



(43) International Publication Date
27 September 2012 (27.09.2012)

(51) International Patent Classification:

G06F 15/16 (2006.01) G06F 9/44 (2006.01)
G06F 13/14 (2006.01)

(21) International Application Number:

PCT/US2012/028667

(22) International Filing Date:

11 March 2012 (11.03.2012)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

13/052,757 21 March 2011 (21.03.2011) US

(71) Applicant (for all designated States except US): **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **LUPU, Corneliu I.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **CALINOIU, Silviu C.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **PETRUTA, Cristian G.**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **JARGALSAIHAN, Batsaihan**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **MATICHUK, Chris Ernest**; c/o Mi-

crosoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (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, 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, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, 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))

[Continued on next page]

(54) Title: DEVICE FLAGS

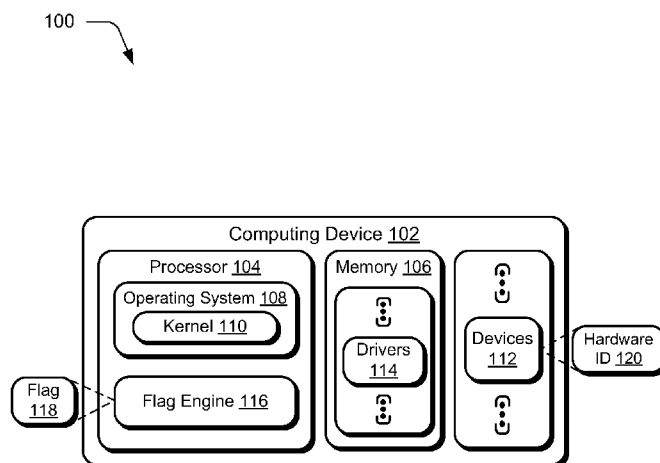


Fig. 1

(57) Abstract: Device flag techniques are described. In one or more implementations, responsive to detection by a computing device that a device has been communicatively coupled to bus of the computing device, a database is queried by the computing device to determine whether one or more flags have been set for the device. Responsive to a determination made as a result of the querying that the one or more flags have been set for the device identifier, information associated with the one or more flags is communicated to software that is associated with the device to configure the software.





Published:

- *without international search report and to be republished
upon receipt of that report (Rule 48.2(g))*

Device Flags

BACKGROUND

10 [0001] Computing devices may include a variety of different hardware devices to expand the functionality available to the computing device. For example, the hardware devices may include internal devices that may be configured within a housing of the computing devices as well as external devices, such as printers, cameras, and so on. Drivers are typically employed to enable communication between an operating system of the computing device and the devices.

[0002] The continuing development of software executed on the computing device, however, may cause changes in how the software interacts with the drivers. Consequently, a driver that is compatible with a previous version of an operating system, for example, may not be compatible with later versions of the operating system. Traditional techniques that were used to combat this incompatibility included writing a newer version of the driver. Although this may restore functionality of the device, the writing of the driver may involve a wait until the driver is available for dissemination, 25 involve a user locating the newer version of the driver, and so on that could lead to user frustration and lost productivity.

SUMMARY

[0003] Device flag techniques are described. In one or more implementations, responsive to detection by a computing device that a device has been communicatively coupled to bus of the computing device, a database is queried by the computing device to determine whether one or more flags have been set for the device. Responsive to a determination 30 made as a result of the querying that the one or more flags have been set for the device identifier, information associated with the one or more flags is communicated to software that is associated with the device to configure the software.

[0004] In one or more implementations, a computing device detects that a device has been communicatively connected to a bus of the computing device and a device identifier of the device is obtained by a bus driver that is executed on the computing device and manages 35 the bus. The device identifier of the device is communicated by the bus driver to an engine that is executable on the computing device. A database is queried by the engine to determine whether one or more flags have been set for the device identifier and responsive to a determination that the one or more flags have been set for the device identifier,

information is communicated that relates to the one or more flags to the bus driver and the information is applied by the bus driver.

[0005] In one or more implementations, usage telemetry data is received via a network at one or more computing devices of a network service, the usage telemetry data identifying
5 firmware of devices, corresponding drivers via which the devices communicate, and configuration of the drivers. The usage telemetry data is processed at one or more computing devices to determine which of the devices described in the usage telemetry data include firmware that involves an update to enable operation of functionality of the device in conjunction with a respective computing device that otherwise would not be enabled.

10 [0006] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

15 [0007] The detailed description is described with reference to the accompanying figures. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The use of the same reference numbers in different instances in the description and the figures may indicate similar or identical items.

[0008] FIG. 1 is an illustration of an environment in an example implementation that is
20 operable to perform device flag techniques.

[0009] FIG. 2 is an illustration of a system in an example implementation showing interaction between a device, driver, and flag engine of FIG 1.

[0010] FIG. 3 is a flow diagram depicting a procedure in an example implementation in which connection of a device to a computing device is detected and initiates a process to
25 check as to whether a flag is set for the device.

[0011] FIG. 4 is a flow diagram depicting a procedure in an example implementation in which a query is performed to determine whether a flag has been set for a particular device.

[0012] FIG. 5 is a flow diagram depicting a procedure in an example implementation in
30 which telemetry data generated as a result of configuration of a driver by one or more flags is leveraged.

DETAILED DESCRIPTION

Overview

[0013] Computing devices may include (e.g., be communicatively coupled to) a wide variety of different devices such that functionality of the device may be utilized by the computing device. For example, a computing device may be communicatively coupled to a printer, optical disc reader/writer, thumb drive, or other peripheral device that may be internal (e.g., within a housing) or external to the computing device. A driver may be utilized to enable communication between the computing device and the device, such as to translate commands and data from an operating system of the computing device into a form that is understandable by the device and vice versa. However, changes to the operating system may cause incompatibilities with the driver and consequently with the device.

[0014] Device flag techniques are described which may be leveraged to maintain compatibility, such as between software of the computing device (e.g., an operating system) and a device and driver of the device. This compatibility may be provided in a variety of ways. For example, a flag engine (e.g., which may be implemented as part of a shim engine) may maintain a database of hardware IDs and corresponding flags. Therefore, when a device is detected as being connected to the computing device (e.g., by a bus driver), the flag engine may be queried to determine if there are any flags set for the device ID. Information relating to this flag may then be communicated to the driver to configure the driver accordingly, such as for code path selection, enable device specific capabilities, and so on, further discussion of which may be found in relation to the following sections.

[0015] In the following discussion, an example environment is first described that may be leveraged according to driver shimming techniques. Example procedures are then described which may also be employed in the example environment as well as other environments. Accordingly, performance of the example procedures is not limited to the example environment and the example environment is not limited to performing the example procedures.

Example Environment

[0016] FIG. 1 is an illustration of an environment 100 in an example implementation that is operable to employ techniques described herein. The illustrated environment 100 includes a computing device 102, which may be configured in a variety of ways. For example, the computing device 102 may be configured as portable game device, mobile

phone, a computer that is capable of communicating over a network (e.g., a desktop computer, one or more servers, an entertainment appliance), a set-top box communicatively coupled to a display device, and so forth. Thus, the computing device 102 may range from full resource devices with substantial memory and processor resources (e.g., personal computers, game consoles) to a low-resource device with limited memory and/or processing resources (e.g., traditional set-top boxes, hand-held game consoles). Additionally, although a single computing device 102 is shown, the computing device 102 may be representative of a plurality of different devices, such as multiple servers utilized by a business to perform operations, a remote control and set-top box combination, and so on.

[0017] The computing device 102 may also include an entity (e.g., software) that causes hardware of the computing device 102 to perform operations, e.g., processors, functional blocks, and so on. For example, the computing device 102 may include a computer-readable medium that may be configured to maintain instructions that cause the computing device, and more particularly hardware of the computing device 102 to perform operations. Thus, the instructions function to configure the hardware to perform the operations and in this way result in transformation of the hardware to perform functions. The instructions may be provided by the computer-readable medium to the computing device 102 through a variety of different configurations.

[0018] One such configuration of a computer-readable medium is signal bearing medium and thus is configured to transmit the instructions (e.g., as a carrier wave) to the hardware of the computing device, such as via a network. The computer-readable medium may also be configured as a computer-readable storage medium and thus is not a signal bearing medium. Examples of a computer-readable storage medium include a random-access memory (RAM), read-only memory (ROM), an optical disc, flash memory, hard disk memory, and other memory devices that may use magnetic, optical, and other techniques to store instructions and other data.

[0019] The computing device 102 is also illustrated as including a processor 104 and memory 106. Processors are not limited by the materials from which they are formed or the processing mechanisms employed therein. For example, processors may be comprised of semiconductor(s) and/or transistors (e.g., electronic integrated circuits (ICs)). In such a context, processor-executable instructions may be electronically-executable instructions. Alternatively, the mechanisms of or for processors, and thus of or for a computing device, may include, but are not limited to, quantum computing, optical

computing, mechanical computing (e.g., using nanotechnology), and so forth. Additionally, although a single processor 104 and memory 106 are shown, a wide variety of types and combinations of memory and/or processors may be employed.

[0020] The computing device 102 is illustrated as also includes an operating system 108
5 having a kernel 110 and one or more drivers 112 that are configured to support communication between the kernel 110 of the operating system 108 and one or more devices 114. The kernel 110 represents a component of the operating system 108 that is typically employed to abstract functionality of underlying devices such as the processor 108, memory 106, and other devices 112 to applications and other software that are
10 executed by the computing device 102. Although the operating system 108 and kernel 110 are illustrated as being executed on the processor 104, these modules are also storable in memory 106.

[0021] The devices 112 may be representative of a variety of different devices that may be employed by the computing device 102. For example, the devices 112 may include
15 peripheral devices, such as printers, scanners, hard drives, and so on. The devices 112 may also represent functionality of other computing devices that may be leveraged by the computing device 102, such as to use a mobile phone as a storage device, access photos on the mobile phone, and so on. Thus, the computing device 102 may leverage a variety of different devices 112 to add to the functionality of the computing device 102.

[0022] In order for the devices 112 to communicate with the computing device 102, and
20 more particularly the operating system 108 and kernel 110 of the computing device, one or more drivers 114 may be employed. Drivers 114 are typically employed to abstract functionality of a corresponding device 112. For example, the driver 114 may be configured as a series of layers to translate commands from the kernel 110 of the operating
25 system 108 into device specific commands that are “understandable” by a respective device.

[0023] A variety of different drivers 114 may be employed by the computing device 102. For example, the driver 114 may be configured as a class driver that is configured to interact with a number of different types of a device 112. For example, a class driver may
30 be configured for use with optical reader/writer, storage devices, networking, and so on. Another example involves a bus driver, such as a driver employed by a USB, PCI, or similar bus such that devices that share a common protocol (e.g., the USB protocol) may leverage the same driver to be compatible with the computing device 102. Other different types of drivers from a device stack (e.g., bus driver, class driver, filter driver, device

driver) may employ these techniques. Further, a variety of other software may also leverage these techniques, such as application to make contextual based code path selection based on flags and corresponding retrieval of data based on a device identifier.

[0024] As previously described, however, there may be some instances in which the driver 114 is not longer compatible with software of the computing devices, such as the kernel 110 of the operating system 108. For example, the operating system 108 may be updated to a newer version, have a service pack applied, and so on that may change how the operating system 108 is configured to communicate with the driver 114.

[0025] Accordingly, the computing device 102 may employ a flag engine 116 that is representative of functionality to employ one or more flags 118 to support compatibility of the drivers 114 with the operating system 108. For instance, the flag engine 116 may receive a hardware ID 120 of a device 112 when the device is communicatively coupled to the computing device 102, e.g., to a bus. The hardware ID 120 may then be used by the flag engine 116 to determine whether one or more flags 118 correspond to the hardware ID 120. Information relating to the flag 118 may then be leveraged by the driver 114 to support communication with the device 112. In this way, behavior of particular devices may be flagged and used as a basis to determine a “fix” to communicate with the device 114. This may be used to support a wide variety of functionality, an example of which is that the computing device 102 may employ a generic driver 114 that may be specifically configured for particular devices 112, further discussion of which may be found in relation to FIG. 2.

[0026] Generally, any of the functions described herein can be implemented using software, firmware, hardware (e.g., fixed logic circuitry), manual processing, or a combination of these implementations. The terms “module,” “engine,” and “functionality” as used herein generally represent hardware, software, firmware, or a combination thereof. In the case of a software implementation, the module, functionality, or logic represents instructions and hardware that performs operations specified by the hardware, e.g., one or more processors and/or functional blocks.

[0027] FIG. 2 illustrates an example implementation 200 of communication of components of the computing device 102 of FIG. 1 to employ flag techniques for driver compatibility. In this example, a device 112 is initiates a communicative connection with a bus 202 of the computing device 102 of FIG. 1. A bus driver 204 may detect this connection and provide a device ID 120 that identifies the device 112. For example, the

device ID 120 may identify particular hardware, firmware, and so on of the device 112. The device ID 120 may also be based on functionality supported by the device 112.

[0028] The flag engine 116 may then receive this device ID 120 and determine whether one or more behaviors of the device 112 have been flagged by examining a database 208.

5 The flag engine 116, for instance, may determine whether a flag 118 corresponds to the device ID 120. In the illustrated example the database 208 is illustrated as being local to the computing device 102 but other instances are also contemplated, such as available at a network service 210 accessible via a network 212.

[0029] If the device ID 120 has a corresponding flag 118 in the database 208, the flag
10 engine 116 may provide corresponding information 214 back to the bus driver 204. For example, the information 214 may describe the flag itself, such as that “bit 5 is set” meaning that a particular power command is not to be given to the device 112 by the bus driver 204. In another example, the flag 118 may be associated with a payload 216 that contains more detailed information that may be used to configure the bus driver 204.

15 [0030] The device 112, for instance, may be configured in accordance with a standard and therefore include standardized behaviors 218, i.e., behaviors that are consistent with the standard. The device 112 may also include extended behaviors 220 that are not included in the standard, such as additional features that may be added to future standards and/or help the device 112 to differentiate itself in the marketplace. By using the techniques
20 described herein, however, these extended behaviors 220 may be supported by the bus driver 204 through use of the flags 118 and/or payloads 216 to configure the bus driver 204 for the extended behaviors 220. This configuration may be supported in a variety of ways, such as to use the flag 118 for code path selection by the bus driver 204, such the payload 216 for device specific capabilities, and so on.

25 [0031] The flag engine 116 may also provide telemetry data to the network service 210, which may be leveraged for a variety of different features. As illustrated in FIG. 2, for instance, the network service 210 is illustrated as including a telemetry module 222 that is representative of functionality to process telemetry data received from the flag engine 116. For example, the telemetry module 222 may be leveraged such that developers of software
30 or devices 112 could indicate whether functionality of a previous versions of software (e.g., the operating system 108) are to be employed to communicate with the device 112 and indicate this using a flag 118.

[0032] The telemetry data, for instance, may include details such as the device ID 120 and a context for events that occur in relation to the bus driver 204 and device 112. This usage

data may then be used to address these events, such as to determine compatibility test requirements for introducing an update, identify particular devices in a market that are to involve an update, determine whether devices have adapted to a later version and thus a flag is no longer relevant (e.g., firmware of the device 112 has been updated to address an updated version of an operating system 108), and so on. Further discussion of these and other techniques may be found in relation to the following procedures.

Example Procedures

[0033] The following discussion describes shared resource access verification techniques that may be implemented utilizing the previously described systems and devices. Aspects of each of the procedures may be implemented in hardware, firmware, or software, or a combination thereof. The procedures are shown as a set of blocks that specify operations performed by one or more devices and are not necessarily limited to the orders shown for performing the operations by the respective blocks. In portions of the following discussion, reference will be made to the environment 100 of FIG. 1 and the system 200 of FIG. 2.

[0034] FIG. 3 depicts a procedure 300 in an example implementation in which connection of a device to a computing device is detected and initiates a process to check as to whether a flag is set for the device. A computing device detects that a device has been communicatively connected to a bus of the computing device (block 302). A bus driver, for instance, may execute on the computing device 102 and detect that a device 114 has been connected to the bus 202 during runtime. In another instance, a class driver may be initiated that corresponds to the device 114 being connected, such as a network call driver for a network device, storage class driver for a storage device, applications may be specifically configured (e.g., and therefore include driver-like functionality), and so on.

[0035] A device identifier of the device is obtained by the bus driver (block 304). The device identifier 120, for instance, may describe hardware and/or firmware employed by the device, specific functions supported by the device, be configured as a unique identifier for a particular device (e.g., a serial number), and so on. The device ID is then obtained by a flag engine that is executed by the computing device (block 306) and a check is performed to determine whether a flag correspond to the device ID (block 308), further discussion of which may be found in relation to the following figure.

[0036] FIG. 4 depicts a procedure 400 in an example implementation in which a query is performed to determine whether a flag has been set for a particular device. Responsive to detection by a computing device that a device has been communicatively coupled to bus of

the computing device, a database is queried by the computing device to determine whether one or more flags have been set for the device (block 402). The flag engine 116, for instance, may receive the request from the driver 114 and query the database 208.

[0037] Responsive to a determination made as a result of the querying that the one or more flags have been set for the device identifier, communicate information associated with the one or more flags to software that is associated with the device to configure the software (block 404). As stated before, the information 214 may be configured in a variety of ways. For example, the information 214 may describes flags 118 that may be used for code path selection by the driver 114, may include a payload 216 associated with the flag that may be used as a basis for configuring device specific behaviors such as extended behaviors 220 that go beyond a standard, and so on.

[0038] FIG. 5 depicts a procedure 500 in an example implementation in which telemetry data generated as a result of configuration of a driver by one or more flags is leveraged. Telemetry data is received via a network at one or more computing devices of a network service, the telemetry data identifying firmware of devices, corresponding drivers via which the devices communicate, and configuration of the drivers (block 502). The flag engine 116, for instance, may store the telemetry data in the database 208 that describes flags 118, corresponding information 214, drivers 114, and devices 112 (e.g., device IDs 12) that were involved in configuring a driver 114 for compatibility with the device 112. This data may then be transmitted at predefined points in time, responsive to when the described operations were performed, and so on to a network service 210 via a network 212.

[0039] The telemetry data is processed at one or more computing device to determine which of the devices described in the telemetry data include firmware that involves an update to enable operation of functionality of the device in conjunction with a respective computing device that otherwise would not be enabled (block 504). Continuing with the previous example, a telemetry module 222 may be employed by the network service 210 to process the telemetry data, which may be used for a variety of purposes. For example, the telemetry data for the device 112 may provide details on which devices 112 (e.g., and corresponding firmware) continue to involve use of flags 118 to provide compatibility, which devices 112 utilize flags 118 to enable device specific capabilities (e.g., extended behaviors 220), which devices have upgraded firmware that no longer requires the use of flags 118, and so on. Developers may then use this information to determine compatibility test requirements when introducing updates to software (e.g., to a new version of an

operating system or service pack update) to ensure device compatibility does not regress for devices that depend on the flags 118 to enable device specific capabilities. A variety of other examples are also contemplated without departing from the spirit and scope thereof.

5 **Conclusion**

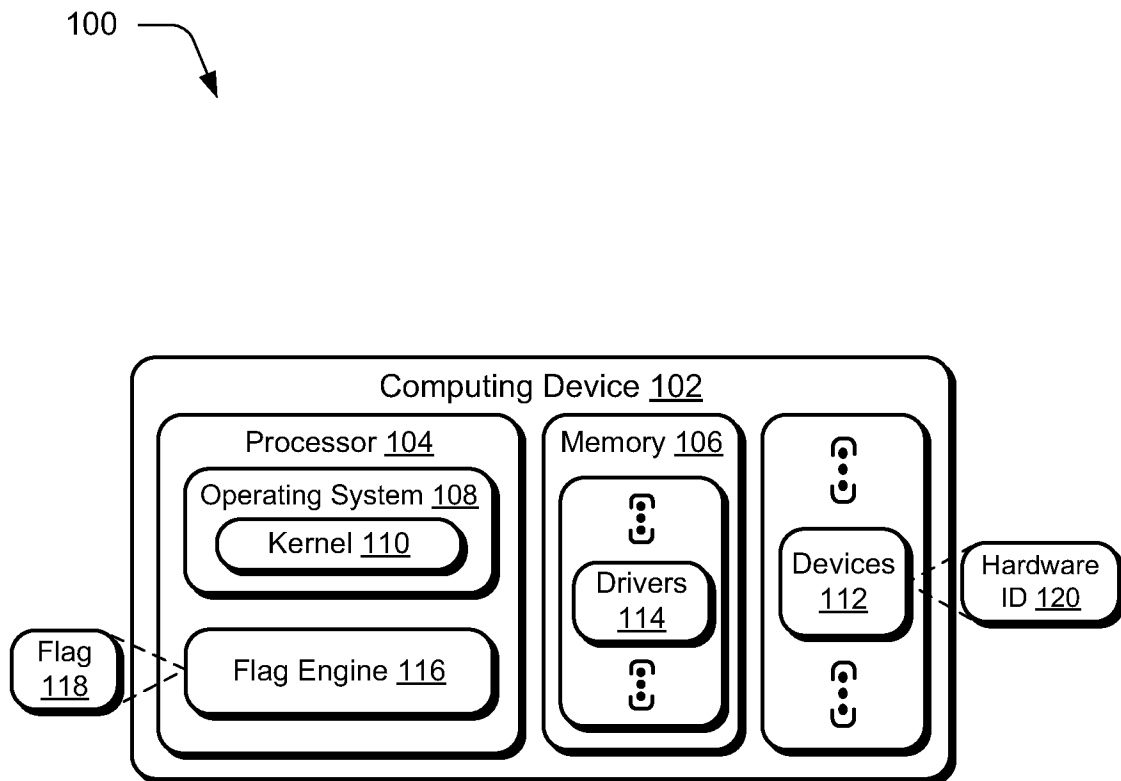
[0040] Although the invention has been described in language specific to structural features and/or methodological acts, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as example forms of implementing the
10 claimed invention.

CLAIMS

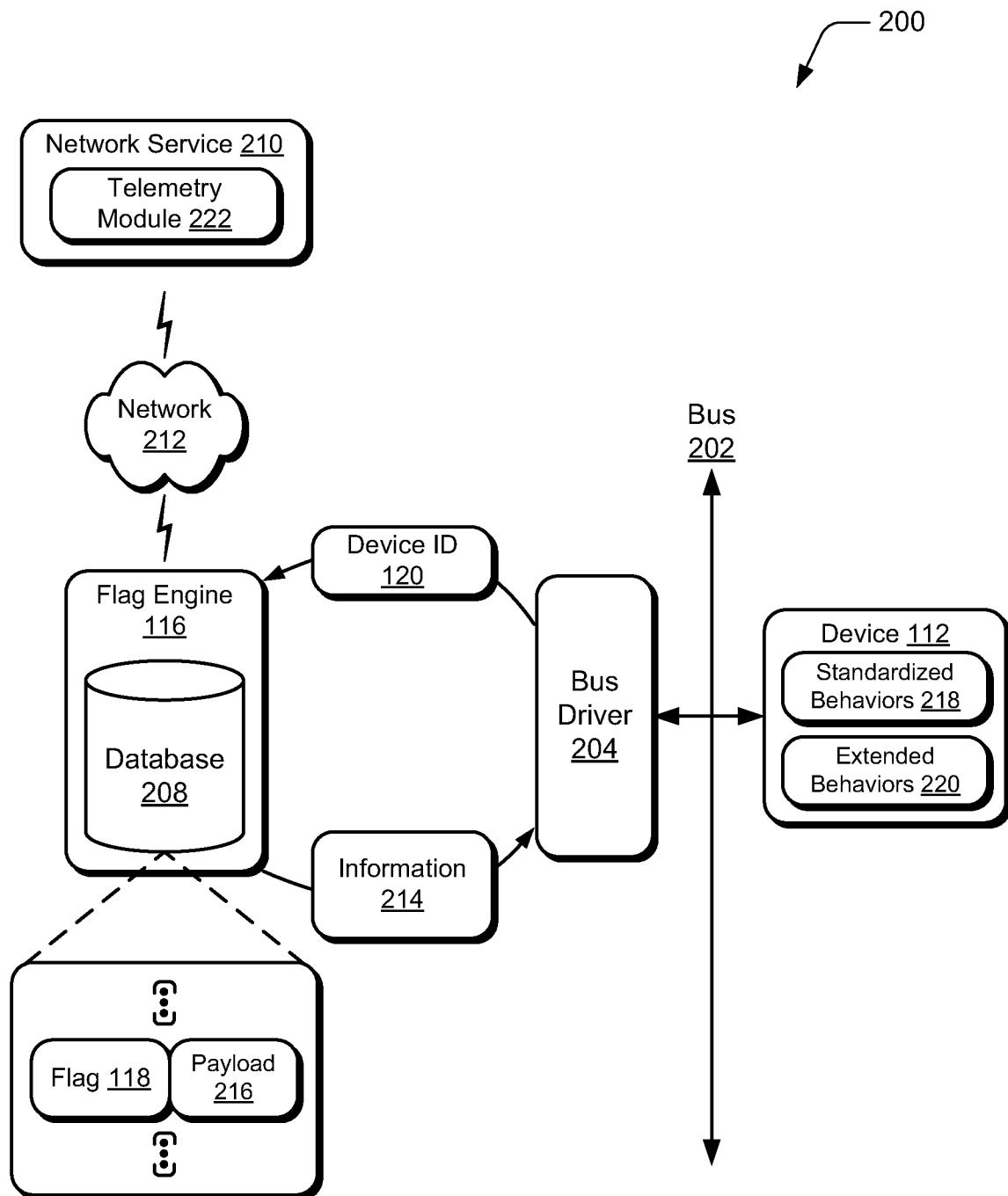
What is claimed is:

1. A method comprising:
responsive to detection by a computing device that a device has been
5 communicatively coupled to a bus of the computing device, querying a database by the
computing device to determine whether one or more flags have been set for the device;
and
responsive to a determination made as a result of the querying that the one or more
flags have been set for the device identifier, communicating information associated with
10 the one or more flags to software that is associated with the device to configure the driver.
2. A method as described in claim 1, wherein the information is configured
for use in code path selection of the software.
3. A method as described in claim 1, wherein the information is to configure
the software for device specific capabilities of the device.
- 15 4. A method as described in claim 1, wherein the software is an application.
5. A method as described in claim 1, wherein the software is a bus driver,
class driver, or filter driver.
6. A method as described in claim 1, wherein the software is configured to
handle capabilities of the device in accordance with a standard and the information is to
20 configure the driver for capabilities of the device that do not involve the standard.
7. A method as described in claim 1, wherein the software is configured to
handle capabilities of the device in accordance with a particular version of an operating
system and the information is to configure the software for a previous version of the
operating system.
- 25 8. A method as described in claim 1, further comprising communicating
telemetry data via a network to a network service that describes usage of the information
by the software.
9. A method as described in claim 8, wherein the telemetry data is usable to
determine compatibility test requirements for introducing an update.
- 30 10. A method as described in claim 8, wherein the telemetry data is usable to
identify particular said devices in a market that are to involve a firmware update to enable
functionality of the device that otherwise would not be enabled.

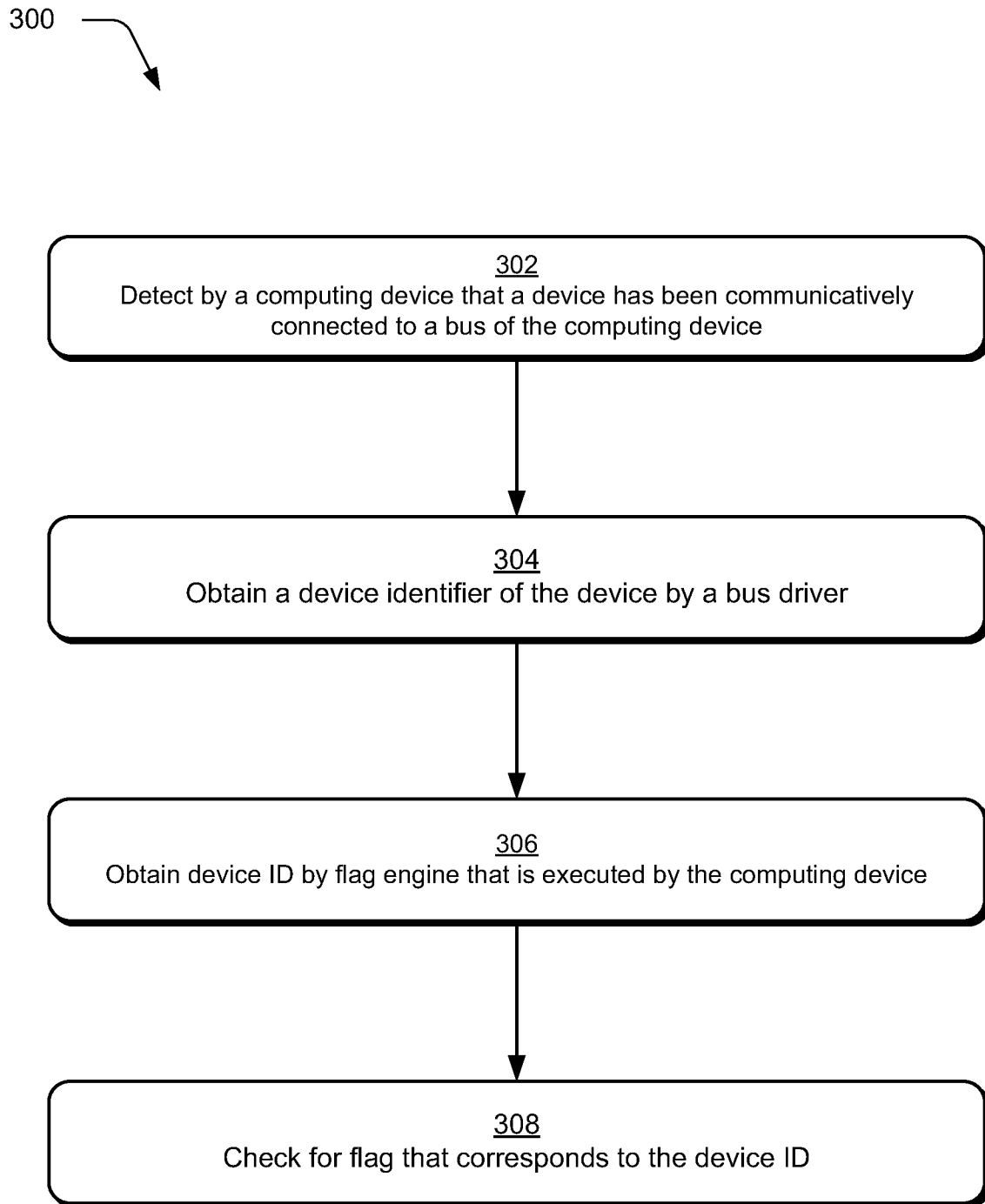
1/5

**Fig. 1**

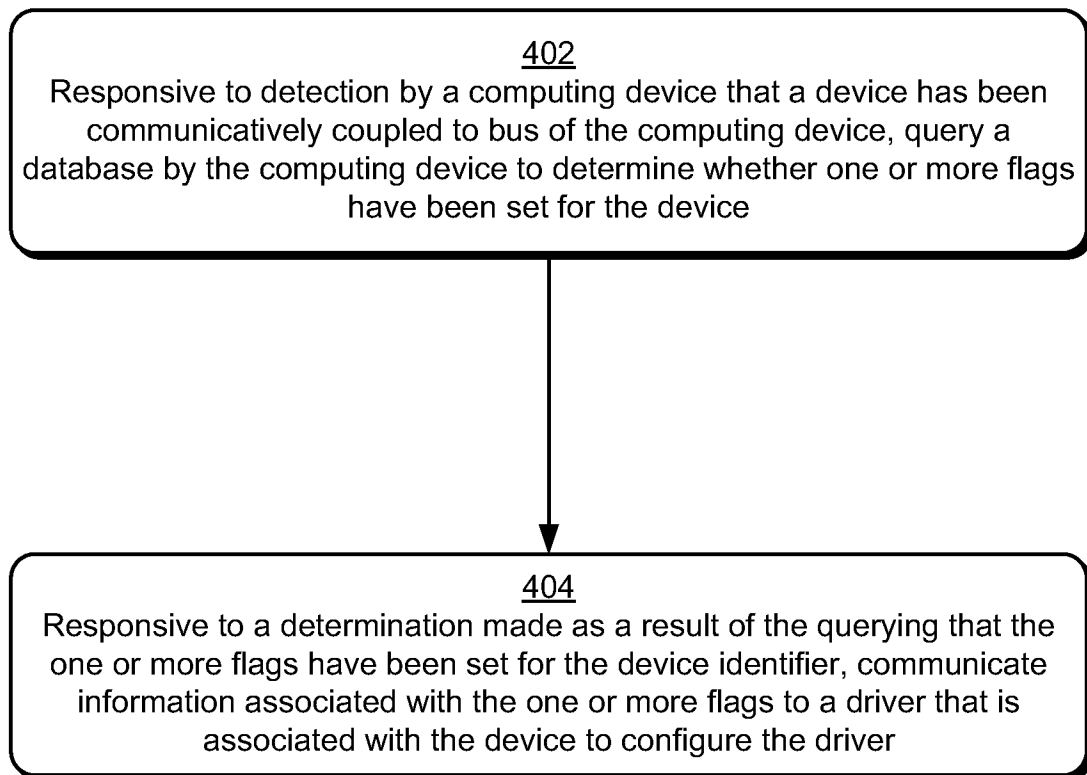
2/5

**Fig. 2**

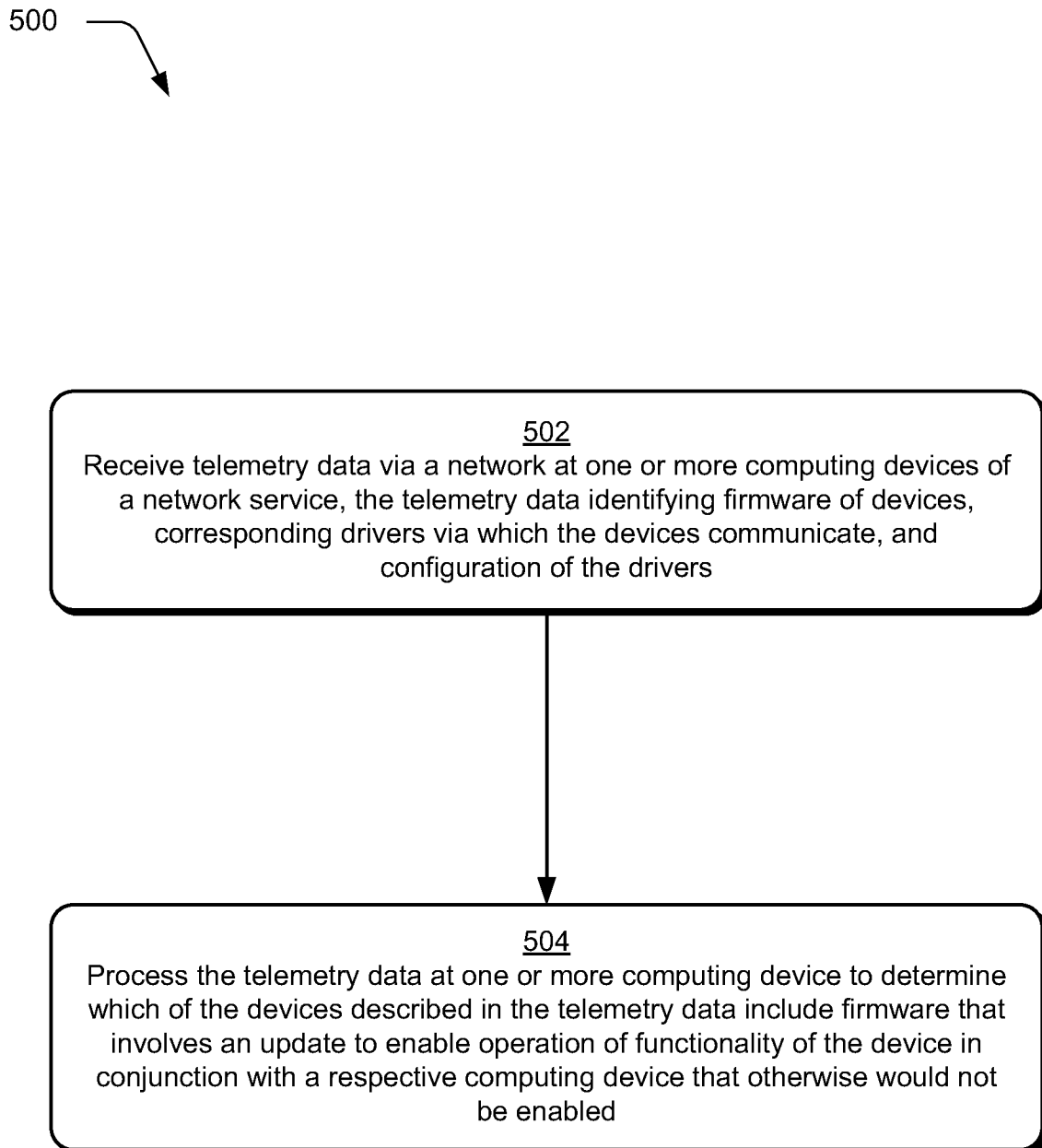
3/5

**Fig. 3**

4/5

400
**Fig. 4**

5/5

**Fig. 5**