions as to which software alterations or immutable images to download, as well as to initiate content distribution. As such, in this example, agent controller **222** and agent **223** coordinate the download of Windows® 11 immutable images onto the remote computer system **121**. Once the OS immutable image is downloaded, the respective device initiates the installation process.

In this manner, the immutable-image deployment to branch retail devices including ATMs is encompassing and self-contained, the immutable image may contain the OS, all software, all patches, all necessary scripts and artifacts for the branch devices to meet operational and security requirements as may be mandated by regulations. And during the installation process, the remote first and second branch retail devices **121**, **131** continuously indicate where in the process the installation is. The continuous monitoring data is sent via messaging databus **230** for display on user interface **204** of the event management utility **202**. After the remote first and second branch retail devices **121**, **131** are booted, a device monitor **270** is leveraged.

Device monitor **270** is an agent capable of running on different retail branch devices ATM's, merchant services terminals, work stations and servers, etc. The device monitor **270** provides two-way communication between event management utility **202** and first branch retail device **121**. The device monitor **270** provides the command control ability to issue real-time commands that the remote computer system receives and then streams back responses via the messaging databus **230** to the event management utility **202**.

By employing the device monitor **270**, the system leverages a network socket **214**. Network socket **214** runs on the remote first and second branch retail devices **121**, **131**, which when booted automatically have a live connection back to event management utility **202**. This approach leverages real-time command control from event management utility **202** to the devices without having to open specific ports.

In FIG. **2**, branch ticketing tool **460** provides users with technology support, and sources data from the messaging databus **230**. Branch ticketing tool **460** provides device and branch data to event management utility **202**. When user **201** initially selects targeted devices, it is the branch ticketing tool **460** that provides all of the retail branch device data for selection by user **201**.

As can be seen, the deployment management system **102** is an event-driven architecture and is not reliant on a request/reply architecture that is much less efficient. Event-driven architecture provides greater flexibility and much more consistency. If a branch retail device or application is offline, interaction may resume when the application comes back online.

Referring to FIG. **2**, the review tool **260** is a smart approval module that enables another team member to

review change requests. A change ticket associated with the change request may be generated, and approved by a second user or team member.

If a release engineer is to upgrade, repave or reimage branch retail devices on the system, then that upgrade can be supervised or approved by another team member. All the events that are requested are queued and upon approval, the system automatically executes all of the requested events.

FIG. **4**A (of FIG. **4**) illustrates the interaction between components of event management utility **202**, the messaging databus **230** and the remote first branch retail device **121** in an example of the present disclosure. Although not shown, the first branch retail device **121** can either be the second branch retail device **131** or the main location **141** computing devices.

Here, event management utility **202** interfaces with the message databus **230**, which itself interacts with the remote first branch retail device **121** to preload and install a build of an immutable image (e.g., an OS image) on the first branch retail device **121**.

In FIG. **4**A, the first branch retail device **121** includes the agent **223** of FIG. **2**, the device gateway **404** and the device target state manager **406**. The agent **223** facilitates the preload and build install of an immutable image. Here, both the device gateway **404** and the device target state manager **406** makeup the agent controller **222** of FIG. **2**.

The device gateway **404** is an API that fronts all of the business logic and communicates with device target state manager **406**. When a target state (e.g., Windows® 11) is selected via event management utility **202**, a system workflow generates an event message for forwarding to the messaging databus **230** regarding the expected target state. The event message may comprise a preload event message and a build event message.

The device target state manager **406** subscribes to the messaging databus **230** to picks up and store event messages. The device state manager **406** is checked by the agent **223**, which runs periodically to check on the device target state. And if the desired target state and the current state do not match, the first branch retail device **121** is queued for uplift. The target state device manager provides instructions to the agent **223** about what the expected target state should be. Based on the preload event message, the agent **223** then triggers the content delivery channel to preload the image. After preload, at the scheduled event time, the agent **223** triggers installation based on the build event message.

In accordance with one example implementation of the present disclosure, the entirety of the image is immutable, and delivered to the first branch retail device **121** such that the immutable image cannot be changed. The immutable image is also auditable because of it is a single image. As opposed to piece-by-piece installation, the entirety of the image is preloaded from the repository **210** (FIG. **2**).

In accordance with an implementation of the present invention, the retail branch device is partitioned into at least two partitions. As used herein, a partition is a logical division of a hard disk that is treated as a separate unit by operating and file systems. Here, the existing base software remains on the first partition while the immutable image is installed on the second partition. As installation progresses, command control is then used to determine the status of the installation. Command control is a two-way interaction between event management utility **202** and the first branch retail device **121**.

If installation is successful, that is, command control determines that there are no errors and assigns a "success" status to the installation, then the system transfers, at a predetermined time, operation of the remote branch computer system from first partition to the second partition. The predetermined time for transfer would be any time after the installation is determined to be successful.

If installation fails, the process is aborted and roll back occurs. In other words, if command control determines that an error has occurred on the remote branch computer system and assigns a "fail" status to the installation, operation of the remote computer system remains with the first partition and is not transferred to the second partition.

The system boots back up on the existing base software on the first partition. The current and previous states are kept on the computer system at all times. This is an enormous efficiency gain for a multi-retail branch-device enterprise network that is deploying immutable images on as many as 60,000 devices.

In accordance with an implementation of the present disclosure, content delivery of an immutable image is based on co-ordination of various events and messages published by the messaging databus **230**.

Referring to FIG. **4**A, the databus **230** may be for example, a GTI databus with databases **230**A, **230**B, **230**C and **230**D. The databases serve to store topics. For example, the database **230**C stores device health topics **421** and device state topics **418**. Here, the agent **223**, the device gateway **404** and the device target state manager **406** stores the device state topic **418** that logs the device states of first branch retail device **121**, that is, whether the computer system is active or inactive.

Here, EMU **202** of FIG. **4**B includes a database **412** (e.g., SQL) and various workflows **420**A, **420**B, **420**C, **420**D, **420**E, **420**F, **420**G, **420**H. Workflow **420**F, for example, is an ETL (Extract, Transform, Load) device state consumer. Thus, the device state topic **418** is consumed by workflow **420**F for storage in the database **412**. Consequently, the API **206** of event management utility **202** can interface with the database **412** to access the device state topic for user interface **204**, service management tool **250** and review tool **260**.

In a similar manner, the agent **223**, the device gateway **404** and the device target state manager **406** also provide device health topic **421** that logs the first branch retail device **121** health states, that is, whether the system is healthy and in the target states. The health state topic is consumed by ETL device health consumer **420**E workflow also for storage in a database **412**. API **206** interfaces with the database **412** to access the device health topic for use by user interface **204**, service management tool **250** and review tool **260**. In this manner, according to one aspect of the present disclosure, the device state and health topics about a retail branch device is continuously available to facilitate immutable image installation and the avoidance of drift.

Referring now to FIG. **4**B, the device state topic consumed by ETL device state consumer **420**F workflow is also picked up by ETL event producer **420**C that logs event topics **430** in the database **230**B of messaging databus **230**. The event topics **430** relate to immutable image deployment events as may be requested by user **201**. Similarly, the device health topic consumed by ETL device health consumer **420**E workflow is also picked up by ETL event device producer **420**D that logs event device topics **427** in database **230**B. At a first layer, the event topic **430** sends related information to the messaging databus **230**, then the event device topics **427** sends the devices for the event in a separate payload. This is because some event metadata information can change, and it is undesirable to change device information after an event has preloaded and also

undesirable to send this same payload over and over again. The orchestration of events and corresponding topics thus provides a basis for proper monitoring of the entirety of the system.

Referring to FIG. **4**B, branch ticketing tool **460** provides device and branch data to event management utility **202**. When user **201** initially selects targeted devices, it is the branch ticketing tool **460** that provides all of the retail branch device data for selection by user **201**.

Branch ticketing tool **460** includes device API **440**, branch collector API **442** and device collector API **444**. Device API **440** stores device data on database **446** that is pushed to branch collector API **442** and device collector API **444**. The device collector API **444** is a producer for device topic **448** stored in a database **230**D. ETL device consumer workflow **420**G then consumes the data. The branch collector API **442** is a producer for branch topic **441** stored in database **430**D. An ETL branch consumer workflow **420**H consumes the data.

FIG. **5** illustrates system event messaging between EMU **202** and the first branch retail device **121** to facilitate downloading of an immutable image **210**A in accordance with examples of the present invention.

To initiate deployment, event management utility **202** publishes an event message **504** onto the messaging databus **230**. As used herein, an event is the occurrence or the scheduling of the deployment of an immutable image. Specifically, here, the event message **504** is a request to deploy software alterations or immutable images onto the first branch retail device **121**. In one implementation, the branch image software **211** (FIG. **2**) that is to be deployed is stored in a repository **210** (FIG. **2**). The repository **210** may be accessible via a secure network socket **214** (FIG. **2**).

In one implementation, the event message **504** published by EMU **202** is comprised of a preload event message and a build event message for the deployment of the branch-image software. Event message **504** can be based on an entity object that may have attributes including an event ID, a name, an access role, a target environment, a start and a stop time, a target build and applicable target devices, for example.

In one example, the preload and build events, may be preceded by a draft, saved, and submitted events. Event statuses are binary for all statuses apart from the build event, which takes continuous feedback from the messaging database **230** to determine build status completion percentages.

The first branch retail device **121** (remote computer system **121**) includes memory **502** in which an agent **223** resides. The agent **223** works in concert with the agent controller **222**, to facilitate the installation and deployment of immutable images such as an (OS) Operating System. The deployment is facilitated by agent controller **222** and/or agent **223** subscribing to and consuming event message **504**. Specifically, the agent controller **222** is the entity that subscribes to event message **504** and monitors the messaging databus **230** to detect the preload event message. The agent **223** check in with agent controller **222** periodically for messages, and responsive to detecting the preload event message, the agent **223** preloads or downloads the image **211** from the repository **210** onto the remote first branch retail device **121** via secure socket network **214**.

Specifically, the image **211** is downloaded to a third partition **505** in memory **502**. The installation and all corresponding processes then occur with respect to second partition **506**. The existing software that is to be reimaged or altered remains stored in a first partition **501**. Installation is controlled and progresses via the existing software in the first partition **501**.

The agent controller **222** is the entity that subscribes to the build event message on messaging databus **230**. The agent **223** checks in periodically (e.g., every five minutes) with agent controller **222** for messages. Upon detection of the build event message, the agent **223** installs the downloaded branch image software **211** onto the first partition **501** of the remote first branch retail device **121** in response detecting the build event message.

The agent controller **222** also subscribes to the messaging databus **203** to publish a status message **506** that indicates the status of the installation or deployment as the process proceeds. EMU **202** subscribes to the status message **506** and can consume those messages for display on its user interface.

The agent controller **222** continuously monitors the installation, and publishes the status message **506** (for example, whether the installation is a "paused," "success," "fail," etc. for example. If installation is successful, that is, the status message **506** publishes "success," then the system transfers operation of the remote branch computer system from first partition **501** to the second partition **506**. However, installation may fail. If status message **506**, publishes a "fail" message, the process is aborted. Operation of the remote computer system, i.e., first branch retail device **121** is not transferred to the second partition **506**. The system may boot backup on the existing base software on the first partition **501** to avoid data and setting losses.

In FIG. **5**, the device monitor **270** provides two-way real-time command and control between the first branch retail device **121** and EMU **202**. This two-way real-time command and control facilitates the trouble shooting of issues. Device Monitor **270** may also publish device details **602** and device inventory to the messaging bus **230** for consumption by EMU **202** for inventory and compliance reporting.

FIG. **6**A illustrates a method **600** for deploying immutable images according to an example of the present disclosure.

In FIG. **6**A, at block **602**, method **600** accesses a signal, over a network via a network communication interface, to initiate an event on a remote first branch retail device. In one example, the event may be initiated by publishing a branch-image deployment event message onto a messaging databus.

Specifically, the accessed signal may be from a user interface to initiate the event on the remote first branch retail device. Note that the event may be initiated on additional remote branch retail devices that satisfy a user-selectable attribute (see e.g., FIG. **3**). Method **600** then proceeds to decision block **604**.

At decision box **604**, it is determined whether the user is authorized to initiate the event. User authorization is determined by the authentication tool **240**, which authenticates that the user is an authorized user. If the user is not an authorized user, a denial of service is returned to the user as shown at block **606**. If the user is an authorized user, method **600** proceeds to decision block **608**.

At decision block **608** (optional), method **600** verifies the alteration or deployment. In one implementation, verification can be by comparing the current status or software version of the remote computer system **121** and a new status indicated by the immutable branch-image deployment event message. The new status is based on the version of the immutable branch-image that is to be deployed. If the current status and the new status do not match, then altera-

tion or reimaging of the software is required on the remote computer system. That is, the software alteration is verified.

If the software alteration verification fails, method **600** reverts and sends a denial of service message to the user at block **606**. If the software alteration verification succeeds, method **600** proceeds to block **610**.

At block **610**, the system publishes an event message for the remote first branch retail device on the databus **230**. Event messages for the additional remote computer systems may also be published on the messaging databus **230** for consumption by agents associated with the additional remote computer systems. In one implementation, the event message may comprise both a preload event message and a build event message. The preload event message serves to coordinate the download of immutable images from a remote location for staging on the local storage of the remote first branch retail device.

Specifically, the scheduling workflow creates and sends the preload event to the messaging databus **230** to trigger the preload process, handled by the controller and controller agent. The agent **223** publishes the preload status to the controller **222** that owns the preload topic in the databus **230**. The preload data consumer will subscribe to the controller **222** owned preload topic to process events and update device state.

At block **612**, method **600** stores the immutable images in the repository **230** that is accessible via the secure network socket **214**.

At block **614** of FIG. **6**B, the system downloads a copy of the immutable image software build onto first partition **501** (FIG. **5**). The messaging databus is monitored to detect the preload event message to initiate the download.

At block **616**, an approval request is sent to the review tool **260**. Review tool **260** enables all deployments, repaves, etc. to be supervised by a team leader or approved by another team member. All the events that are requested are queued and upon approval, the system automatically executes the deployment. Method **600** then proceeds to decision box **620**.

At decision box **620**, if the approval request is not granted, then a denial of service is sent to the user at block **622**. If the approval request is granted, method **600** proceeds to block **624**.

At block **624**, the immutable image on the first partition **501** of the first branch detail device. Installation on the second partition **506** may be triggered, by the agent, in response detecting the build event message.

FIG. **7**A illustrates example instructions stored on a non-transitory computer-readable storage medium **700** for deploying immutable branch-images according to one example of the present disclosure, and FIG. **7**B illustrates an example computing device **750** according to the present disclosure.

As shown in FIG. **7**A, the non-transitory computer-readable storage medium **700** includes instruction **702**, instruction **704**, instruction **706**, instruction **708**, **710** and **712**.

Instruction **702** may cause a processor **752** to access, over a network via the network communication interface, a signal to initiate an event on a remote branch computer system and publish an immutable branch-image deployment event message for the event unto a messaging databus. The event on the remote computer system is a deployment of immutable branch-image software to alter or reimage existing software stored or executed on the remote computer system.

Instruction **704** may cause a processor **752** to validate that the deployment of the immutable branch-image software is required on the remote branch computer system.

Instruction **706** may cause a processor **752** generate a new partition in a memory of the remote branch computer system.

Instruction **708** may cause a processor **752** to retrieve the immutable branch-image software via a secure network socket in response to the immutable branch-image deployment event message received via the messaging databus.

Instruction **710** may cause a processor **752** to install the immutable branch-image software on the new partition of the memory.

Instruction **712** may cause a processor **752** to transfer, at a predetermined time, operation of the remote branch computer system from an original partition of the memory to the new partition of the memory.

The non-transitory computer-readable storage medium **700** may be any electronic, magnetic, optical, or other physical storage device that stores executable instructions. For example, the non-transitory computer-readable storage medium **700** may be random access memory (RAM), an electrically-erasable programmable read-only memory (EEPROM), a storage drive, an optical disc, or the like. The non-transitory computer-readable storage medium **700** can be encoded to store executable instructions that cause the processor **752** to perform operations according to examples of the disclosure.

The present disclosure may employ a software stack to enlist the underlying tools, frameworks, and libraries used to build and run example applications of the present disclosure. Such a software stack may include PHP, React®, Cassandra™ Hadoop®, Swift®, etc. The software stack may include both frontend and backend technologies including programming languages, web frameworks servers, and operating systems. The frontend may include JavaScript®, HTML, CSS, and UI frameworks and libraries. In one example, a MEAN (MongoDB®, Express.js, AngularJS, and Node.js) stack may be employed. In another example, a LAMP (Linux®, Apache®, MySQL®, and PHP) stack may be utilized.

As used herein, references to modules, engines, etc. refer to self-contained computer modules that have lines of executable code instructions. Each such module comprised of software instructions can be standalone.

While particular examples have been described, various modifications, changes and substitutions are intended in the foregoing disclosures, and it will be appreciated that in some instances some features of particular examples will be employed without a corresponding use of other features without departing from the scope and spirit as set forth. Therefore, many modifications may be made to adapt a particular situation or material to the essential scope and spirit.

Any suitable programming language can be used to implement the routines of particular examples including C, C++, Java®, JavaScript®, assembly language, etc. Different programming techniques can be employed such as procedural or object oriented. The routines may execute on specialized processors.

The specialized processor may include memory to store a set of instructions. The instructions may be either permanently or temporarily stored in the memory or memories of the processing machine. The processor executes the instructions that are stored in the memory or memories in order to process data. The set of instructions may include various instructions that perform a particular task or tasks, such as those tasks described above. Such a set of instructions for performing a particular task may be characterized as a software program.

As used in the description herein and throughout the claims that follow, "a", "an", and "the" includes plural references unless the context clearly dictates otherwise. Also, as used in the description herein and throughout the claims that follow, the meaning of "in" includes "in" and "on" unless the context clearly dictates otherwise.

While the above is a complete description of specific examples of the disclosure, additional examples are also possible. Thus, the above description should not be taken as limiting the scope of the disclosure, which is defined by the appended claims along with their full scope of equivalents.

The invention claimed is:

1. A server system comprising:

a network communications interface;

a storage medium having instructions;

one or more processors that execute the instructions to perform operations comprising:

implementing an authentication protocol that authenticates whether a user is an approved user authorized to implement a software deployment event;

accessing a signal, over a secure network socket via the network communication interface that operates on a remote branch computer system comprising a first branch corresponding to a first retail computing device and a second branch corresponding to a second retail computing device, to initiate an event comprising the software deployment event on the remote branch computer system by the approved user;

publishing an immutable branch-image deployment event message for the event onto a messaging databus, wherein the event on the remote branch computer system is a deployment of an immutable branch-image software to alter or reimage existing software stored or executed on the remote branch computer system;

implementing a control plane that utilizes the published immutable branch-image deployment event message in executing a preload of an event message and a build event associated with a build event message;

validating that the deployment of the immutable branch-image software is required on the remote branch computer system;

generating a new partition in a memory of the remote branch computer system;

retrieving the immutable branch-image software via the secure network socket in response to the immutable branch-image deployment event message received via the messaging databus;

installing the immutable branch-image software in the new partition of the memory; and

transferring, at a predetermined time, operation of the remote branch computer system from an original partition of the memory to the new partition of the memory.

2. The server system of claim **1**, wherein the one or more processors perform operations comprising:

determining that an error has occurred on the remote branch computer system operating on the new partition of the memory; and

retaining operation of the remote branch computer system on the original partition of the memory.

3. The server system of claim **1**, wherein the one or more processors perform operations comprising:

determining a current status of the remote branch computer system;

comparing the current status to a new status included in the immutable branch-image deployment event message; and

in response to the current status not matching the new status, determining that the alteration or reimaging of the software is required on the remote branch computer system.

4. The server system of claim **1**, wherein the signal is a change request, via a user interface, to discover the remote branch computer system for deployment based on at least one attribute; and

identify that the remote branch computer system satisfies the at least one attribute.

5. The server system of claim **4**, wherein the at least one attribute includes a type of the remote branch computer system, an identification of the remote branch computer system, a location of the remote branch computer system, a health status of the remote branch computer system, and a current software build of the remote branch computer system.

6. The server system of claim **1**, wherein the remote branch computer system is one of a merchant services terminal and an automatic teller machine.

7. The server system of claim **1**, wherein the one or more processors perform operations comprising:

discovering that additional remote branch computer systems satisfy at least one attribute; and

publishing event messages for the additional remote branch computer systems on the messaging databus, wherein agents communicating with the additional remote branch computer systems subscribe to the messaging databus.

8. The server system of claim **1**, wherein the one or more processors perform operations comprising:

publishing, to the messaging databus, a message that indicates a status of the alteration of the software.

9. A non-transitory computer-readable storage medium including computer-executable instructions stored thereon which, when executed by a processor, causes the processor to perform the following operations:

implementing an authentication protocol that authenticates whether a user is an approved user authorized to implement a software deployment event;

accessing a signal, over a secure network socket via a network communication interface that operates on a remote branch computer system comprising a first branch corresponding to a first retail computing device and a second branch corresponding to a second retail computing device, to initiate an event comprising the software deployment event on the remote branch computer system by the approved user;

publishing an immutable branch-image deployment event message for the event onto a messaging databus, wherein the event on the remote branch computer system is a deployment of an immutable branch-image software to alter or reimage existing software stored or executed on the remote branch computer system;

implementing a control plane that utilizes the published immutable branch-image deployment event message in executing a preload of an event message and a build event associated with a build event message;

validating that the deployment of the immutable branch-image software is required on the remote branch computer system;

generating a new partition in a memory of the remote branch computer system;

retrieving the immutable branch-image software via the secure network socket in response to the immutable branch-image deployment event message received via the messaging databus;

installing the immutable branch-image software on the new partition of the memory, and

transferring, at a predetermined time, operation of the remote branch computer system from an original partition of the memory to the new partition of the memory.

10. The non-transitory computer-readable storage medium of claim **9**, wherein the instructions further comprise:

determining that an error has occurred on the remote branch computer system operating on the new partition of the memory; and

transferring operation of the remote branch computer system from the new partition of the memory to the original partition of the memory.

11. The non-transitory computer-readable storage medium of claim **10**, wherein the instructions further comprise,

determining a current status of the remote branch computer system;

comparing the current status to a new status included in the immutable branch-image deployment event message; and

in response to the current status not matching the new status, determining that the alteration of the software is required on the remote branch computer system.

12. The non-transitory computer-readable storage medium of claim **10**, wherein the signal is a change request, via a user interface, to discover the remote branch computer system for deployment based on at least one attribute; and

identify that the remote branch computer system satisfies the at least one attribute.

13. The non-transitory computer-readable storage medium of claim **12**, wherein the at least one attribute comprises one or more of an identification of the remote branch computer system, a type of the remote branch computer system, a location of the remote branch computer system, a version of the software of the remote branch computer system, and a health status of the remote branch computer system.

14. The non-transitory computer-readable storage medium of claim **10**, wherein the instructions further comprise;

identifying that one or more additional remote branch computer systems satisfy at least one criterion; and

publishing event messages for the one or more additional remote branch computer systems on the messaging databus, wherein agents communicating with the one or more additional remote branch computer systems subscribe to the messaging databus.

15. The non-transitory computer-readable storage medium of claim **14**, wherein the instructions further comprise:

receiving, via the messaging databus, a message that indicates a status of the alteration of the software; and

generating a visual indication of the status via a user interface.

16. The non-transitory computer-readable storage medium of claim **9**, wherein the remote branch computer system is one of a merchant services terminal and an automatic teller machine.

17. A computer-implemented method comprising:

implementing an authentication protocol that authenticates whether a user is an approved user authorized to implement a software deployment event;

accessing, over a secure network socket via a network communication interface that operates on a remote branch computer system comprising a first branch corresponding to a first retail computing device and a second branch corresponding to a second retail computing device, a signal comprising a change request to initiate an event comprising the software deployment event, on the remote branch computer system by the approved user;

publishing an immutable branch-image deployment event message for the event onto a messaging databus, wherein the event on the remote branch computer system is a deployment of an immutable branch-image software to alter or reimage existing software stored or executed on the remote branch computer system;

implementing a control plane that utilizes the published immutable branch-image deployment event message in executing a preload of an event message and a build event associated with a build event message;

validating that the deployment of the immutable branch-image software is required on the remote branch computer system;

generating a new partition in a memory of the remote branch computer system;

retrieving the immutable branch-image software via a secure network socket in response to the immutable branch-image deployment event message received via the messaging databus;

installing the immutable branch-image software on the new partition of the memory, and

transferring, at a predetermined time, operation of the remote branch computer system from an original partition of the memory to the new partition of the memory.

18. The method of claim **17** further comprising:

determining that an error has occurred on the remote branch computer system operating on the new partition of the memory; and

transferring operation of the remote branch computer system from the new partition of the memory to the original partition of the memory.

19. The method of **17** further comprising:

determining a current status of the remote branch computer system;

comparing the current status to a new status included in the immutable branch-image deployment event message; and

in response to the current status not matching the new status, determining that the alteration of the software is required on the remote branch computer system.

20. The method of claim **17** further comprising:

generating a second new partition and downloading the immutable branch-image software onto the second new partition.

* * * * *