Innovation, Sciences et Développement économique Canada
Office de la Propriété Intellectuelle du Canada

Innovation, Science and Economic Development Canada
Canadian Intellectual Property Office

CA 2988332 C 2021/08/17

(11)(21) 2 988 332

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2016/07/05
(87) Date publication PCT/PCT Publication Date: 2016/12/08
(45) Date de délivrance/Issue Date: 2021/08/17
(85) Entrée phase nationale/National Entry: 2017/12/04
(86) N° demande PCT/PCT Application No.: US 2016/041019
(87) N° publication PCT/PCT Publication No.: 2016/197155

(51) Cl.Int./Int.Cl. *G06F 21/60* (2013.01), *G06F 12/16* (2006.01), *G06F 21/50* (2013.01)

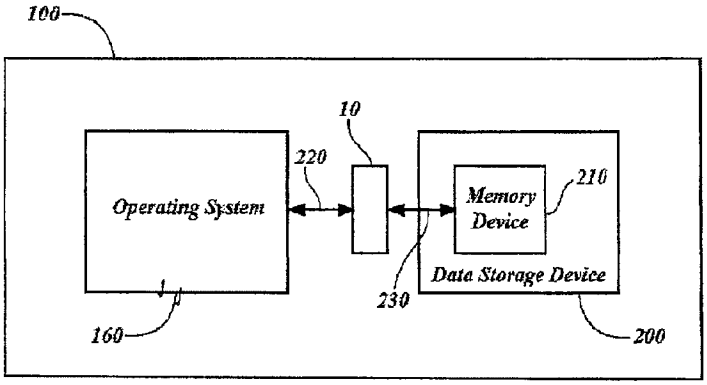
(72) Inventeur/Inventor:
COPELAND, SCOTT R., US

(73) Propriétaire/Owner:
VIIRII, LLC, US

(74) Agent: OSLER, HOSKIN & HARCOURT LLP

(54) Titre : SYSTEME D'EXPLOITATION INDEPENDANT, SOUS-SYSTEME DE MEMORISATION DE DONNEES SECURISEE

(54) Title: OPERATING SYSTEM INDEPENDENT, SECURE DATA STORAGE SUBSYSTEM



(57) Abrégé/Abstract:
An intermediary data handler is used in a Secured Data Storage Subsystem (SDSS), to provide a host electrical computer system with security of certain data stored in memory of the computer system's static data storage device. The intermediary data handler is functionally disposed between the operating system (OS) and data storage device of the host computer. The data handler has Processor, Memory, and User Interface circuits, and resident software adapted to generate mocked-up response data in reply to an unauthorized read/write communication from the OS, the mock data response being automatically formatted to have a content and data-structure format acceptable by the host OS, while isolating and controlling the original communication from the OS. The SDSS includes host software adapted to integrate operation and function of the intermediary data handler with the host computer system to accomplish the security of data stored on the storage device.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 December 2016 (08.12.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/197155 A1

(51) International Patent Classification:
G06F 21/60 (2013.01) *G06F 12/16* (2006.01)
G06F 21/50 (2013.01)

(21) International Application Number:
PCT/US2016/041019

(22) International Filing Date:
5 July 2016 (05.07.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/170,116 2 June 2015 (02.06.2015) US

(71) Applicant: **VIIRII, LLC** [US/US]; 1085 Burnt Hickory Road NW, Marietta, GA 30064 (US).

(72) Inventor: **COPELAND, Scott, R.**; 1710 Effie Lane, Pasadena, TX (US).

(74) Agent: **PERNIA, Sherman, D.**; P.O. Box 58554, Houston, TX 77258 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

— *with information concerning request for restoration of the right of priority in respect of one or more priority claims (Rules 26bis.3 and 48.2(b)(vii))*

(54) **Title**: OPERATING SYSTEM INDEPENDENT, SECURE DATA STORAGE SUBSYSTEM

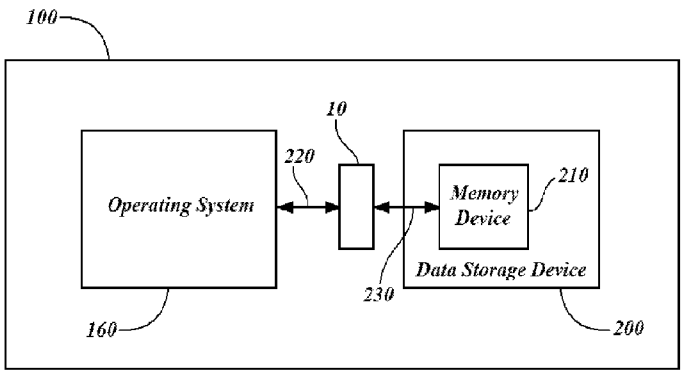


Figure 1B

(57) **Abstract**: An intermediary data handler is used in a Secured Data Storage Subsystem (SDSS), to provide a host electrical computer system with security of certain data stored in memory of the computer system's static data storage device. The intermediary data handler is functionally disposed between the operating system (OS) and data storage device of the host computer. The data handler has Processor, Memory, and User Interface circuits, and resident software adapted to generate mocked-up response data in reply to an unauthorized read/write communication from the OS, the mock data response being automatically formatted to have a content and data-structure format acceptable by the host OS, while isolating and controlling the original communication from the OS. The SDSS includes host software adapted to integrate operation and function of the intermediary data handler with the host computer system to accomplish the security of data stored on the storage device.

WO 2016/197155 A1

OPERATING SYSTEM INDEPENDENT, SECURE DATA STORAGE SUBSYSTEM

Field of the Invention

[0002] The present invention is in the field of electrical computer or digital data processing system processes and apparatuses for selecting and accessing data storage. Specifically, the present invention comprises systems, methods, and apparatus for preventing file data alteration not consistent with defined security policy. More specifically, the subject matter of the present invention comprises means to prevent data/file tampering by limiting write access to authorized entities or processes.

Summary of the Invention

[0003] The present Secured Data Storage Subsystem (SDSS) is independent of the operating system (OS) of the electrical computer/data processing system in which it is being used. The SDSS includes an intermediary data handler device that is functionally disposed between the OS and the memory of the main data storage device of an electrical computer or data processing system. The purpose of the intermediary data handler is to protect certain data (e.g., registry files) on the storage device from modification (e.g., by malware or unauthorized access). In the embodiments illustrated herein, the SDSS subsystem takes an incoming modify/write request from the operating system, and processes the request (and any associated data), not according to the operating system's instructions, but according to its own file system and rule set instructions - without writing to or modifying the target file on the storage device. The SDSS then "represents" to the OS that it has complied with the request and replies with an appropriate mocked-up data response. That is, the outgoing data the mock response has the content and structural format that is expected by the OS in a reply from the storage device. The SDSS technology purposefully misreports the result of the modify/write request targeting protected data and files, while maintaining autonomous control over the protected data and how it is handled. This provides the security and integrity of the protected data on a storage device, and has the advantage of being usable across platforms, without regard to the particular OS of the host electrical computer or data processing system.

[0004] Since the SDSS technology stores the protected data according to its own structure/rule set, and mimics in its replies to the OS whatever data structure the OS expects to see, the data stored on a storage device with the SDSS technology is readily accessible to any particular OS, without needing to change to the partition or format structure. The present SDSS can mimic the NTFS format while in a Microsoft Windows computer in one moment, and seconds later mimic the EXT4 format while being accessed by a Linux computer. The SDSS technologies intermediary data handler can mimic every format structure and partition scheme used by substantially all current operating systems. The present invention utilizes

two distinct modes of operation. Normal mode allows the SDSS to handle read and write requests by an operating system to fulfill normal daily operations. The Administrative mode allows the SDSS to provide options to a User (i.e., an system administrator) form outside of the computer and the operating system. A User can utilize a mobile computing device and its WiFi/Bluetooth capability to speak directly with the SDSS hardware and utilize administrative functions independent of the storage devices or operating systems control. This system leaves hackers with no ability to influence the handling of data on the storage device as they do not have access to the SDSS firmware and can only influence the operating system. Even if the operating system is compromised, the SDSS technology is independent and will not allow the operating system to corrupt protected data in the storage device memory.

Reference Numerals

10 - Intermediary Data Handler

15 - Communications Layer (Intermediary Data Handler **10** & Generated Data **20**)

20 - Generated (outgoing) Data

25 - Communications Layer (Intermediary Data Handler **10** & Admin Only Data **30**)

30 - Admin Only Data

40 - Converted Data Structure

45 - OS Specific Data Structure(s): e.g., a) Linux; b) Windows; c) OS X

50 - Data Conversion by Intermediary Data Handler **10**

100 - Computing Device

110 - Communications Layer, Read/Write requests between Computing Device **100** & Storage Device **200**

120 - User Device

130 - User Communications App

140 - Signal (Wireless)

150 - USB Connection

160 - Operating System(s): a) Linux; b) Windows; c) OS X

170 - CPU

175 - RAM

178 - SDSS/Data Handler Software/Instruction Set

180 - BIOS (Basic Input Output System) / ROM (Read Only Memory)

185 - Wireless Communication Hardware

190 - Administrator Device

195 - Admin Software Application

200 - Storage Device

210 - Storage Device Memory

220 - Communications Layer (Storage Device **200** & Intermediary Data Handler **10**)

230 - Communications Layer (Data Handler **10** & Storage Device Memory **210**)

300 - Received Write Request Process

305 - Start: processing received Write Request

310 - Process

320 - Decision

330 - Process

340 - Process

350 - Process

360 - Process

370 - End: processing received Write Request

400 - Received Read Request Process

405 - Start: processing received Read Request

410 - Process

420 - Process

430 - Process

440 - Process

450 - End: processing received Read Request

500 - Change OS Platform Emulation

500 - Start: change OS emulation process

510 - Decision

520 - Decision

530 - Process

540 - Decision

550 - End: change OS emulation process

800 - Administrative Mode Process

805 - Start: enter Administrative Mode

810 - Decision

820 - Decision

830 - Process

840 - Decision

850 - Process

860 - End: exit Administrative Mode

Brief Description of Figures

[0005] Figures 1A & 1B are schematic overviews: (A) of a typical prior art electrical computer system, and (B) of an electrical computer system including the present Secure Data Storage Subsystem.

[0006] Figure 2 illustrates hardware circuits useful in an Intermediary Data Handler.

[0007] Figure 3A shows a User Device communicating with the Intermediary Data Handler.

[0008] Figure 3B shows an Admin Device using an Admin App to communicate with multiple Intermediary Data Handlers in an enterprise setting.

[0009] Figure 3C a flow chart showing the process of a device switching the Intermediary Data Handler to an administrative mode.

[0010] Figure 4A shows the Intermediary Data Handler orchestrating data on an internal storage device.

[0011] Figure 4B shows the Intermediary Data Handler orchestrating data on an external storage device.

[0012] Figure 5 shows the intermediary data handler converting the data from the operating system to the storage medium and vice versa.

[0013] Figure 6A shows how the Intermediary Data Handler misreports the actual data structure of the storage device to mimic the structure that the OS is expecting to see.

[0014] Figure 6B a flow chart showing, from the perspective of the Intermediary Data Handler, how a User would change the operating system emulation.

[0015] Figure 7A a flow chart showing the operating sending a write request to the storage device.

[0016] Figure 7B a flow chart showing the operating sending a read request to the storage device.

Description of the Invention

[0017] Referring now to the drawings, the details of preferred embodiments of the present invention are graphically and schematically illustrated. Like elements in the drawings are represented by like numbers, and any similar elements are represented by like numbers with a different lower case letter suffix.

[0018] The heart of the present Secured Data Storage Subsystem (SDSS) is an Intermediary Data Handler **10**. In the typical electrical computer system illustrated in Figure 1A, read/write instructions are communicated by the Operating System ("OS") directly to the computer's memory storage device **200**. The memory storage device in turn acts on the instructions and writes to or reads from the storage device memory **210**, communicated a reply communication to the OS with the result of the read/write call. Thus, in the typical computer system, the memory storage device **200** operates on any read/write call from the OS. In contrast, as shown in Figure 1B, a computer system enabled with the present invention, the OS does not directly communicate with the computer's memory storage device **200**, but with the present Intermediary Data Handler **10**. In like fashion, the computer's memory storage device **200** does not communicate directly with the OS, but only through the Intermediary Data Handler **10**. In other words, in practicing the present invention, all read/write calls from the OS are first processed by the Intermediary Data Handler **10**.

[0019] Functionally, the Intermediary Data Handler **10** sits between the Operating System **160** of a Computing Device **100** and the Storage Device Memory **210** of the Storage Device **200**. See Figure 1B. As the Computing Device **100** sends calls/requests to the Storage Device **200** via the communication Layer **110** they are intercepted by the Intermediary Data Handler **10** via the Layer **220**. The Intermediary Data Handler **10** then, independent of the Operating System **160**, converts the Data **40** into its own format and stores it via the Layer **230**. The Intermediary Data Handler **10** applies its own rule set to how the Data **40** is stored, the format it is stored in and how it can be accessed and manipulated. The Intermediary Data Handler **10** will misreport the Data **40** structure to the Operating

System **160** as to make the technology completely transparent to the Operating System **160** and the end User. By maintaining control of the Data **40** with hardware that acts independent of User and Operating System **160** instructions, the Intermediary Data Handler **10** can prevent malicious code from effecting meaningful changes to the Operating System **160** files while at the same time ensuring the integrity of User files. Hackers cannot compromise the SDSS/Intermediary Data Handler software **178**, because Data Handler Software **178** acts independently of all other software on the computing device **100** with which it is associated.

[0020] The structure of the Intermediary Data Handler **10** should be sufficient that it can maintain absolute autonomy from all other components of the Computing Device **100** (see Figure 2). The hardware inserted between Layer **220** and Layer **230** comprises Processor circuitry (e.g., a CPU) **170**, Memory circuitry (e.g., RAM) **175**, and in some embodiments: BIOS/ROM **180** circuitry and User Communication Hardware **185**. The data handler **10** includes SDSS/Data Handler Software **178**, which is adapted to enable the Data Handler **10** and its hardware/firmware components to carry out its functions.

[0021] The User Communication Hardware **185** of the Intermediary Data Handler **10** is used to communicate with a wireless User Device **120** or an Admin Device **190** (see Figure 3A to 3C). A User Device **120** with a User App **130** or an Admin Device **190** with an Admin App **195** can be used to establish a Wireless communication session **140** with the Intermediary Data Handler **10**, which is preferably onboard the Storage Device **200**. The Wireless session **140** allows the User App **130** or the Admin App **195** to provide the end User with various options such as restoring data, switching to and from and administrative mode, maintenance options, emulation options, etc.

[0022] As shown in Figure 3C, a User Device **120** or an Admin Device **190** can switch the Intermediary Data Handler **10** to an Administrative Mode **800**. Switching into Administrative Mode **800** allows the Intermediary Data Handler **10** full read and write access to the Storage Device **200** with which it is associated. When initiating the Administrative Mode Start process **805**, a decision **810** is made

to detect whether the Intermediary Device Handler **10** is in an administrative mode. If "Yes," then a decision is made to determine if the User has valid authentication **820**. If "Yes," then the SDSS allows full Read/Write Access **830**. Upon Session Termination **840** the Intermediary Data Handler **10** device returns the SDSS to its normal/secure mode **850** and the Intermediary Data Handler **10** exits **860** the Administrative Mode **800**.

[0023] The Intermediary Data Handler **10** segregates the data sent to the Storage Device **100** into two separate file structures via Layer **15** and Layer **25** (see Figure 4A). One structure is the Generated Data **20** structure, which is incoming data to be stored on the storage device memory **210**. This is the structure where all data that is sent to the Storage Device **200** is directed by default. The other structure is the Admin Only Files **30**. This is where important files, which should not be altered without the User's express instructions, are stored. Files in the Admin Only Files **30** section can only be altered if the User switches the Intermediary Data Handler **10** into administrative mode. If the Computing Device **100** sends an instruction to the storage device telling it to alter a file that resides in the Admin Only Files **30** section and the Intermediary Data Handler **10** is not in an administrative mode, then the instruction is disregarded, misreported, or redirected to the Generated Data **20** section. By employing this methodology the Intermediary Data Handler **10** can maintain the integrity of the Admin Only Files **30** data and prevent malicious instructions from committing any offense. The same can be accomplished by using an external storage device using a USB **150** connection (see Figure 4B).

[0024] The Intermediary Data Handler **10** uses the Data Conversion **50** to convert to and from its dual structured format to one contiguous format that is appropriate to the Operating System **160** it is working with (see Figure 5). The Data **45** coming from the Operating System **160** passes through the Data Conversion **50** and is transformed into Data **40** before being stored. The same is true in reverse. Data **40** coming from the Storage Device **200** passes through the Data Conversion

50 implemented by the Intermediary Data Handler **10** and is made to mimic the appropriate Data **45** structure that is native to the Operating System **160**.

[0025] Mimicking a file system allows the Data **40** on a Storage Device **200** to be acceptable to another system without making any actual partition or formatting changes and still be perceived as native by the host system (see Figures 6A and 6B). By being able to mimic an Operating System **160** Data **45** structure, the Storage Device **200** can be placed into a Computer **100** with Linux **160a**, Windows **160b**, OS X **160c**, etc., and be accepted by the Operating System **160** as being a device with a native data structure such as **45a**, **45b** and **45c**. This allows the Storage Device **200** to be move from one Computing Device **100** to another Computing Device **100** with a different Operating System **160** without having any compatibility issues.

[0026] Figure 6B exemplifies the Change OS Emulation Process **500** for changing which OS **160** is being emulated by the SDSS. The OS Emulation Selection Process **500** when started **505** first does a Connectivity Check **510** to determine whether a User Device **120** is connected to the Intermediary Data Handler **10**. If "Yes," then the Selection Process **500** checks to see if the User Device **120** is requesting an operating system change **520**. If "Yes", then the OS Emulation Selection Process **500** implements the requested emulation change **530**. If Terminate Session **540** decision is subsequently made, then End Session **550** procedure is implemented.

[0027] The Intermediary Data Handler **10** facilitates requests from the Operating System **160** to write information to the Storage Device **200** following a simple base logic (see Figure 7A). When the Received Write Request Process **300** is started the Intermediary Data Handler **10** receives a Write Request **310** from the operating system **160**. The SDSS checks to see if the Intermediary Data Handler **10** is in Administration Mode **320**. If the answer is "Yes," then the Intermediary Data Handler **10** will allow full write privileges **340**. If the answer is "No," then the Intermediary Data Handler **10** will Restrict **330** the Write Request to the Generated Data **20** zone. The Intermediary Data Handler **10** will Convert Data **350** to appropriate Converted Data Structure **40**. The Converted Data Structure **40** is

then Written **360** to the Storage Device **200** and the Write Request Process **300** is Ended **370**.

[0028] Read Requests received from the OS **160** are processed following a similar logic: Received Read Request Process **400** (see figure 7B). The Read Request Process **400** is Started **405** when the Intermediary Data Handler **10** detects a Read Request **410** from the Operating System **160**. The Intermediary Data Handler **10** then proceeds to Find Read Data **420** on the Storage Device **200**. The Intermediary Data Handler **10** then Converts **430** the Found Data to the OS Specific Data Structure **45** compatible with the current Operating System **160** platform, and Fulfills the Read Request **440**, after which the Received Read Request Process **400** is Ended **450**.

[0029] While the above description contains many specifics, these should not be construed as limitations on the scope of the invention, but rather as exemplifications of one or another preferred embodiment thereof. Other variations are possible, which would be obvious to one skilled in the art. Accordingly, the scope of the invention should be determined by the scope of the appended claims and their equivalents, and not just by the embodiments.

WHAT IS CLAIMED IS:

Claims

1. An operating system independent Secured Data Storage Subsystem (SDSS) for use with an electrical computer/digital data processing apparatus, the SDSS comprising:

- a data storage device of said electrical computer/digital data processing apparatus in indirect communications with the operating system;
- an intermediary data handler has a local processing unit, a local buffer memory, a local data storage memory, and a user communication circuit, with the intermediary data handler being functionally disposed in communications between the operating system of said electrical computer/digital data processing apparatus and a storage device memory of the computer data storage device; and
- an SDSS/Data Handler Software/Instruction Set is resident in the local data storage memory and executable by the intermediary data handler, and the SDSS/Data Handler Software/Instruction Set enabling the intermediary data handler to carry out its functions, being adapted to intercept a read/write request from the operating system and to generate mock response data to the read/write request, the mock response data being in a data-structure format that is expected by the operating system in a reply from the data storage device, while separately preventing modification of data on the storage device to ensure the integrity of the data in the storage device memory.

2. An operating system independent Secured Data Storage System (SDSS) for use with an electrical computer/digital data processing apparatus having a data storage device, the SDSS comprising:
- an intermediary data handler device having processor, memory and user communication circuitries, and the intermediary data handler functionally disposed in communications between the operating system of said electrical computer/digital data processing apparatus and a storage memory of the data storage device,
- an SDSS/Data Handler Software/Instruction Set, resident on and executable by the intermediary data handler, the data SDSS/Data Handler Software/Instruction Set adapted in response to a read/write communication from the operating system to

generate a mock data reply, the mock data reply having a content and data-structure format acceptable by the operating system in a reply from the data storage device, while isolating and maintaining control over the original read/write communication from the operating system; and

- the SDSS/Data Handler Software/Instruction Set further adapted to enable the SDSS to accomplish the security of data stored in the storage memory of the data storage device.

3. An intermediary data handler for use with a Secured Data Storage Subsystem (SDSS), the intermediary data handler comprising:

- a processor circuit, a memory circuit, and a user interface circuit, the intermediary data handler functionally disposed in communications between an operating system of a host electrical computer/digital data processing apparatus and a storage memory of a data storage device; and

- a Data Handler Software/Instruction Set, resident on and executable by the intermediary data handler, the Data Handler Software/Instruction Set adapted to generate a mock data reply in response to a read/write communication from the operating system, the mock data reply having a content and data-structure format acceptable by the operating system in a reply from the data storage device, while isolating and separately maintaining control over the original read/write communication from the operating system, and further adapted to integrate operation and function of the intermediary data handler with the host electrical computer/digital data processing apparatus to accomplish the security of data stored in the storage memory of the data storage device.

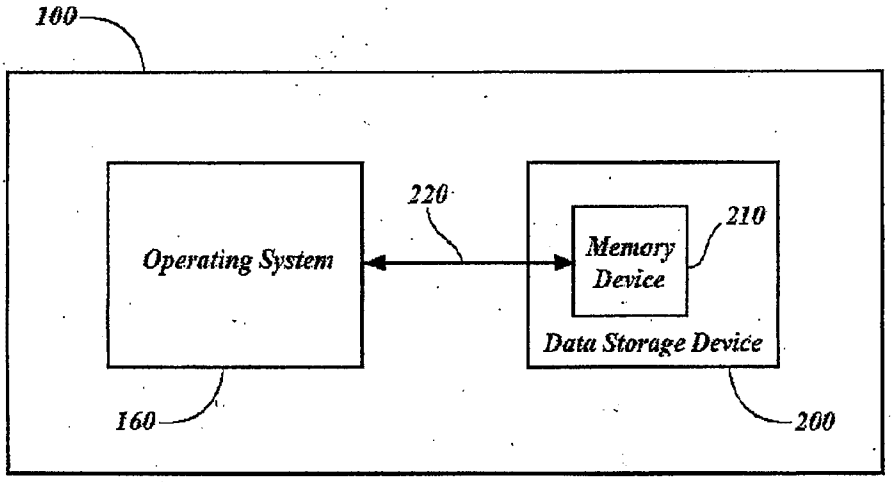


Figure 1A
PRIOR ART

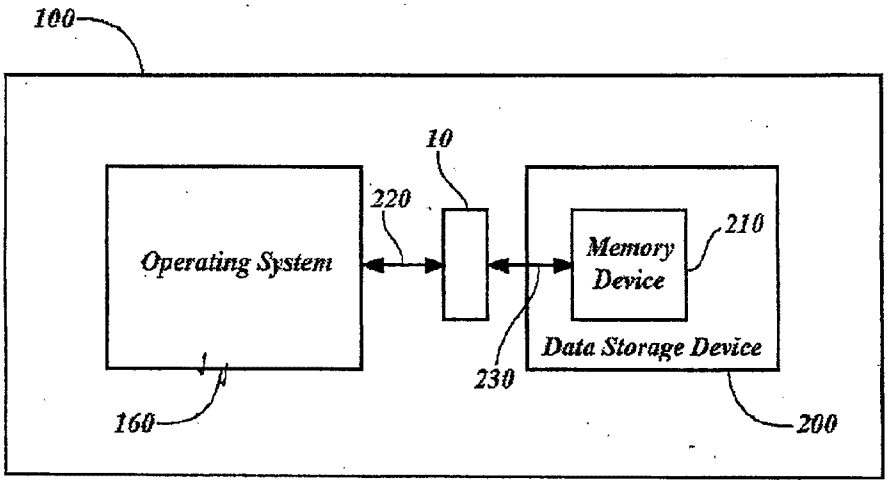


Figure 1B

Figure 2

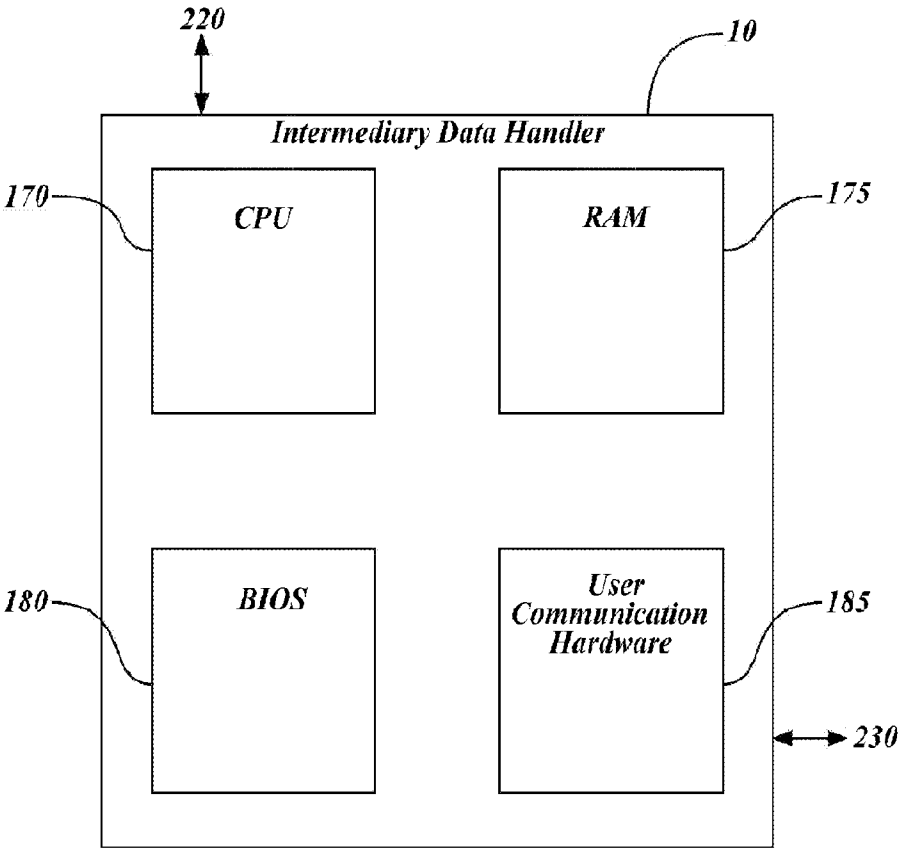

220
10
Intermediary Data Handler
170
CPU
RAM
175
BIOS
180
User Communication Hardware
185
230

Figure 3A

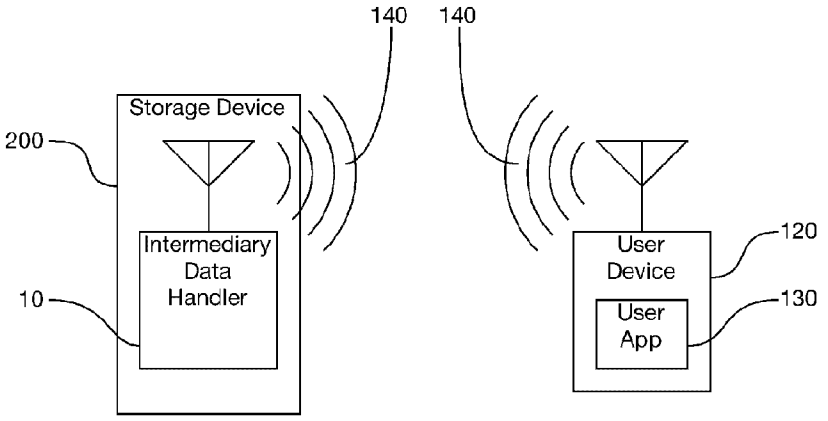

140
140
Storage Device
200
Intermediary
Data
Handler
10
User
Device
120
User
App
130

Figure 3B

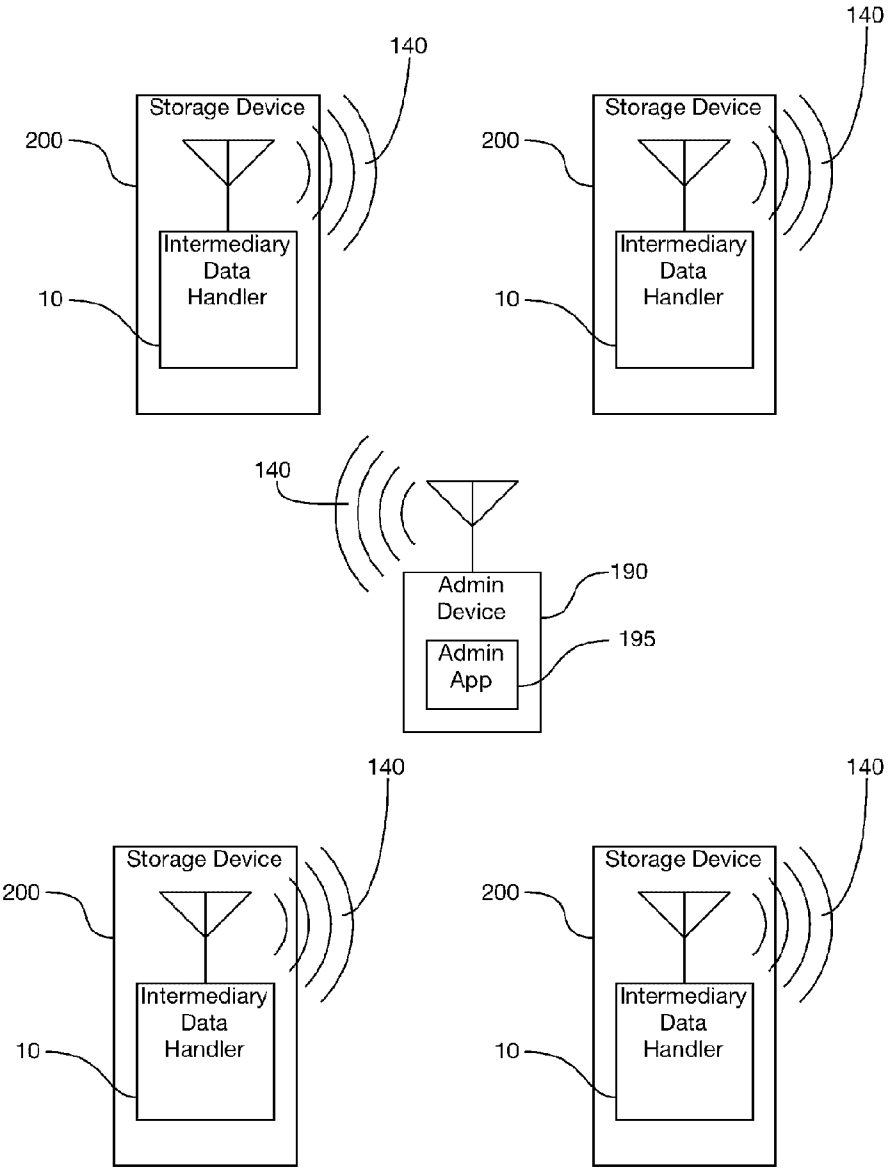

140
Storage Device
200
Intermediary
Data
Handler
10
140
Storage Device
200
Intermediary
Data
Handler
10
140
Admin
Device
190
Admin
App
195
140
Storage Device
200
Intermediary
Data
Handler
10
140
Storage Device
200
Intermediary
Data
Handler
10

Figure 3C

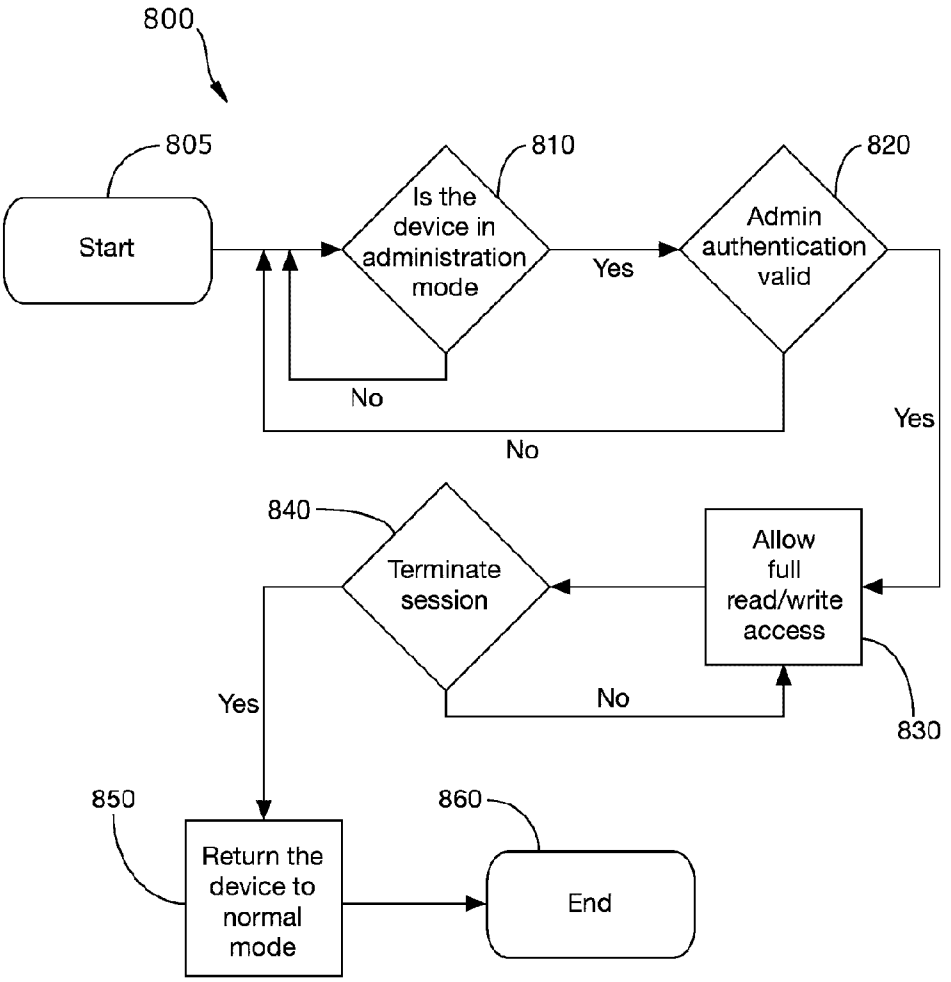

800
805
Start
810
Is the device in administration mode
Yes
No
820
Admin authentication valid
Yes
No
840
Terminate session
Yes
Allow full read/write access
No
830
850
Return the device to normal mode
860
End

Figure 4A

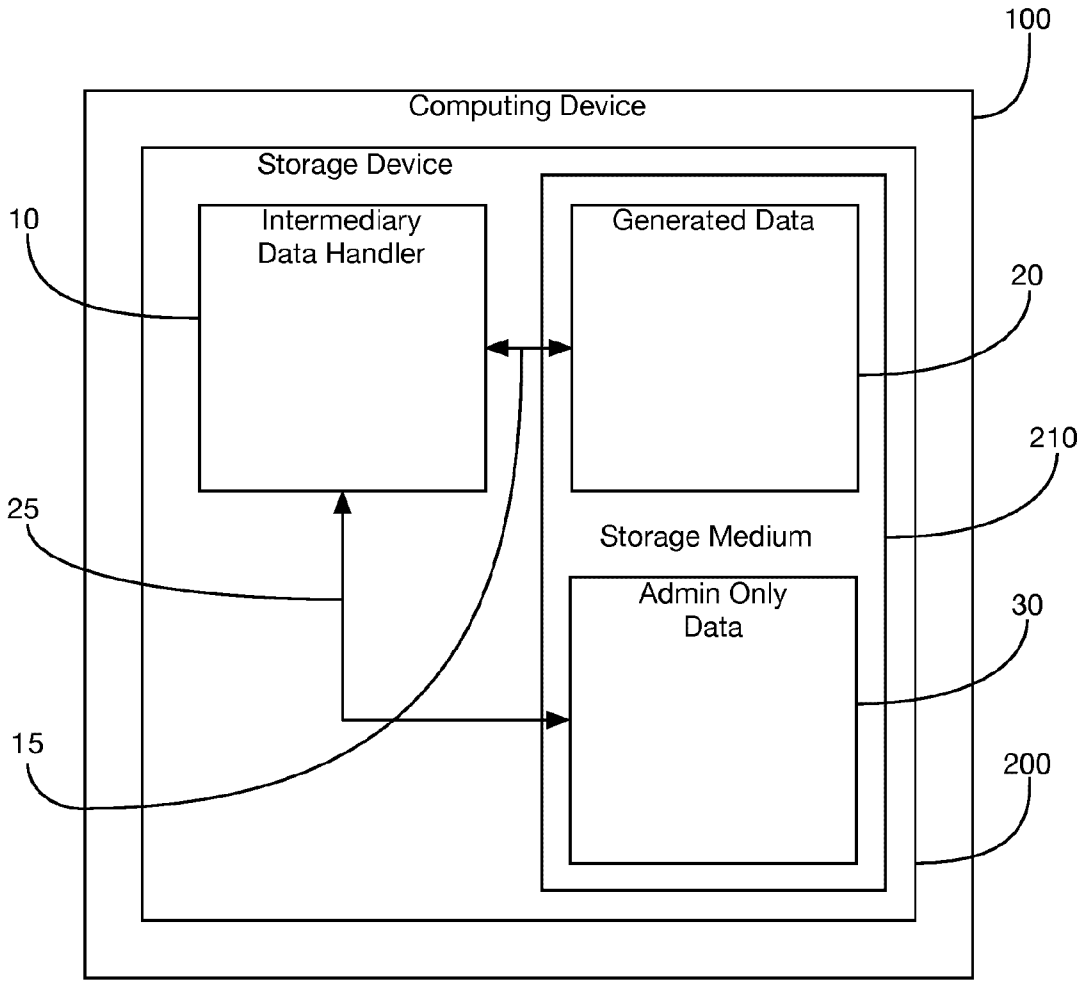

100
Computing Device
Storage Device
10
Intermediary
Data Handler
Generated Data
20
210
25
Storage Medium
Admin Only
Data
30
15
200

Figure 4B

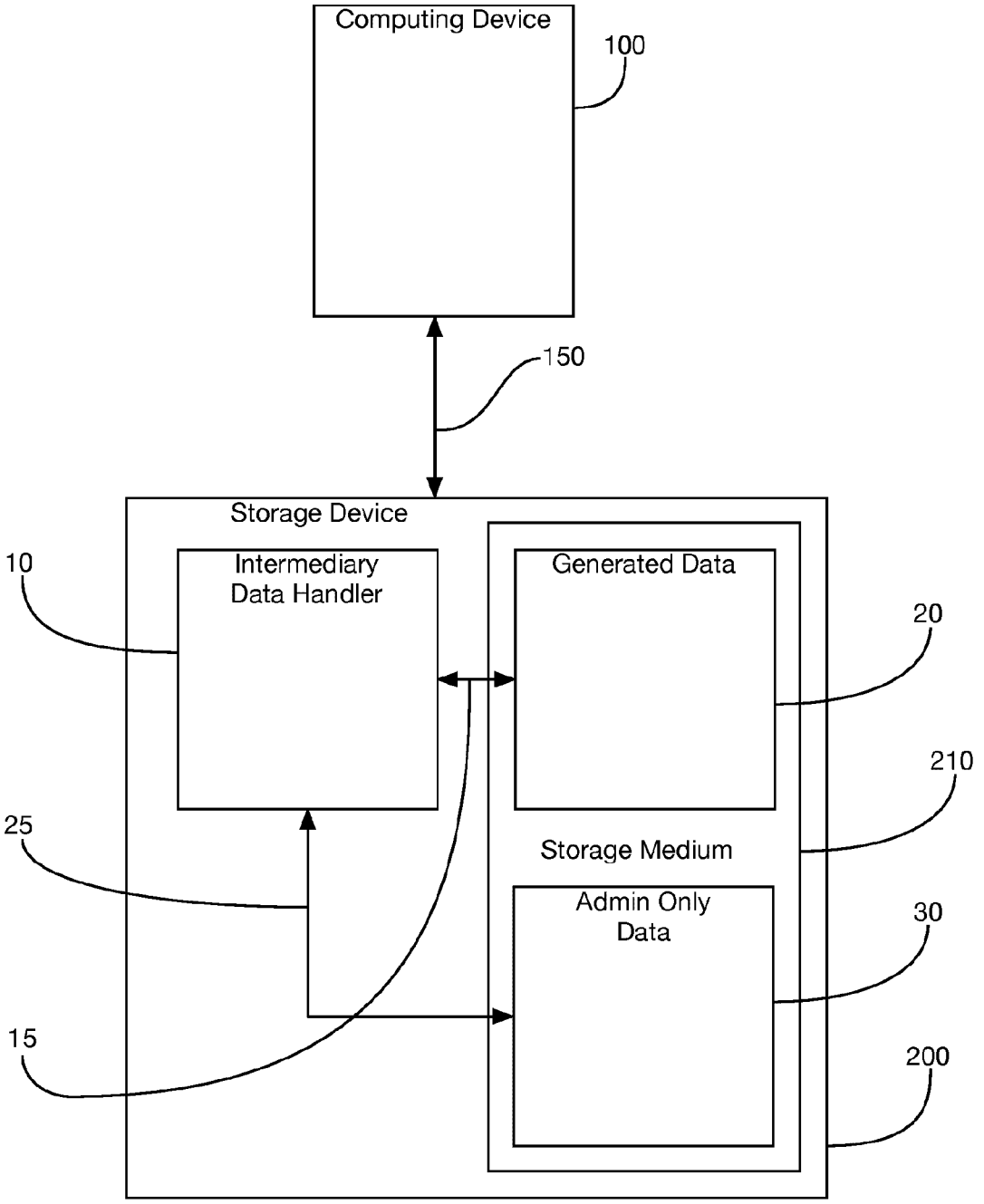

Computing Device
100
150
Storage Device
10
Intermediary
Data Handler
Generated Data
20
210
25
Storage Medium
Admin Only
Data
30
15
200

Figure 5

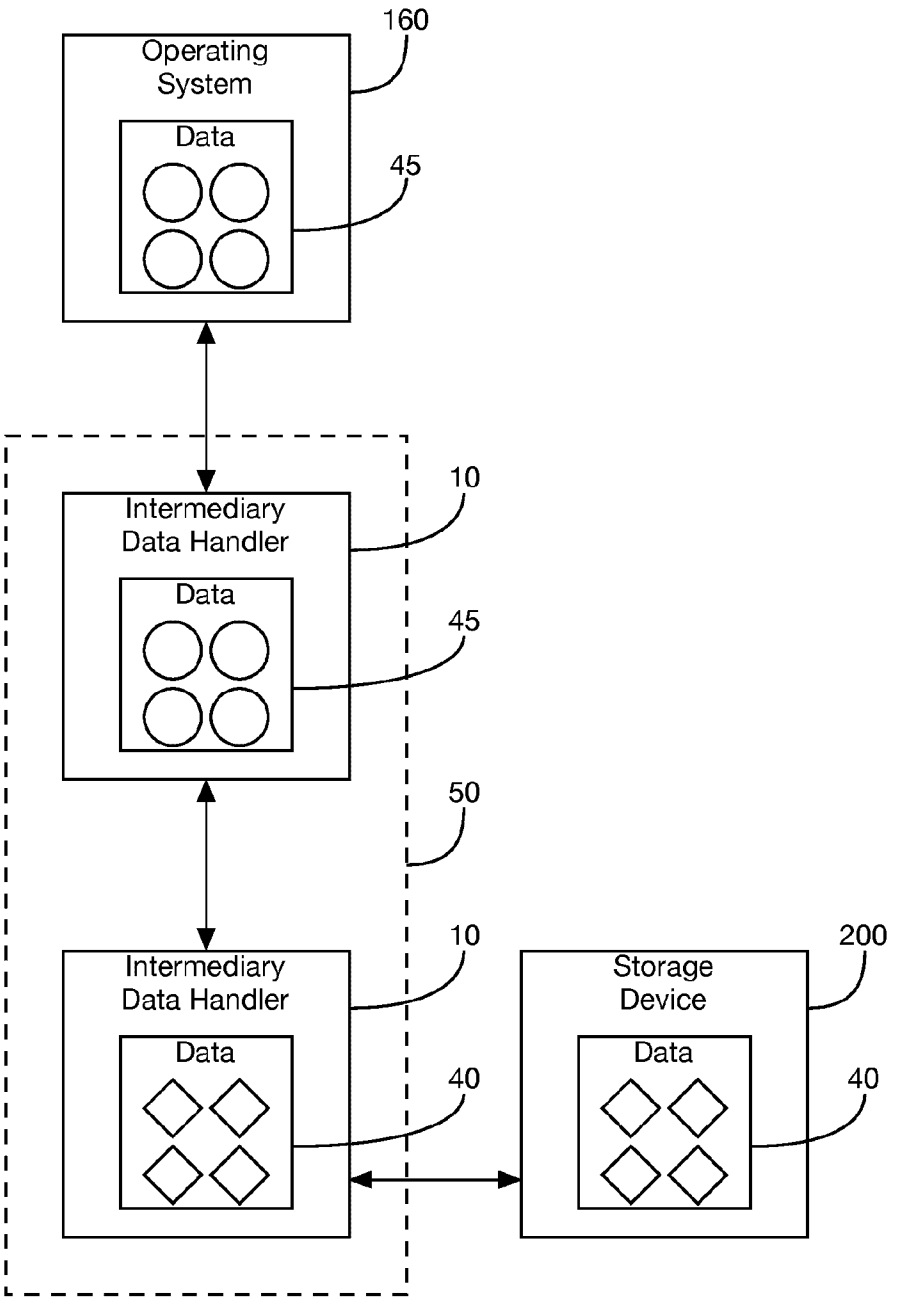

160
Operating System
Data
45
10
Intermediary Data Handler
Data
45
50
10
Intermediary Data Handler
Data
40
200
Storage Device
Data
40

Figure 6A

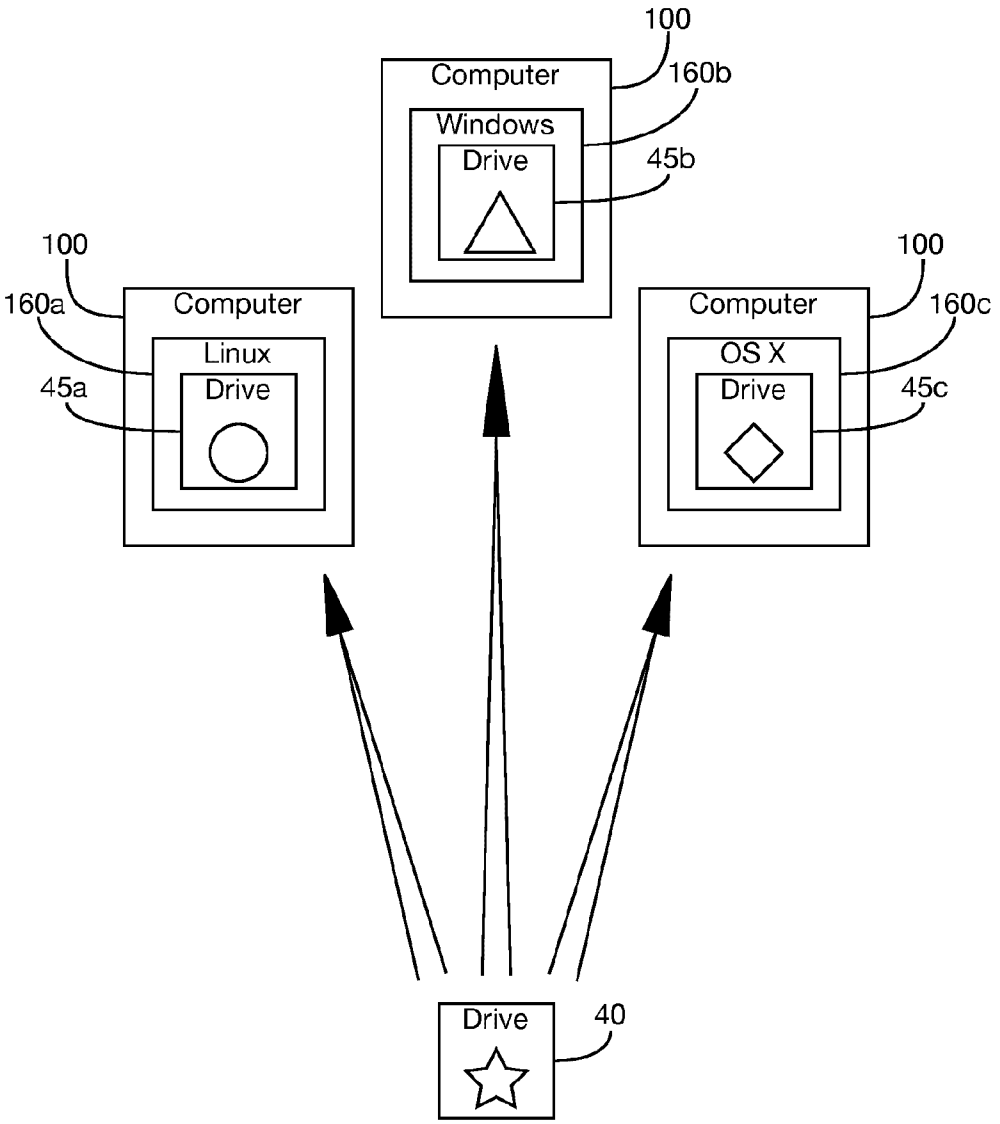

100
Computer
160b
Windows
Drive
45b
100
160a
Computer
Linux
Drive
45a
100
Computer
160c
OS X
Drive
45c
Drive
40

Figure 6B

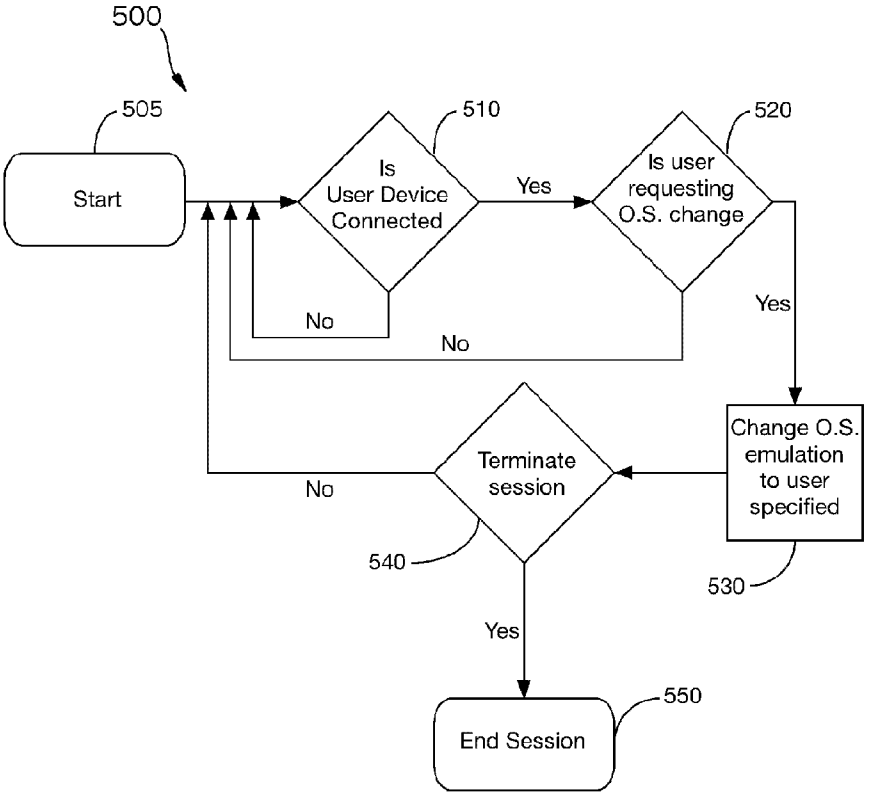

500
505
Start
510
Is
User Device
Connected
Yes
520
Is user
requesting
O.S. change
Yes
No
No
Change O.S.
emulation
to user
specified
530
Terminate
session
No
540
Yes
550
End Session

Figure 7A

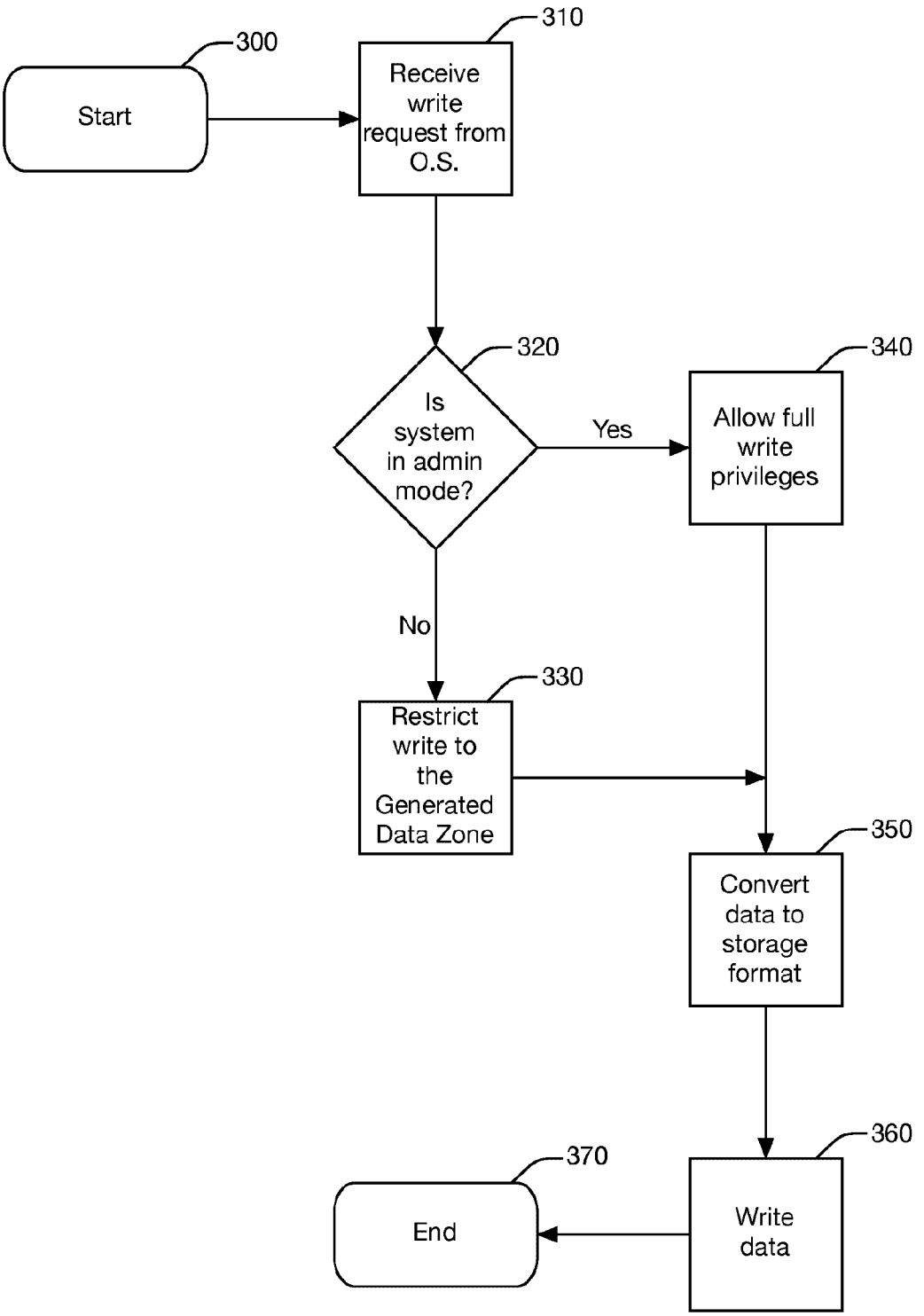
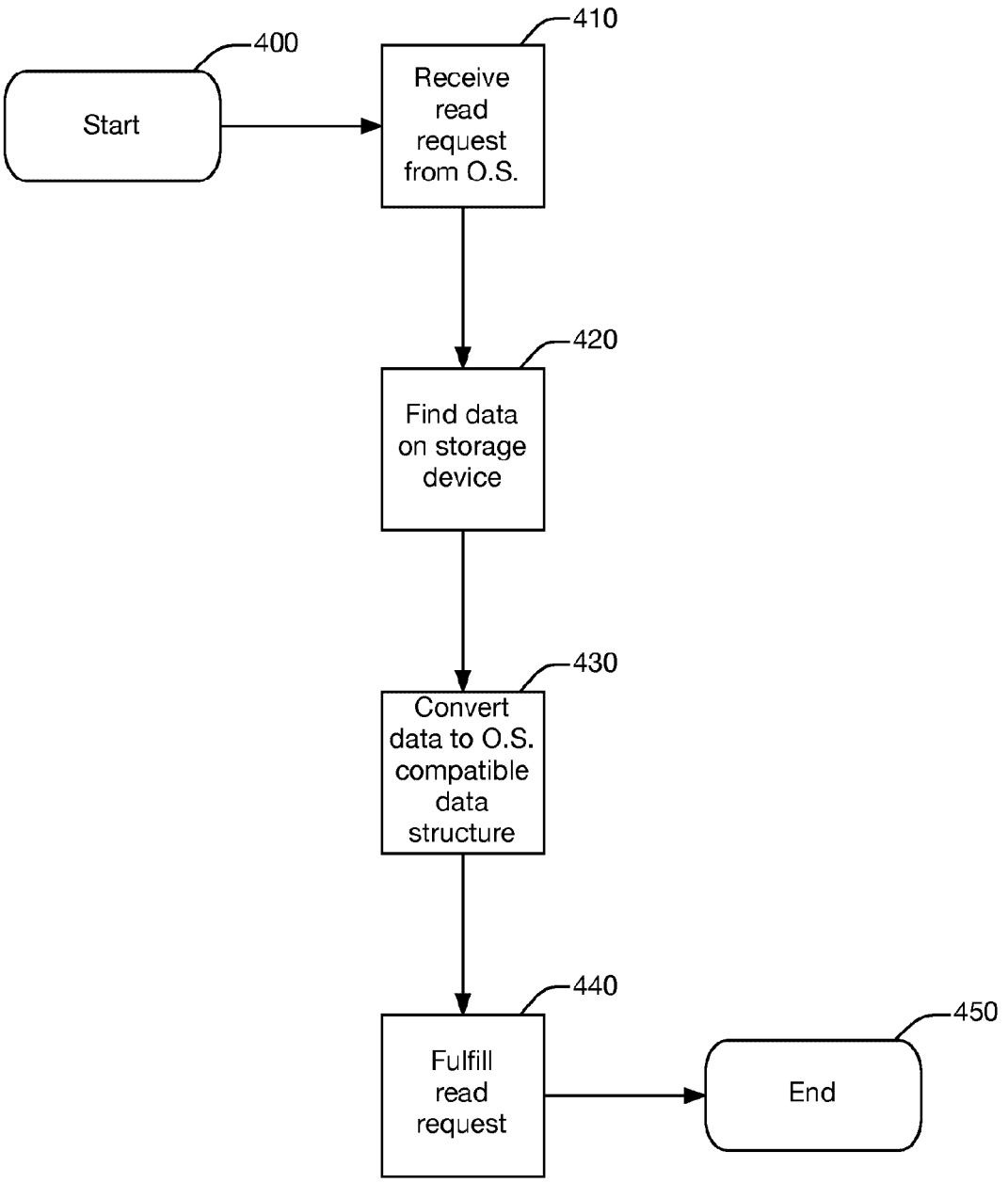
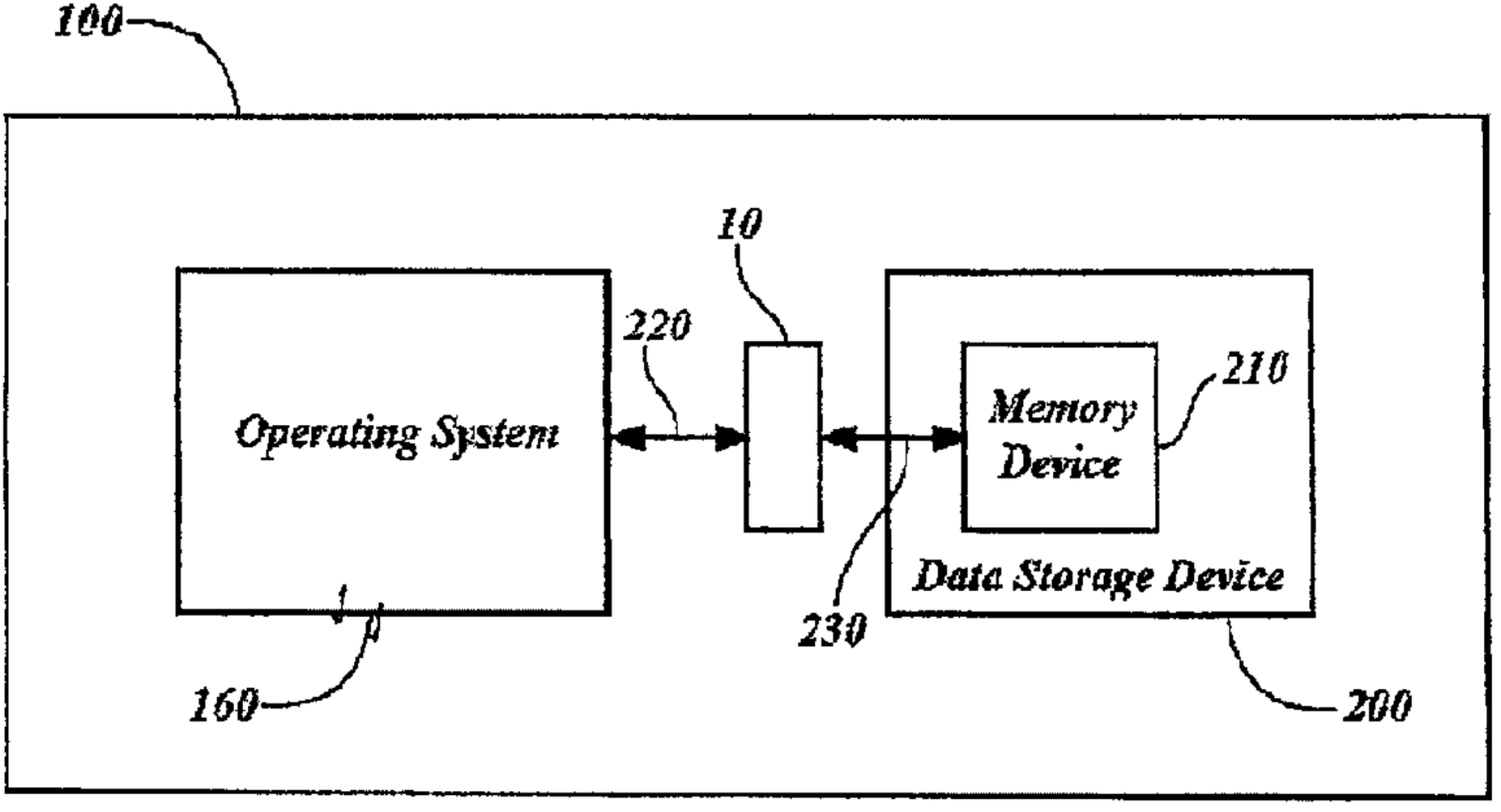

300
Start
310
Receive write request from O.S.
320
Is system in admin mode?
Yes
340
Allow full write privileges
No
330
Restrict write to the Generated Data Zone
350
Convert data to storage format
360
Write data
370
End

Figure 7B


400
Start
410
Receive read request from O.S.
420
Find data on storage device
430
Convert data to O.S. compatible data structure
440
Fulfill read request
450
End


100
10
220
Operating System
Memory Device
210
Data Storage Device
230
160
200