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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 July 2017 (06.07.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/116803 A1

(51) International Patent Classification:
H04L 12/24 (2006.01) *H04L 29/08* (2006.01)
G06F 9/44 (2006.01)

(21) International Application Number:
PCT/US2016/067545

(22) International Filing Date:
19 December 2016 (19.12.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/983,244 29 December 2015 (29.12.2015) US

(71) Applicant: **ITRON, INC.** [US/US]; 2111 N. Molter Road, Liberty Lake, WA 99019 (US).

(72) Inventors: **GILLMORE, Matthew**; 2111 N. Molter Road, Liberty Lake, WA 99019 (US). **OTTING, James**; 2111 N. Molter Road, Liberty Lake, WA 99019 (US).

(74) Agent: **DESANTIS, Luanne, M.**; Itron, Inc., 2111 N. Molter Road, Liberty Lake, WA 99019 (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, DJ, 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, KH, KN, KP, KR, KW, 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).

[Continued on next page]

(54) Title: UNIVERSAL ABSTRACTION LAYER AND MANAGEMENT OF RESOURCE DEVICES OVER A NETWORK

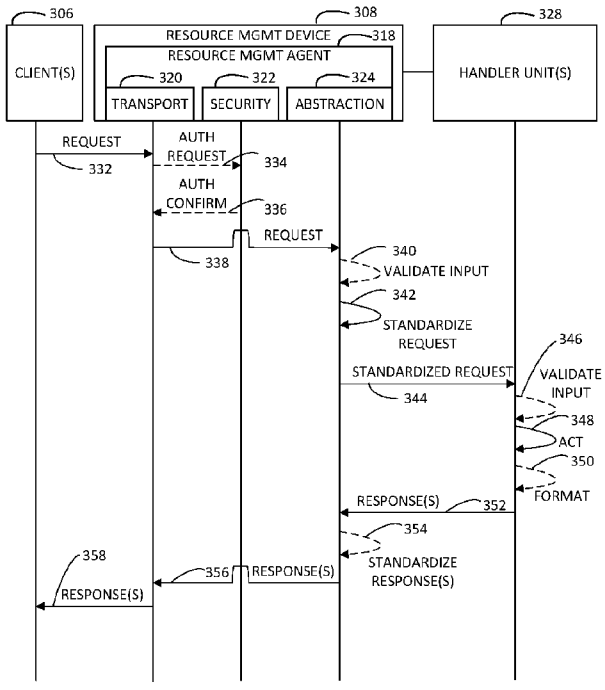


FIG. 3

(57) Abstract: An example method of managing a resource includes receiving a request associated with one or more of at least one resource device from one or more other devices over a network; standardizing the request using standardization definitions that are universal across different types of resource devices and their associated drivers; providing the standardized request to one or more handler units associated with the one or more resource devices; receiving a response regarding the request from the one or more handler units; preparing a packaged response; and sending the packaged response to the one or more other devices that sent the request. The method may further include determining whether input information accompanying the request is valid, and if the input information accompanying the request is invalid, sending an error message to the sender of the request. Devices that incorporate such methodology are also disclosed.

WO 2017/116803 A1

Published:

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

UNIVERSAL ABSTRACTION LAYER AND MANAGEMENT OF RESOURCE DEVICES OVER A NETWORK

RELATED APPLICATIONS

[0001] This PCT international application is a continuation of U.S. Pat. Appl. No. 14/983,244, filed December 29, 2015, entitled "Universal Abstraction Layer and Management of Resource Devices."

FIELD OF THE DISCLOSURE

[0002] This disclosure relates generally to networked or cloud-based resource management and communication systems.

BACKGROUND

[0003] As more and more "smart" devices are being developed, the need to interconnect them grows. The so-called "Internet of Things" (IoT) is an environment in which objects (even including, but not limited to, animals or people) are given unique identifiers and are able to transfer data over a network, with or without interaction. This concept has evolved from technologies such as wireless communications, micro-electromechanical systems (MEMS), and the Internet. In recent years, the IoT has exploded, with sensors and basic control systems constituting a large portion of the market. The IoT technical space has rapidly grown, with already hundreds of competing standards, technologies, and platforms. However, key considerations have not always been taken into account. For example, as IoT networks grow, so do security threats. In addition, with the rapid growth of the market, it is difficult to predict all future needs and requirements. Of the existing IoT platforms available currently, some may offer varying levels of security and extensibility, but many, if not all, of these require a high level of programming knowledge to build and compile software from scratch or require the use of limited software development kits (SDKs) that may not even be in a language familiar to many developers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] FIG. 1 is an illustration of an example IoT network environment in which example methods, apparatus, and articles of manufacture disclosed herein may be implemented, according to an embodiment of this disclosure.

[0005] FIG. 2 is a block diagram representative of example device 108A from FIG. 1, according to an embodiment of this disclosure.

[0006] FIGs. 3 and 4 are sequence diagrams showing examples of client request(s) involving resource device(s) via a resource management device, according to embodiments of this disclosure.

[0007] FIG. 5 is a flow diagram illustrating an example process flow of a client request and response from the perspective of a resource management device, according to embodiments of this disclosure.

[0008] FIG. 6 is a flow diagram illustrating an example of an error check of a request at an abstraction module of a resource management device, according to an embodiment of this disclosure.

[0009] FIG. 7 is a flow diagram illustrating an example method of polling a resource device, according to an embodiment of this disclosure.

[0010] FIG. 8 is a flow diagram illustrating an example method of retrieving data from a resource device from the perspective of a handler unit, according to embodiments of this disclosure.

[0011] FIG. 9 is a flow diagram illustrating an example method of receiving and acting on a request associated with a resource device from the perspective of a handler unit, according to an embodiment of this disclosure.

[0012] FIG. 10 is a flow diagram illustrating an example method of polling a resource device from the perspective of a handler unit, according to an embodiment of this disclosure.

[0013] FIG. 11 is a block diagram of an example authentication chip, according to an embodiment of this disclosure.

[0014] FIG. 12 is a sequence diagram showing an example authentication procedure, according to an embodiment of this disclosure.

[0015] FIG. 13 is a flow diagram illustrating an example authentication process from the perspective of one of two communicating devices, according to an embodiment of this disclosure.

[0016] FIG. 14 is a flow diagram illustrating an example method of challenge generation and response processing, according to an embodiment of this disclosure.

[0017] FIG. 15 is a flow diagram illustrating an example method of authentication from the perspective of an authentication chip, according to an embodiment of this disclosure.

[0018] FIG. 16 is a flow diagram illustrating an example method of challenge generation and response processing from the perspective of an authentication chip, according to an embodiment of this disclosure.

[0019] FIG. 17 is a block diagram of an example client or resource management device (i.e., endpoint device), according to an embodiment of this disclosure.

[0020] In the drawings, the leftmost digit(s) of a reference number may identify the drawing in which the reference number first appears.

DETAILED DESCRIPTION

[0021] While there are many IoT solutions currently being developed, there remains a need for a more unified, secure, and easily extensible platform. The following description discloses various technology aspects of an IoT platform that are directed to that end. For example, device communication is handled by agent software running on each resource management device, where data collection, command-and-control messages, registration, firmware updates, etc., may be managed in a unified system with end-to-end security. An abstraction layer may allow the agent software to be extended with new resource devices, from the simplest of sensors to complex control systems. Integrating new devices may be accomplished easily via basic configuration files or via scripts using any language and/or tools with which a developer is familiar. To further unify the platform, current or developing standards may also be incorporated.

[0022] A core of the platform includes agent software running on each resource management device, which is extended by a modular abstraction layer, as further described below. In general, networks of IoT devices may be managed by a web user interface (UI) hosted in the cloud. For the purposes of this disclosure, however, the nature of the management UI is irrelevant. The IoT communication architecture makes no distinction between an endpoint (e.g., a node, device, etc.) in the cloud, or any other device on the network.

[0023] As further described below, the agent software may serve several purposes, including, for example, data transport, security/encryption, device authentication, access

control list management and enforcement, interpretation and execution of abstraction layer configuration, firmware/software transfer/upgrades, service directory/indexing, basic device configuration and health management, etc.

[0024] In embodiments discussed in this disclosure, each device may be directly addressable from any other device on the network, and this may be done without network address translation (NAT) or proxy services. The embodiments disclosed herein may be implemented on any IP-based network. In embodiments, Internet Protocol Version 6 (IPv6) may be used, as it provides address space (both public and private) to allow globally unique addresses for the potential billions of IoT devices that may be deployed worldwide in the upcoming decade. In embodiments, IPv6 multicast groups may also be utilized. To reach networks that are not IPv6-ready, supplemental solutions (such as VPN tunneling) may be used. Embodiments disclosed herein are not limited to IPv6, however, as other routing protocol solutions may be used (e.g., IPv4, RPL (Routing Protocol for Low Power and Lossy Networks), etc.), as would be understood by one of ordinary skill in the relevant arts. For the purposes of this description, any routing protocol may be used that will provide a seamless network fabric that will allow data packets to be routed directly from any node to any other on a global scale (with the possible exception of nodes under firewalls or other measures designed intentionally to segregate networks for security).

Data Transport

[0025] In embodiments described herein, data transport may be accomplished using a data transport protocol, such as, for example, Constrained Application Protocol (CoAP). CoAP is a User Datagram Protocol (UDP)-based protocol that features a subset of Hypertext Transfer Protocol (HTTP) functionality and is designed for use in IoT and other constrained resource environments. CoAP uses Uniform Resource Identifiers (URIs) (paths) in a method similar to HTTP. URIs are convenient because they are human readable, instantly recognizable, hierarchical, and extensible. Layout and organization of resource devices is therefore easy to arrange into meaningful relationships, such that, for example, object properties or configuration parameters are exposed in ways that are easily understandable. If used, anything configurable, readable, or writable by the agent software may be part of the URI schema. Any resource devices (e.g., sensors, actuators, etc.), controls, configuration parameters, etc., may be exposed as resource URIs. CoAP also is useful as a resource discovery tool, as it can provide a richly

formatted listing of all resource devices managed by a resource management device, including metadata about the resource devices, such as resource type, human readable titles, and interface descriptions. Another useful feature of CoAP is its proposed "Observe" option extension, which extends the basic GET method in a way that can be thought of as "subscribing" to a URI. Using this feature, a client (such as a cloud data store, for example) that wants continual updates of device (e.g., sensor) data does not have to poll a device repeatedly. Instead, an initial request can be sent, indicating observation, and the device itself is responsible for sending further updates (e.g., whenever the resource device's state changes, etc.). By extending the URI schema with parameters for thresholds or minimal intervals, for example, minimizing network resource device usage can be accomplished. CoAP also defines how it interacts in multicast environments and has its own registered "All CoAP Nodes" variable scope multicast address. In networks with large numbers of homogeneous (or nearly homogeneous) devices, multicast can reduce network overhead considerably, as well as facilitate device discovery. This functionality may be useful for group management functions (e.g., to turn on all the lights in a particular group), firmware updates/transfers, device discovery, etc. While CoAP has many advantages, the data transport protocol used is not limited to CoAP, as other data transport protocols may be used (e.g., HTTP, HTTPS (HTTP Secure), MQTT (MQ Telemetry Transport), AMQP (Advanced Message Queueing Protocol), XMPP (Extensible Messaging and Presence Protocol), etc.), as would be understood by one of ordinary skill in the relevant arts.

Transport Security

[0026] Transport security is another important aspect of an IoT solution. An example of a transport security implementation that may be used in the disclosed IoT platform is Datagram Transport Layer Security (DTLS). DTLS is an extension of Transport Layer Security (TLS) for use with unreliable transports such as UDP. DTLS can do for UDP and CoAP what TLS (and previously, Secure Sockets Layer (SSL)) does for HTTP. While DTLS has many useful features, some desired features may need to be provided as modifications. For example, raw public keys may be incorporated. Raw keys are not signed, and therefore cannot be authenticated directly, so other authentication methods (such as described below) may be needed. Another example of a potential modification to DTLS is one that would allow its use with multicast. Multicast is a communication method in which a transmitted single data packet

can be replicated to multiple receivers. While DTLS is provided here as an example, the transport security method used is not limited to DTLS, as other transport security solutions may be used.

[0027] In many current security solutions, certificates are used to verify whether or not one communicating device can trust another communicating device. However, with the vast number of devices potentially communicating in an IoT environment, it is difficult to manage such a large number of certificates. In the description below, embodiments of a hardware cryptographic authentication solution are described that use a unique device serial number and an encrypted key that is written once and cannot be read by the host operating system. In one embodiment, this is accomplished with a hardware dongle, such that device identity can be guaranteed without the need to maintain a classic certificate authority structure, or embedding unique certificates or keys in firmware of each device at manufacturing time.

[0028] The device authentication discussed above (and described in detail below) should also integrate with the chosen transport security implementation (e.g., DTLS, etc.). In one example of how this may be accomplished, raw public and private key pairs may be generated on a device (and may be regenerated on demand) initially to establish a secure transport session. Once the session is secured, each device may verify the identity of each other via the hardware authentication solution described herein. If each device is verified by the other, then the communication session may be considered authenticated for the device serial numbers in use. In embodiments, this authentication may last for the life of the session. In an embodiment, the communicating devices may also optionally exchange sub-device IDs in addition to serial numbers. Sub-device IDs may indicate different applications or users sharing the same physical device, but which may need to have separate access controls in place for security and/or permission reasons. When multiple users and/or applications need to have simultaneous access on a device, a new transport session may be used for each device serial number and sub-device ID combination. This may be particularly important for privilege separation in a multi-user interface such as a central web management server.

Permissions

[0029] With the transport layer secured, and the identity (and optionally user/application) of the session endpoint verified, another aspect of security to take into account may include determining what permissions a given session should have on a per-resource level, for example.

One way of accomplishing this is through use of access control lists (ACLs). ACLs may be provided for any resource (e.g., via a URI, etc.) and method (GET, PUT, POST, DELETE, etc.) combination for any particular resource device. ACLs can be keyed to the device serial number (and optionally to the device serial number / sub-device ID combination). The transport security method in use for a device's communications may also be associated with a specific device serial number (and possibly sub-device ID), which will allow an ACL to be checked for every request. ACL inheritance (i.e., applying a rule to a resource and all of its children) and/or wildcards/matching rules may also be supported. ACLs may be managed remotely, by any other device/user that has the appropriate permissions to update the ACLs. In this way, there does not need to be a central authority that has control. Rather, any device/user that is allowed access to, for example, "PUT" or "POST" new ACLs to the ACL control resource may change the access rules on a device, depending on the set permissions.

[0030] A "cloud" server may be used as a web-based device management system. This system may allow configuration changes to be pushed to devices individually or en masse (see below). It may also monitor devices for basic health and network status, and may allow "browsing" devices and their resources. A cloud server is similar to any other node in the IoT network. It may use the same communications and security protocols, may provide the same type of resource scheme, and may have its own hardware cryptographic chip and device identity. One difference, however, is that the default ACL that is provided with device firmware may have all access granted to the cloud server device. That may allow further ACLs and other updates to be pushed to devices from the cloud. The default firmware on new devices may also designate the cloud server as a "check-in" server in order to register with the network.

[0031] One special application of ACLs will be for multicast use. ACL rules for a resource device may be created to allow multicast to that resource device, separate from other (e.g., unicast) rules. In the ACL storage itself, such rules may provide a multicast address/group that should be joined by the device to listen, the group key/cipher specification to be used, the address of the allowed/expected sender(s) using that key/cipher, etc. As an example, a group of individual IoT lights in an auditorium may all be administratively grouped so that they can be turned on/off simultaneously. Once the group is created, the controlling system (e.g., from in the cloud) may update the ACLs on the grouped devices, adding the new multicast ACL for the on/off and other lighting-related features of the devices, authorizing itself (and optionally a switch/control/input device) as a sender, and providing a key cipher. After this initial configuration, broadcast messages may be sent to the new multicast address by either a central

control or the switch device, and the messages may be decrypted by the individual devices using the key/cipher in the ACL. The messages may then be processed and the command (e.g., turning the lights on or off) may be executed.

Abstraction Layer

[0032] Resource management and the ability to easily allow new IoT resources to be mapped into the URI space provided by the agent software of a resource management device may be accomplished by an abstraction layer in the herein described IoT platform, according to embodiments. The abstraction layer can allow resource devices operating on a variety of different interfaces (e.g., 12C, GPIO, A2D, etc.) to be accessible as URI resources that may be accessed from transport protocols (e.g., CoAP, HTTP, etc.). The abstraction layer may accomplish this using a configuration file which may directly provide basic metadata for simple resource devices (e.g., basic sensors) that may already be supported by well-known drivers (e.g., Linux kernel drivers, etc.). However, for more complex systems (e.g., serial control of a power inverter, etc.), more complex URIs may be mapped, and processing may be extended using custom software programs. In an embodiment, the custom software programs may be written in any language. Abstraction configurations may be provided during firmware provisioning, or may be updated/changed/removed through data transport management URIs, for example. A benefit of this abstraction layer is the ability to easily change a resource devices abstraction properties with a configuration change without the need for re-compilation, etc.

[0033] In embodiments, abstraction layer configurations, and resource handler programs, format data in a standardized manner. In this way, similar types of resource devices (e.g., similar types of sensors or other data sources, etc.) can be read and their information parsed in a homogeneous fashion. For example, all readings from temperature sensors may be provided as a floating point value in degrees Celsius, with metadata describing range and accuracy. In this way, the underlying physical device and drivers become irrelevant to the rest of the IoT platform, and all similar temperature sensors may be treated identically by upstream services (e.g., data collection, analytics, and/or other services). Some configuration examples handled by the abstraction layer follow.

[0034] A basic configuration example may involve a digital temperature sensor. The digital temperature sensor may already have drivers provided by Linux, for example. The digital temperature sensor may provide a temperature in Celsius in a virtual file at a specific

location (e.g., /sys/bus/i2c/devices/i2c-1/0-50/temp1_input). This location may specify the location from which requested data may be retrieved by a handler unit, for example. The associated URI configuration may look like, for example:

```
URI:    /sensor/temp/1/celsius
            method:  GET
                    handler: "cat /sys/bus/i2c/devices/i2c-1/0-50/temp1_input"
                    handler_io: stdio_plain
                    content_type: text/plain
                    interface: sensor_ro
                    unit: temp/Celsius
                    min:  -30.00
                    max: 140.00
                    accuracy: 0.025
```

In this example, a GET to the URI may be handled by running the Linux "cat" program with the argument of the device file. The command may read the contents of that virtual file to standard output. With handler_io set to stdio_plain, the agent expects input/output through the standard input/output system, in plain (i.e., unstructured) format. That is, the output may be passed directly into the payload of a CoAP response, for example. The remainder of the configuration shown above may provide metadata (e.g., for the CoAP system) for both directory services and resource discovery (e.g., this is a sensor type interface, defined to be readable in units Celsius), for example, as well as content types for the actual payload. Other types of resource devices, such as a webcam, for example, may return a binary payload of a jpeg image or other arbitrary data, but may still be configured in this basic way.

[0035] Controls may also be utilized with the basic input/output handler model discussed above. Consider the following example involving a light:

```
URI:    /control/light/1/on
        method: PUT
                handler: "echo $payload > /sys/bus/i2c-1/0-55/relay1_output"
                handler_io: stdio_args
                content_type: text/plain
```

```
        interface: boolean
        payload:
                type: boolean
```

In this example, an additional feature of input sanitization is shown. A configuration file may specify expected input types and ranges, as well as more general regular expressions, by which input may be checked or sanitized prior to being passed as an argument to a program, for example. Here, by defining an expected payload as a boolean, the payload can be verified by agent software, and if it is anything other than a 0 or 1, an error can be returned to the client device that made the request. If the payload is valid, it can be passed as an argument to the control program. In this case, the control program is an "echo", meaning no code is needed. URI parameters (e.g., query strings) can be sanitized in a similar manner, as shown in the example below:

```
URI:    /control/light/1
        method: PUT
                handler: "/opt/bin/update-light.p1 $on  \
                        -o $overheat_temp  \
                        -n $name  \
                        -b $brightness"
                handler_io: stdio_args
                input_parameters:
                        on:
                                type: boolean
                        name:
                                type: string
                                max_length: 50
                        overheat_temp:
                                type: float
                                min: -25.5
                                max: 98.6
                        brightness:
                                type: string
```

regex: "(ON|OFF|HIGH|LOW)"

For this example, a valid request may then look like, for example: coap://node-17/control/light/1?on=1&name=Light1&overheat_temp=50.0&brightness=LOW. If desired, the raw payload may be passed to stdin without sanitization (e.g., using stdio_plain functionality), and input sanitization may be left to the handler program.

[0036] A more complex example may involve communicating over a serial interface with a solar power inverter, for example. Obtaining a reading from a solar power inverter may involve sending several commands, parsing responses, etc. More advanced tasks (e.g., taking a panel offline, reconfiguring other panels to balance the system, etc.) may be quite complex. Some serial interfaces may involve navigating a human-readable menu, for example, or searching through pages of output to obtain the requested value. These complex tasks may be abstracted into an external program. An example of this type of configuration may look like:

URI: /inverter/configure
method: POST
handler: "/opt/solar_control.py"
handler_io: stdio_yaml
interface: brandx_inverter

In this example, a structured data object is passed, with a result (e.g., payload) and other information that is available about the request. Standard input/output may still be used but it may be formatted as a standard data object. In this example, it is formatted as a YAML object. YAML is a human readable data format. While this example refers to a YAML data object, it should be understood by those of ordinary skill in the relevant art that this is not to be limiting, as other data formats may also be used (e.g., JavaScript Object Notation ((JSON), etc.). Further to this example, the payload of the request (e.g., a CoAP request) may be arbitrary, but may be wrapped in a standardized object (e.g., a YAML object) that also provides metadata about the request. This metadata may include, for example, identification of the client initiating the request, transport security metadata (e.g., cipher strength, lifetime, etc.), the URI and method, any URI-based parameters (e.g., query string), any other relevant information, etc. The external program is then free to process the payload (and metadata) as needed. Upon completion, the program may be expected to return its own structured data object (e.g., its own

YAML object) through standard output. That structured data object may then define the response, including, for example, the content type, headers, the payload, etc., and may indicate success, or any valid error codes and/or other responses.

[0037] From the above examples, standardized parameters used for data retrieval / configuration may include, but are not to be limited to, for example, uniform resource identifiers (URIs), method types, handler definitions, input/output definitions, content types, interface types, unit types, minimum indicators, maximum indicators, accuracy indicators, etc. Standardized parameters used for activation / configuration may include, but are not to be limited to, for example, uniform resource identifiers (URIs), method types, handler definitions, input/output definitions, content types, interface types, payload types, one or more input parameters, types of each input parameter, maximum indications for one or more input parameters, minimum indications for one or more input parameters, type options, etc. The standardized parameter definitions used by the abstraction layer may be provided to, changed, or removed from a resource management device by way of updates to the agent software of the resource management device. In an embodiment, the standardized definitions used by the abstraction layer may be provided to, changed, or removed from a resource management device via use of one or more URIs.

Resource Observation

[0038] As mentioned earlier, at least one data transport protocol (e.g., CoAP) has "Observe" functionality. In embodiments, the abstraction layer may handle observations in various ways. In one example, a handler program for the observe method can be directly specified, and can run continuously, emitting a new payload every time a data value is retrieved, every time a resource device's value changes, every time a new payload meets specified criteria or is in accordance with a specified observation condition (e.g., meets/exceeds a set threshold that may be, for example, specified in the request or via configuration), etc. In an embodiment, data retrieval and/or emission may occur according to a specified time interval. A new payload may be provided to the agent software, and may subsequently be provided to any clients that may have requested observation. In another example, observation may be simulated by the agent software itself. In this example, the agent may call the URIs GET method, for example, and store the payload. A minimum polling interval may be defined in a configuration file, and the agent may execute the GET handler at that interval (or more or less frequently, if requested

by a client, for example), comparing the newly returned payload to the cached value. If there is a change beyond a threshold (which may also be defined in the configuration file), this change may be transmitted in a message to the observing clients. For large payloads, a hash of the payload may be stored and compared, rather than the payload itself, to avoid memory exhaustion.

[0039] The abstraction layer and also the above-introduced hardware cryptographic authentication will be described in more detail below, with examples.

[0040] Embodiments are now described with reference to the figures, where like reference numbers may indicate identical or functionally similar elements. While specific configurations and arrangements are discussed, it should be understood that this is done for illustrative purposes only. A person skilled in the relevant art will recognize that other configurations and arrangements can be used without departing from the spirit and scope of the description. It will be apparent to a person skilled in the relevant art that this can also be employed in a variety of other systems and applications other than what is described herein.

[0041] FIG. 1 illustrates an example IoT network environment 100, according to an embodiment of this disclosure. In an example, and for ease of describing the embodiments herein, IoT network environment 100 may be an advanced meter infrastructure (AMI) network or other similar network. Environment 100 may include a central office 102, which may be associated with a data processing entity (e.g., a utility company in the case of an AMI network). The central office may communicate through one or more networks 104, which may be the Internet or other network having widespread or local functionality. Data collectors or client devices 106A-106M (collectively, clients 106) may be configured to communicate with a plurality of downstream devices. In the example shown, a plurality of network nodes, such as endpoints 108A-108N (collectively, endpoints 108) may be configured in a mesh network, star network or other configuration. One or more of the endpoints 108 may be configured for communication with one or more client devices 106 via network(s) 104. In an embodiment, an endpoint device 108 may also be configured as a data collector device 106. In an embodiment, endpoints 108 may access network(s) 104 via a routing or gateway device (not shown). While in this example, client devices 106 are shown as network nodes, in other embodiments, client devices 106 may be located at the central office and/or may be embodied in mobile data collection devices. In an example operation, a client device 106 may request data or send a command (e.g., a configuration command, an execution command, etc.) or other

communications to one or more endpoints 108. In an AMI network example, requested data may include consumption information (e.g., associated with an electric, gas or water meter), sensor information (e.g., temperature readings, flow rate information, leak detection information, etc.), etc. Additionally, one or more client devices 106 may send software updates, firmware updates, instructions or other information to one or more endpoints 108.

[0042] In an expanded view, a simplified block diagram 109 shows various main components of an example endpoint 108A. Endpoint 108A may include a communication system 110 and a processing unit 112. Communication system 110 may include communication interface(s) that may be implemented in hardware or a combination of hardware and software, and may provide wired or wireless network interface(s) to one or more networks, such as network(s) 104 of FIG. 1. Communication interface(s) may be a part of, or connected with, an interface circuit and/or may include or connect with communication devices such as a transmitter, a receiver, a transceiver, a modem and/or network interface card to facilitate exchange of data with another devices. Network connectivity (e.g., local area networking (LAN), wide area networking (WAN), etc.) may be provided by one or more of Fiber, Ethernet, cellular 3G, 4G, 4G LTE, WiMAX, satellite, etc. Processing unit 112 may include one or more processors 114 and a memory 116. All of these components of endpoint 108A contribute to allowing communications to and from endpoint 108A. This block diagram is also representative of similar components that may comprise an example client device 106 and similarly contribute to allowing communications to and from a client device 106. In fact, as stated above, in embodiments of this disclosure, which can be implemented in a cloud-based environment, an endpoint device 108 may be configured to act as a client device 106, and vice versa.

[0043] FIG. 2 illustrates a further expanded view of components of an example endpoint device 108A, according to an embodiment of this disclosure. As was shown in FIG. 1, endpoint device 108A may include communication system 110, processing unit 112, one or more processors 114, and memory 116. Memory 116 may include agent software module 218, which, when executed by processor(s) 114, may control communications with client devices 106 (FIG. 1), and manage its associated resource devices, using a transport module 220, a transport security module 222, an abstraction module 224, and optionally an observation module 226, as described in more detail below. Transport module 220 manages data transport using a data transport protocol (e.g., CoAP, or other data transport protocol, as discussed earlier in this document). Transport security module 222 manages communications security using a

transport security protocol (e.g., DTLS, or other transport security protocol, as discussed earlier in this document). Abstraction module 224 handles the management of the substance of the communications to and from resource devices associated with endpoint device 108A, which serves to standardize the information passed to and from the resource devices and in turn easily extends the capabilities to incorporate more, and new, resource devices. Abstraction module 224 will be described in more detail in the following paragraphs. Agent software module 218 may optionally include observation module 226 for observations of associated resource devices as described above and further described below. Processing unit 112 may optionally further include an authentication unit 227, for further authentication beyond what may be provided by transport security module 222. Authentication unit 227 will be further described below. For the handling of each associated resource device, processing unit 112 may include one or more handler units (e.g., handler unit 228A and 228B), each associated with one or more resource devices associated with endpoint device 108A (e.g., resource devices 230A and 230B). Resource devices may include, for example, sensors, actuators, or any other device capable of one or more controllable and/or data providing functions. Examples of sensors may include, but are not to be limited to, temperature sensors, meters (e.g., water, electric, gas, oil, etc.), leak detection sensors, motion sensors, light brightness sensors, air quality sensors, solar panel error detection sensors, etc. Examples of actuators may include, but are not to be limited to, utility valve controllers, street light controllers, traffic light controllers, solar panel controllers, heating/ventilation/air conditioning (HVAC) controllers, etc. Many other types of sensors and actuators may also be contemplated. Different types of resource devices (e.g., sensors and actuators) may also work in conjunction with one another. In FIG. 2, only two resource devices are shown as being associated with endpoint device 108A. This is by way of example and not limitation, as any number of resource devices may be associated with an endpoint device. Similarly, in FIG. 2, two handler units are shown by way of example and not limitation. Any number of handler units and any number of resource devices may be associated with an endpoint device. In addition, a single handler unit may manage more than once resource device. For purposes of the description herein, an endpoint device such as endpoint device 108A may be referred to herein as a resource management device that manages resources such as its (here shown) associated resource devices 230A and 230B.

[0044] FIGs. 3 and 4 are sequence diagrams showing examples of client requests involving (e.g., for data from, for a configuration change of, or for an action by) a resource device via a resource management device, according to embodiments of this disclosure. Referring to FIG.

3, a client device 306 (e.g., a client device 106 of FIG. 1) may send a request (332) to resource management device 308 (e.g., endpoint device 108A of FIGS. 1 and 2). At resource manage device 308, a transport module 320 of a resource management agent 318 receives the communicated request and sends an authentication request (334) to transport security module 322 of resource management agent 318 for authentication of the sending device. If authentication is successful, an authentication confirmation message (336) may be sent to transport module 320. Alternatively, if authentication is not successful, an error message may be returned to transport module 320 (which may in turn by sent to client 306), with no further request processing. Upon successful authentication of the sending device, the request is provided (338) from transport module 320 to abstraction module 324 of resource management agent 318. In an embodiment, abstraction module 324 may optionally validate the input parameters (340) provided in the request. If one or more input parameters are invalid or otherwise incorrect, an error message may be provided to transport module 320 (which may, in turn, be sent on to the requesting client device(s) (not shown)). Otherwise (or if validation of input is not employed), the request is standardized (342) by abstraction module 324, and the standardized request is provided (344) to one or more handler units 328 (e.g., handler unit 228A and/or 228B). If input validation was not performed by abstraction module 324, it may optionally be performed (346) by handler unit 328. If one or more input parameters are invalid or otherwise incorrect, an error message may be provided to resource management agent 318 (which may, in turn, be sent on to the requesting client device(s) (not shown)). Otherwise (or if validation of input is not employed), the handler unit 328 acts on the request (348) with the associated resource device (e.g., resource device 230A or 230B). In an embodiment handler unit 328 may optionally send an acknowledgement response to the resource management agent acknowledging receipt of the request. Handler unit 328 may optionally format any data (350) retrieved from its associated resource device as previously instructed (e.g., via configuration file), or as instructed in the data request. Handler unit 328 may then send one or more responses (352) to abstraction module 324. The response may include requested data, for example, or may include an error message if the request could not be retrieved (e.g., if input parameters were incorrect or invalid, if the resource device was down or malfunctioning, if requested data could not be formatted as required or requested, etc.). Abstraction module 324 may optionally standardize the received response (354) prior to providing the received response (356) to transport module 320, which in turn may provide the response (358) to the requesting client device(s) 306.

[0045] FIG. 4 illustrates a similar sequence diagram as shown in FIG. 3, except that, for observation of one or more resource devices, the resource management agent may include an observation module 460, or incorporate observation features into the abstraction module. When a response (e.g., requested data) is received (at 352) from a handler unit 328 that retrieved information from an associated resource device, the abstraction/observation module 460 may process and/or store the response (462) (e.g., in memory 116 (of FIG. 1 or 2) or other memory location) as an observed response for later comparison with subsequently received responses as part of the observation feature. In an embodiment involving observation, the standardized request provided to handler unit 328 (at 344) may instruct handler unit 328 to regularly poll an associated resource device for data. In an alternative embodiment involving observation, abstraction/observation module 460 may repeatedly send requests to handler unit 328 to retrieve the requested data. For resources being observed, the response(s) provided at 356 and/or 358 may include, for example, every data value retrieved, only data value changes, data values in accordance with a specified observation condition (e.g., meets or exceeds a predetermined threshold that may be, for example, specified in the request or via configuration), data values retrieved at a specified polling interval, etc.

[0046] FIG. 5 is a flow diagram illustrating an example process flow 500 of a client request and response from the perspective of a resource management device (e.g., an endpoint device 108/308), according to embodiments of this disclosure. At 502, a request associated with one or more of at least one resource device (e.g., resource devices 230A and/or 230B) managed by the resource management device may be received at a transport module (e.g., transport module 220/320) of agent software (e.g., agent software 218/318) of the resource management device from one or more other devices (e.g., client devices 106/306) over a network (e.g., network(s) 104). The request may include a request for one or more of: a data reading from one or more of the resource devices, a configuration change for one or more of the resource devices, or an action by one or more of the resource devices. An identity of each of the one or more requesting devices may optionally be authenticated at 504. Authentication may be performed by a transport security module (e.g., transport security module 222/322) of the agent software, and may optionally include authentication using the hardware cryptographic authentication described herein. At 506, the request may be provided to an abstraction module (e.g., abstraction module 224/324/460) of the agent software for standardization. The request may be standardized using standardization definitions that are universal across different types of resource devices and their associated devices, for example. At 508, the standardized request

may be provided to one or more handler units (e.g., handler units 228/328) associated with the one or more resource devices. At 510, a response may be received at the abstraction module from the one or more handler units. The response may be, for example, requested data reading(s) of the resource devices, a confirmation message (e.g., of a configuration change or successful action), or an error message (e.g., configuration change failure, unsuccessful action, unexpected or incorrect input parameters, etc.). At 512, the received response may optionally be standardized by the abstraction module. At 514, the response may be provided to the transport module to prepare the response for sending to the requesting devices at 516.

[0047] FIGs. 6-10 illustrate various examples of request events from the perspectives of various elements of a resource management device and associated resource devices, according to embodiments of the present disclosure.

[0048] FIG. 6 is a flow diagram illustrating an example of block 506 of FIG. 5, according to an embodiment. In this example, an error check of a request is performed at the abstraction module prior to standardization of the request. At 602, the request may be received at the abstraction module. At 604, the abstraction module may determine whether the inputs or parameters of the request are invalid (e.g., unexpected, incorrect, etc.). If so, at 606 an error message may be prepared/sent to the resource management agent (which in turn may be sent to requesting client device(s)). If not, at 608, the request is standardized, and at 610, the process 500 continues at block 508 (FIG. 5).

[0049] FIG. 7 is a flow diagram illustrating an example method 700 of polling a resource device, according to an embodiment of this disclosure. Method 700 is from the perspective of a resource management device that, for example, may have received a request from a client device to observe one or more resource devices. In an alternative embodiment, the request for observation may originate at the resource management agent (e.g., in order to process a client request or otherwise learn information regarding the resource device). At 702, one or more handler units may be directed to regularly poll a resource device or devices to be observed and to return payload information for the observed resource device(s). A polling frequency, or minimum polling interval, may have been included in the initial client request, may have been specified by the agent software of the resource management device, or may have been previously set for the handler unit(s) (e.g., in a configuration file). In an embodiment, the resource management agent may repeatedly send requests to the one or more handler units to retrieve payloads. At 704, payload(s) returned by the handler unit(s) may be received and stored. At 706, a newly returned payload may be compared with a previously stored payload.

At 708, the newly returned payload may be stored if in accordance with defined criteria. For example, every newly returned payload may be stored, or storing may occur only if the newly returned payload differs from a previous payload, or storing may occur based on other criteria. Returned payload data may be sent to observing clients. In embodiments, every received payload may be sent to observing clients, a change in payload may be sent to observing clients, a payload in accordance with a specified observation condition (e.g., meets or exceeds a predetermined threshold that may be, for example, specified in the request or via configuration) may be sent to observing clients, etc.

[0050] FIG. 8 is a flow diagram illustrating an example method 800 of retrieving data from a resource device from the perspective of a handler unit, according to embodiments of this disclosure. At 802, a standardized request may be received, from a resource management agent (e.g., agent software of a resource management device), requesting a data read from a resource device that is managed by the resource management device. At 804, the requested data may be retrieved from the resource device in accordance with the standardized request. At 806, the retrieved data may optionally be formatted in accordance with the standardized request or as the handler unit may itself be configured to format data (e.g., in accordance with a configuration file). At 808, the retrieved data may be provided to the resource management agent.

[0051] FIG. 9 is a flow diagram illustrating an example method 900 of receiving and acting on a request associated with a resource device from the perspective of a handler unit, according to an embodiment. At 902, a standardized request associated with a resource device may be received by the handler unit from a resource management agent. At 904, the handler unit may determine if one or more inputs or parameters of the request are invalid (e.g., unexpected, incorrect, etc.). If so, at 906 an error message may be prepared/sent to the resource management agent. If not, at 908, the handler unit may perform the action requested (e.g., retrieve data from the resource device, configure the resource device, activate the resource device, etc.) in accordance with the standardized request. At 910, a response may be sent to the resource management agent. The response may include, for example, the requested data (possibly formatted by the handler unit), a confirmation that a requested configuration or activation was executed successfully, or an error message that the request was not able to be performed.

[0052] FIG. 10 is a flow diagram illustrating an example method 1000 of polling a resource device from the perspective of a handler unit, according to an embodiment. At 1002, a request to continuously poll for data from an associated resource device may be received by a handler

unit from a resource management agent. At 1004, the requested data may be continuously retrieved from the resource device (e.g., at a regular interval specified in the request, or known by the handler unit (e.g., in a configuration file or setting of the handler unit)), and an initial retrieved payload may be provided to the resource management agent. At 1006, the handler unit may send to the resource management agent new payload. In embodiments, the handler unit may send a new payload in any of the following example situations: every time a data value is retrieved from the resource device, every time a data value changes from the previous value, when a retrieved data value is in accordance with a specified observation condition (e.g., meets or exceeds a predetermined threshold that may be, for example, specified in the request or via configuration), etc.

Hardware Cryptographic Authentication

[0053] As discussed above, an important aspect of an IoT platform is security. While a transport security layer, such as transport security module 222 of FIG. 2, may normally use certificates to verify whether a communicating device can trust another communicating device, the use of certificates may not be the best security solution for an IoT platform. The large number of devices potentially communicating in an IoT environment, and any constraints of the network, may make it difficult to manage such a large number of certificates. Referring back to FIG. 2, authentication unit 227 may work in conjunction with transport security module 222 to provide a more manageable solution in the form of hardware cryptographic authentication, according to embodiments described herein. To implement this solution, authentication unit 227 may comprise an authentication chip that stores a write-once, never-read shared authentication key, a unique serial number, and various authentication code functions (e.g., Message Authentication Code (MAC) functions and/or hash-based MAC functions (HMAC)). The unique serial number is unique to the authentication chip, and is used to provide a unique identification of the device when connected to the authentication chip. The unique serial number may be readable from the chip, but not changeable on the chip. For these reasons, the unique serial number may be referred to herein as the serial number of the device to which it is connected. If the authentication chip is moved from one device to another, the unique serial number will no longer be associated with the device it was moved from, and will then be associated with the device it was moved to. The authentication codes may be generated by the chip itself, and may include, or be based upon, the serial number and the authentication

key. In addition, the chip may optionally generate one or more challenges to further strengthen security with a device with which the device associated with the chip is in communication. A challenge may be used to verify possession of the same or associated authentication key, for example. Ideally, a challenge may be a randomized string that is never to be reused, and cannot be guessed in advance. The authentication chip may be housed inside a device (e.g., a resource management device), or may be connected to a device using, for example, an external hardware dongle. Any devices that have the same key or an associated key (e.g., a key that belongs to a known group of admissible keys) burned in the authentication chip at manufacturing, for example, may be able to verify the identity of each other (e.g., by recognizing that the other device has an admissible key, by verifying a response to a provided challenge, etc.), and this is done in a manner that cannot be observed or copied by another entity. With this security mechanism in place, device identity may be safely confirmed without needing to maintain a classic certificate authority structure, and if using a hardware dongle, without embedding unique certificates or keys in the firmware of each device at device manufacturing time. Another advantage of using a hardware dongle is, if the key is somehow exposed or compromised, devices may be "rekeyed" by changing their dongles.

[0054] In an embodiment, the authentication chips may be "burned" with the key information by the manufacturers of the resource management devices that are to use the chips prior to shipping the resource management devices to customers for deployment. In an alternative embodiment, the chips may be shipped "blank" (i.e., with no key information) to the customers in case the customers want to use their own authentication keys and manage their own networks. This, however, may provide a segregated network, as devices with different key information will not be able to verify each other's identity. This may be advantageous in some cases, however, for customers who wish to host their own control centers.

[0055] FIG. 11 is a block diagram of an example authentication chip 1127, according to an embodiment. Authentication chip 1127 may be, or may be a part of, authentication unit 227 of FIG. 2, for example, which may be housed within a resource management device, or may be housed in a hardware dongle externally connected to a resource management device via a port or connection point of the resource management device, for example. Authentication chip 1127 may include one or more processor(s) 1178 and a memory 1180. Processor(s) 1178 may be implemented by, for example but not limitation, one or more integrated circuits, logic circuits, microprocessors, controllers, etc. Processor(s) 1178 may include a local memory (e.g., a cache) (not shown). Memory 1180 may be a volatile and/or a non-volatile memory. Volatile

memory may be implemented by, for example but not limitation, Synchronous Dynamic Random Access Memory (SDRAM), Dynamic Random Access Memory (DRAM), RAMBUS Dynamic Random Access Memory (RDRAM) and/or any other type of random access memory device. Non-volatile memory may be implemented by flash memory and/or any other desired type of memory device. Access to memory 1180 may be controlled by a memory controller (not shown). Memory 1180 may store, for example, a serial number 1182, authentication key 1184, and authentication codes 1186. Memory 1180 may also store authentication logic 1188. The data stored in memory 1180 may be used by processor(s) 1178 (e.g., by executing authentication logic 1188) to facilitate the authentication functions described herein.

[0056] As discussed earlier in this document, the hardware authentication solution may integrate with the transport security layer (e.g., with transport security module 222 of FIG. 2). For example, raw public/private keys may be generated on a device (and may be regenerated on demand) and may be used to initially establish a transport security session (e.g., a DTLS session). Once the session is secured, each device may verify the identity of each other via the hardware authentication solution described herein. If each device is verified by the other, then the communication session may be considered authenticated for the device serial numbers in use. As discussed above, in an embodiment, the communicating devices may also optionally exchange sub-device IDs in addition to serial numbers. Sub-device IDs may indicate different applications or users sharing the same physical device, but which may need to have separate access controls in place for security and/or permission reasons. When multiple users and/or applications need to have simultaneous access on a device, a new transport session may be used for each device serial number and sub-device ID combination.

[0057] FIG. 12 is a sequence diagram showing an example authentication process 1200, according to an embodiment. FIG. 12 illustrates authentication between two communicating devices. Device 1208 (Device 1) may be, for example, a resource management device (such as a resource management device 108) that manages a resource device (such as resource 230A or resource 230B of FIG. 2, for example). Device 1206 (Device 2) may be, for example, a client device (such as a client device 106) that may send a request to a resource management device to request data or request configuration or activation of an associated resource device. Devices 1206 and 1208 may each include one or more main processors as well as one or more authentication processors. The authentication processor(s) may be part of an authentication chip either internal to devices 1206 and 1208, or external to devices 1206 and 1208 (such as via a hardware dongle). In process 1200, an initial secure connection is established, at 1201,

between devices 1206 and 1208 via their main processor(s). The initial secure connection may be initiated by the requesting device (e.g., device 1206) and may be, for example, a DTLS connection. At 1203, a main processor of device 1206 may send a request for identity authentication to device 1208. The identity authentication request may be encrypted.

[0058] At 1205, a main processor of device 1208 may pass the identity authentication request to an authentication processor of an authentication chip of device 1208. At 1207, the authentication processor may use authentication functions stored on the authentication chip to generate one or more authentication codes using a unique serial number and key, each stored on the authentication chip. In addition, or optionally, one or more challenge(s) may be generated to provide to device 1206 as a request for identity authentication of device 1206. At 1209, the generated authentication code(s) (and challenge(s)) may be provided to the main processor of device 1208, and at 1211, a response to the identity authentication request may be returned to device 1206 with the generated authentication code(s) (and challenge(s)).

[0059] At 1211, device 1206 may receive the response/authentication code(s)/challenge(s), and at 1213, a main processor of device 1206 may pass the authentication code(s)/challenge(s) to an authentication processor of an authentication chip of device 1206. At 1215, the authentication processor may process the authentication code(s) received from device 1208 (e.g., based on a key of the authentication chip of device 1206) to determine if the code(s) are in accordance with its key. If not, authentication of device 1208 fails, and an authentication failure message may be provided to a main processor of device 1206 and optionally to device 1208. If so, the authentication is considered successful, and at 1217, the authentication processor of device 1206 may generate its own authentication codes (e.g., based on its own key) in response to the challenge received from device 1208. The generated authentication codes may be provided to a main processor of device 1206 at 1219. At 1221, a response to the received challenge, including the generated authentication codes of device 1206, may be sent to device 1208.

[0060] The response with authentication code(s) from device 1206 may be received by a main processor of device 1208 and passed on to an authentication processor of device 1208 at 1223. At 1225, the authentication processor of device 1208 may process the authentication code(s) received from device 1206 (e.g., based on the key of the authentication chip of device 1208) to determine if the code(s) are in accordance with its key. The authentication results may be provided to a main processor of device 1208 at 1227. If the authentication of device 1206 failed, an authentication failure message may be provided to device 1206. If

authentication was successful, a secure connection may be considered verified between devices 1206 and 1208 (at 1229).

[0061] FIG. 13 is a flow diagram illustrating an example authentication process 1300 between two devices, according to an embodiment, and from the perspective of a first device. At 1302, an initial secure connection may be established between the first device and a second device over a network using a communications protocol. At 1304, an identity authentication request may be received at the first device, from the second device. At 1306, the identity authentication request may be passed to an authentication processor of an authentication chip of the first device. At 1308, one or more authentication codes may be generated by the authentication processor using one or more authentication code functions stored on the authentication chip. The one or more authentication codes may be generated using a unique serial number of the first device and an authentication key, both of which may be stored on the authentication chip. At 1310, the authentication processor may provide the one or more authentication codes to the main processor to be sent to the second device as a response.

[0062] Included in the authentication code(s), or in addition to the authentication code(s), the authentication processor may generate one or more challenge(s) to be sent to the second device to request authentication of the second device. For example, the one or more challenge(s) may be sent to the second device along with the response. FIG. 14 illustrates an example method 1400 of challenge generation and response processing, according to an embodiment of this disclosure, which can be considered an extension of authentication process 1300. At 1402, the authentication processor may generate one or more challenge(s) to request authentication of the identity of the second device. At 1404, the generated challenge(s) may be provided to a main processor to be provided to the second device. At 1406, the main processor may receive a challenge response from the second device, which may be provided to the authentication processor at 1408. At 1410, the authentication processor may process the challenge response using one or more authentication code functions and the authentication key. The challenge response may include one or more authentication code(s) generated by the second device, and the processing of the challenge response may include processing the one or more authentication code(s) of the second device. For successful authentication, the authentication code(s) provided by the second device are to be in accordance with the authentication key of the authentication chip of the first device. In an embodiment, for successful authentication, the authentication key used by the second device is to identically match the authentication key stored on the authentication chip of the first device. In another

embodiment, the authentication key provided by the second device is to be in a predetermined range of, or a member of, a group of admissible keys. The results of the authentication verification may be provided to the main processor to indicate successful or unsuccessful authentication.

[0063] FIG. 15 is a flow diagram illustrating an example method 1500 of authentication from the perspective of an authentication chip, according to an embodiment. At 1502, an authentication processor of an authentication chip of a first device may receive an identity authentication request from a main processor of the first device that was received from a second device in communication with the first device over a network. At 1504, one or more authentication code functions (stored on the authentication chip) may be executed to generate one or more authentication codes in response to the identity authentication request. The generated authentication code(s) may be based on a unique serial number and authentication key of the first device. At 1506, the one or more generated authentication code(s) may be provided to the main processor to be provided to the second device.

[0064] Included in the authentication code(s), or in addition to the authentication code(s), the authentication processor may generate one or more challenge(s) to be sent to the second device to request authentication of the second device. FIG. 16 illustrates an example method 1600 of challenge generation and response processing, according to an embodiment of this disclosure, which can be considered an extension of authentication method 1500. At 1602, the authentication processor may generate one or more challenge(s) to request authentication of the identity of the second device. At 1604, the generated challenge(s) may be provided to a main processor to be provided to the second device. At 1606, a challenge response may be received from the main processor that was received from the second device. At 1608, the authentication processor may process the challenge response using one or more authentication code functions and the authentication key. The challenge response may include one or more authentication code(s) generated by the second device, and the processing of the challenge response may include processing the one or more authentication code(s) of the second device. For successful authentication, the authentication code(s) provided by the second device are to be in accordance with the authentication key of the authentication chip of the first device. In an embodiment, for successful authentication, the authentication key used by the second device is to identically match the authentication key stored on the authentication chip of the first device. In another embodiment, the authentication key provided by the second device is to be in a predetermined range of, or a member of, a group of admissible keys. The results of the

authentication verification may be provided to the main processor (at 1610) to indicate successful or unsuccessful authentication.

[0065] One or more features disclosed herein may be implemented in hardware, software, firmware, and/or combinations thereof, including discrete and integrated circuit logic, application specific integrated circuit (ASIC) logic, and microcontrollers, and may be implemented as part of a domain-specific integrated circuit package, or a combination of integrated circuit packages. The terms software and firmware, as used herein, refer to a computer program product including at least one computer readable medium having computer program logic, such as computer-executable instructions, stored therein to cause a computer system to perform one or more features and/or combinations of features disclosed herein. The computer readable medium may be transitory or non-transitory. An example of a transitory computer readable medium may be a digital signal transmitted over a radio frequency or over an electrical conductor, through a local or wide area network, or through a network such as the Internet. An example of a non-transitory computer readable medium may be a compact disk, a flash memory, SRAM, DRAM, a hard drive, a solid state drive, or other data storage device.

[0066] A processing platform of a client device (e.g., client device(s) 106, computing devices located at central office 102, mobile collection devices (not shown)) or an endpoint device (e.g., endpoint device(s) 108, of FIG. 1) may be embodied in any type of mobile or non-mobile computing device. Examples of mobile devices may include, but are not to be limited to, laptop computers, ultra-laptop computers, tablets, touch pads, portable computers, handheld computers, palmtop computers, personal digital assistants (PDAs), e-readers, cellular telephones, combination cellular telephone/PDAs, mobile smart devices (e.g., smart phones, smart tablets, etc.), mobile internet devices (MIDs), mobile messaging devices, mobile data communication devices, mobile media playing devices, cameras, mobile gaming consoles, wearable devices, mobile industrial field devices, etc. Examples of non-mobile devices may include, but are not to be limited to, servers, personal computers (PCs), Internet appliances, televisions, smart televisions, data communication devices, media playing devices, gaming consoles, industrial field devices, etc.

[0067] FIG. 17 is a block diagram of an example mobile or non-mobile client device or endpoint device, according to embodiments. Processing platform 1700 may include one or more processors 1741, memory 1743, one or more secondary storage devices 1745, one or more input/output devices 1747, and/or one or more communication interfaces 1749, in

communication via a bus, line, or similar implementation (not shown). Processor(s) 1741 may be implemented by, for example but not limitation, one or more integrated circuits, logic circuits, microprocessors, controllers, etc. Processor(s) 1741 may include a local memory 1751 (e.g., a cache). Memory 1743 may include a volatile and/or a non-volatile memory. Volatile memory may be implemented by, for example but not limitation, Synchronous Dynamic Random Access Memory (SDRAM), Dynamic Random Access Memory (DRAM), RAMBUS Dynamic Random Access Memory (RDRAM) and/or any other type of random access memory device. Non-volatile memory may be implemented by flash memory and/or any other desired type of memory device. Access to memory 1743 may be controlled by a memory controller (not shown). Data stored in local memory 1751 and/or memory 1743 may be used by processor(s) 1741 to facilitate the controlling of a resource device, according to embodiments of this disclosure.

[0068] Input/output devices 1747 may allow a user to interface with processor(s) 1741. Input devices may allow a user to enter data and/or commands for processor(s) 1741. Input devices may include, for example, an audio sensor, a microphone, a camera (e.g., still, video, etc.), a keyboard, a button, a mouse, a touchscreen, a track-pad, a trackball, isopoint, a voice recognition system, etc. Output devices may provide or present information to a user. Output devices may include, for example, display devices (e.g., a light emitting diode (LED), an organic light emitting diode (OLED), a liquid crystal display, a cathode ray tube display (CRT), a touchscreen, a tactile output device, a printer, speakers, etc.). The input/output devices 1747 may be connected to processor(s) 1741, for example, with an interface circuit (not shown). The interface circuit may be implemented by any type of interface standard, such as, for example, an Ethernet interface, a universal serial bus (USB), a PCI express interface, etc. For use with an output device, the interface circuit may include a graphics driver card, chip, and/or processor.

[0069] Communication interface(s) 1749 may be implemented in hardware or a combination of hardware and software, and may provide wired or wireless network interface(s) to one or more networks, such as network(s) 104 of FIG. 1. Communication interface(s) 1749 may be a part of, or connected with, the interface circuit discussed above, and/or may include or connect with communication devices such as a transmitter, a receiver, a transceiver, a modem and/or network interface card to facilitate exchange of data with external devices (e.g., computing devices of any kind) via a network, such as network(s) 104.

[0070] Secondary storage device(s) 1745 may store processing logic 1753 (e.g., software) to be executed by processor(s) 1741 and/or data 1755. Processing logic 1753 and data 1755 may be used by processor(s) 1751 to facilitate communications between devices and/or managing a resource device, according to embodiments of this disclosure. Processing logic 1753 may include instructions for executing the methodology described herein, for example. Examples of secondary storage device(s) 1745 may include one or more hard drive disks, compact disk (CD) drives, digital versatile disk (DVD) drives, Blu-ray disk drives, redundant array of independent disks (RAID) systems, floppy disk drives, flash drives, etc. Data and/or processing logic may be stored on a removable tangible computer readable storage medium (e.g., a floppy disk, a CD, a DVD, a Blu-ray disk, etc.) using one or more of the secondary storage device(s) 1745.

[0071] The technology disclosed herein provides a secure universal way to manage resource devices (sensors, actuators, etc.) in an IoT communications environment. A secure device authentication method allows two devices to authenticate each other's identity in a private manner undetectable and uncopiable by others. Communicating devices may generate their own public/private key pairs (regenerating as needed for security reasons, etc.). A new request may start with a transport security (e.g., DTLS) handshake, using these keys. Once the handshake is complete, another authentication method may be used to verify each other's identity (e.g., serial number and optional sub-device ID) via a hardware cryptographic chip with authentication code functionality (e.g., using challenges/responses). Once this second authentication is complete, the data transport session is associated with those specific devices (and optionally sub-devices), and may be trusted for the lifetime of the session, for example. If a device has multiple applications or users which may require different access levels, then a sub-device ID (e.g., unique to the user/application) may be provided during authentication, and separate data transport sessions may be initiated/maintained for each device/sub-device combination.

[0072] An IoT abstraction layer allows resource devices operating on a variety of different interfaces (e.g., I2C, GPIO, A2D, etc.) to be accessible as URI resources that can be accessed from various transport protocols (e.g., CoAP, HTTP, etc.) This abstraction layer allows, via configuration, the ability for resource devices to be mapped to URI resources in a flexible and extensible manner, with no need for re-compilation or constant software updates. At its most basic, a configuration may be used for each module that desires to add functionality. These

modules may support as little as a single sensor, or may support highly complex systems. The module configuration may define what resources are being offered by a new module (e.g., resource types, content type, etc.) and what methods are available for each resource. It may define a program/command to run for given requests. Multiple (and differing) resources may share the same programs/commands.

[0073] The particular examples and scenarios used in this document are for ease of understanding and are not to be limiting. Though described for use with data collection and managing of sensors in a utility metering environment, features described herein may be used in many other contexts and situations that may or may not involve data collection or utility metering. For example, the features discussed herein may be beneficial in many other contexts in which sensors, actuators, etc., are used, and may even extend to general communications (e.g., telecommunications networks, mobile device networks, cellular networks, satellite networks, computer networks, broadcasting networks, etc.). Some examples of other contexts include pipeline systems and related safety (e.g., methane detection, leak detection, metering (flow rate, etc.), etc.), street lighting systems (e.g., on/off of lights or groups of lights, dimming, automatic controls (e.g., turning on/off when brightness (or dimness) reaches a threshold, turning on when a vehicle is present, etc.), air quality systems, solar power systems (metering, panel adjustment, problem detection, etc.), heating/ventilating/air conditioning (HVAC) control systems (automatic pre-cooling/heating, sub-metering, circuit level controls/sensing, etc.), etc. Many other industrial, field, business, municipal, home, and personal systems may also be contemplated that would benefit from what is described herein.

[0074] Although certain example methods, apparatus and articles of manufacture have been disclosed herein, the scope of coverage of this patent is not limited thereto. On the contrary, this patent covers all methods, apparatus and articles of manufacture fairly falling within the scope of the claims of this patent.

What Is Claimed Is:

1. A method of managing a resource (230), the method comprising:

receiving, by a transport module (220, 320) of a resource management agent (218, 318) executed by a processor (114) of a resource management device (108, 308), a request (332) associated with one or more of at least one resource device (230) communicably coupled with the resource management device, from one or more other devices (106, 306) over a network (104);

providing the request (338) to an abstraction module (224, 324, 460) for standardization and standardizing the request (342) using standardization definitions that are universal across different types of resource devices and their associated drivers;

providing the standardized request (344) to one or more handler units (228, 328) associated with the one or more resource devices;

receiving a response (352) regarding the request at the abstraction module from the one or more handler units;

providing the response to the transport module (356) for package preparation and preparing a packaged response; and

sending the packaged response (358) to the one or more other devices that sent the request.

2. The method of claim 1, further comprising:

authenticating (334, 336) an identity of each of the one or more other devices (106, 306) that sent the request.

3. The method of claim 1, wherein the providing the request to the abstraction module (224, 324, 460) includes:

determining, by the abstraction module, whether input information accompanying the request is valid (340); and

if the input information accompanying the request is invalid, sending an error message to the one or more other devices (106, 306) that sent the request.

4. The method of claim 1, wherein the request is a request for data, and the response includes at least one of:

an acknowledgement that the request was received;

requested data in accordance with standardization parameters of the standardized request; or

an error message regarding the request.

5. The method of claim 1, further comprising:

standardizing (354) the received response by the abstraction module (224, 324, 460).

6. The method of claim 1, wherein the request is a request for a configuration change of one or more of the resource devices (230), and the response is one of:

a confirmation that the requested configuration change was successful; and

an error message regarding the configuration change request.

7. The method of claim 6, wherein the response is an error message and is at least one of:

an indication that the requested configuration change was unsuccessful; or

an indication that input information that accompanied the configuration change request was invalid.

8. The method of claim 1, wherein the request is a request for activation of one or more of the resource devices (230), and the response is one of:

a confirmation that the requested activation was successful; and

an error message regarding the activation request.

9. The method of claim 8, wherein the response is an error message and is at least one of:

an indication that the requested activation was unsuccessful; or

an indication that input information that accompanied the activation request was invalid.

10. The method of claim 1, wherein the request is a request to observe data from one or more resource devices (230).

11. The method of claim 10, wherein the standardized request includes instructions to continuously poll for data from one or more resource devices (230), and wherein the method further comprises:

receiving a new response at at least one of:

whenever data is retrieved from an observed resource device;

whenever a data value of an observed resource device changes;

whenever a data value of an observed resource device is in accordance with a specified observation condition; or

a defined polling interval.

12. The method of claim 1, further comprising:

directing one or more of the one or more handler units (228, 328) to poll one or more resource devices (230) to be observed and to return a payload for each observed resource device; and

storing the returned payloads (462).

13. The method of claim 12, wherein the directing includes sending an observation request to the one or more handler units (228, 328) to poll the observed resource devices (230) at a minimum polling interval defined via a configuration file or defined by the observation request.

14. The method of claim 13, further comprising, for each observed resource device (230):

comparing a newly returned payload with a stored payload; and

when the newly returned payload is in accordance with defined criteria, storing the newly returned payload (462).

15. The method of claim 12, wherein, for a large payload, the storing includes storing a hash of the large payload to avoid memory exhaustion.

16. The method of claim 1, further comprising:

updating the standardization definitions.

17. The method of claim 16, wherein the standardization definitions are updated via use of management uniform resource identifiers (URLs).

18. A computer-readable medium (1745) having computer program logic (1753) stored thereon, the computer program logic including instructions to cause a resource management agent (218, 318) run by a processor (114) of a resource management device (108, 308) to execute the method of any of claims 1-17.

19. A resource management device (108, 308) comprising:

at least one processor (114, 1741);

a communications system (110, 1749) communicably coupled with the at least one processor, the communications system capable of communicating with other devices over a network (104); and

a memory (116, 1743) communicably coupled with the at least one processor, the memory configured to store:

at least one handler unit (228, 328) configured to direct one or more of at least one resource device (230) communicably coupled with the resource management device (108, 308); and

a resource management agent (218, 318) that includes:

a transport module (220, 320) configured to prepare messages for processing, the messages being at least one of received from or to be sent to one or more of the other devices over the network via the communications system; and

an abstraction module (224, 324, 460) configured to standardize at least one of

requests from the transport module to be provided to the at least one handler unit; or

responses received from the at least one handler unit to be provided to the transport module,

using standardization definitions that are universal across different types of resource devices and their associated drivers.

20. The resource management device (108, 308) of claim 19, wherein the resource management agent (218, 318) further includes a security module (222, 322) configured to authenticate an identity of the one or more other devices that attempts to communicate with the resource management device prior to processing a request from the one or more other devices.

21. The resource management device (108, 308) of claim 20, wherein the resource management agent (218, 318) is configured to:

receive, at the transport module (220, 320), a request (332) associated with one or more of the at least one resource device (230) from one or more of the other devices over the network (104) via the communications system (110, 1749);

provide the request (338) to the abstraction module (224, 324, 460) for standardization and standardize the request (342);

provide the standardized request (344) to one or more of the at least one handler unit (228, 328);

receive a response (352) regarding the request at the abstraction module from the one or more of the at least one handler unit;

provide the response to the transport module (356) for package preparation and prepare a packaged response; and

send the packaged response (358) to the one or more of the other devices that sent the request.

22. The resource management device (108, 308) of claim 21, wherein the resource management agent (218, 318) is configured to, as part of the providing the request to the abstraction module (224, 324, 460):

determine, by the abstraction module, whether input information accompanying the request is valid (340); and

if the input information accompanying the request is invalid, send an error message to the one or more of the other devices that sent the request.

23. The resource management device (108, 308) of claim 21, wherein the request is a request for data, and the response includes at least one of:

an acknowledgment that the request was received;

requested data in accordance with standardization parameters of the standardized request; or

an error message regarding the request.

24. The resource management device (108, 308) of claim 23, wherein the standardization parameters include one or more of: a uniform resource identifier (URI), a method type, a handler definition, an input/output definition, a content type, an interface type, a unit type, a minimum indicator, a maximum indicator, or an accuracy indicator.

25. The resource management device (108, 308) of claim 23, wherein the standardized request specifies a virtual file from which the requested data is to be retrieved by the at least one handler unit (228, 328).

26. The resource management device (108, 308) of claim 21, wherein the request is a request to observe data from one or more resource devices (230).

27. The resource management device (108, 308) of claim 26, wherein the standardized request includes instructions to:

continuously poll for data from the one or more resource devices; and

return, as the response, a new payload at at least one of:

whenever data is retrieved from an observed resource device;

whenever a data value of an observed resource device changes;

whenever a data value of an observed resource device is in accordance with a specified observation condition; or

a defined polling interval.

28. The resource management device (108, 308) of claim 21, wherein the resource management agent (218, 318) is further configured to:

standardize the received response (354) by the abstraction module (224, 324, 460).

29. The resource management device (108, 308) of claim 21, wherein the request is a request for a configuration change for one or more of the resource devices (230), and the response is one of:

a confirmation that the requested configuration change was successful; and

an error message regarding the configuration change request.

30. The resource management device (108, 308) of claim 29, wherein the response is an error message and is at least one of:

an indication that the requested configuration change was unsuccessful; or

an indication that input information that accompanied the configuration change request was invalid.

31. The resource management device (108, 308) of claim 29, wherein the standardization of the configuration change request sets parameters including one or more of: a uniform resource identifier (URI), a method type, a handler definition, an input/output definition, a content type, an interface type, payload type, one or more input parameters, types of each input parameter, maximum indication for one or more input parameters, minimum indication for one or more input parameters, or type options.

32. The resource management device (108, 308) of claim 21, wherein the request is a request for activation of one or more of the resource devices (230), and the response is one of:

a confirmation that requested activation was successful; and

an error message regarding the requested activation.

33. The resource management device (108, 308) of claim 32, wherein the response is an error message and is at least one of:

an indication that the requested activation was unsuccessful; or

an indication that input information that accompanied the activation request was invalid.

34. The resource management device (108, 308) of claim 32, wherein the standardization of the activation request sets parameters including one or more of: a uniform resource identifier (URI), a method type, a handler definition, an input/output definition, a content type, an interface type, payload type, one or more input parameters, types of each input parameter, maximum indication for one or more input parameters, minimum indication for one or more input parameters, or type options.

35. The resource management device (108, 308) of claim 19, wherein the standardization involves using at least one configuration file and associated metadata.

36. The resource management device (108, 308) of claim 19, wherein the standardization involves mapping at least one uniform resource identifier (URI).

37. The resource management device (108, 308) of claim 19, wherein the standardization involves a call to one or more programs written in any programming language.

38. The resource management device (108, 308) of claim 37, wherein the standardization further involves receiving one or more outputs from the one or more called programs.

39. The resource management device (108, 308) of claim 19, wherein the abstraction module (224, 324, 460) is configured to have the standardization definitions provided, changed, or removed during updates to the resource management agent (218, 318).

40. The resource management device (108, 308) of claim 19, wherein the abstraction module (224, 324, 460) is configured to have the standardization definitions provided, changed, or removed via use of management uniform resource identifiers (URIs).

41. The resource management device (108, 308) of claim 19, wherein the resource management agent (218, 318) further includes an observation module (226, 460), wherein the observation module is configured to invoke one or more handler units (228, 328) of the at least one handler unit to retrieve and store a payload for one or more observed resource devices (230).

42. The resource management device (108, 308) of claim 41, wherein the observation module (226, 460) is configured to, as part of the invoking the one or more handler units (228, 328):

send an observation request to the one or more handler units to poll the observed resource devices (230) at a minimum polling interval defined via a configuration file or defined by the observation request.

43. The resource management device (108, 308) of claim 42, wherein the observation module (226, 460) is further configured to:

compare a newly returned payload with a stored payload; and

when the newly returned payload differs from the stored payload, store the newly returned payload and return the newly returned payload to the resource management agent.

44. The resource management device (108, 308) of claim 41, wherein, for a large payload, a hash of the large payload is stored to avoid memory exhaustion.

45. The resource management device (108, 308) of claim 19, wherein the at least one resource device (230) includes one or more of: at least one sensor or at least one actuator.

46. The resource management device (108, 308) of claim 45, wherein the at least one resource device (230) includes one or more of: a temperature sensor, a meter, a leak detection sensor, a motion sensor, a light brightness sensor, an air quality sensor, a solar panel error detection sensor.

47. The resource management device (108, 308) of claim 45, wherein the at least one resource device includes one or more of: a utility valve controller, a street light controller, a traffic light controller, a solar panel controller, or a heating/ventilation/air conditioning (HVAC) controller.

48. The resource management device (108, 308) of claim 19, wherein the at least one resource device (230) is embodied in the resource management device.

49. The resource management device (108, 308) of claim 19, wherein the at least one resource device (230) is external to, but communicably accessible to, the resource management device.

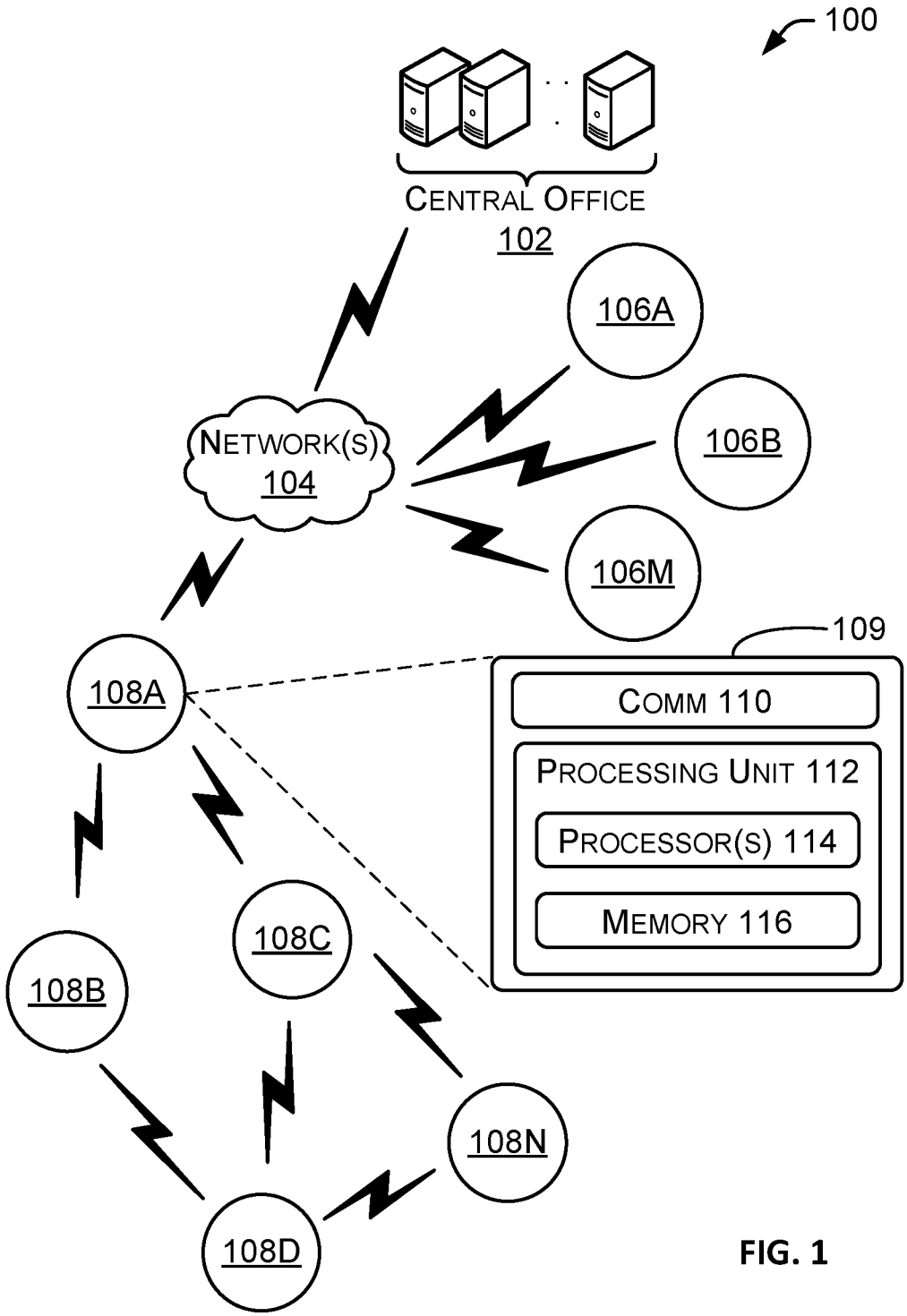

100
CENTRAL OFFICE
102
106A
106B
106M
NETWORK(S)
104
109
108A
COMM 110
PROCESSING UNIT 112
PROCESSOR(S) 114
MEMORY 116
108B
108C
108N
108D

FIG. 1

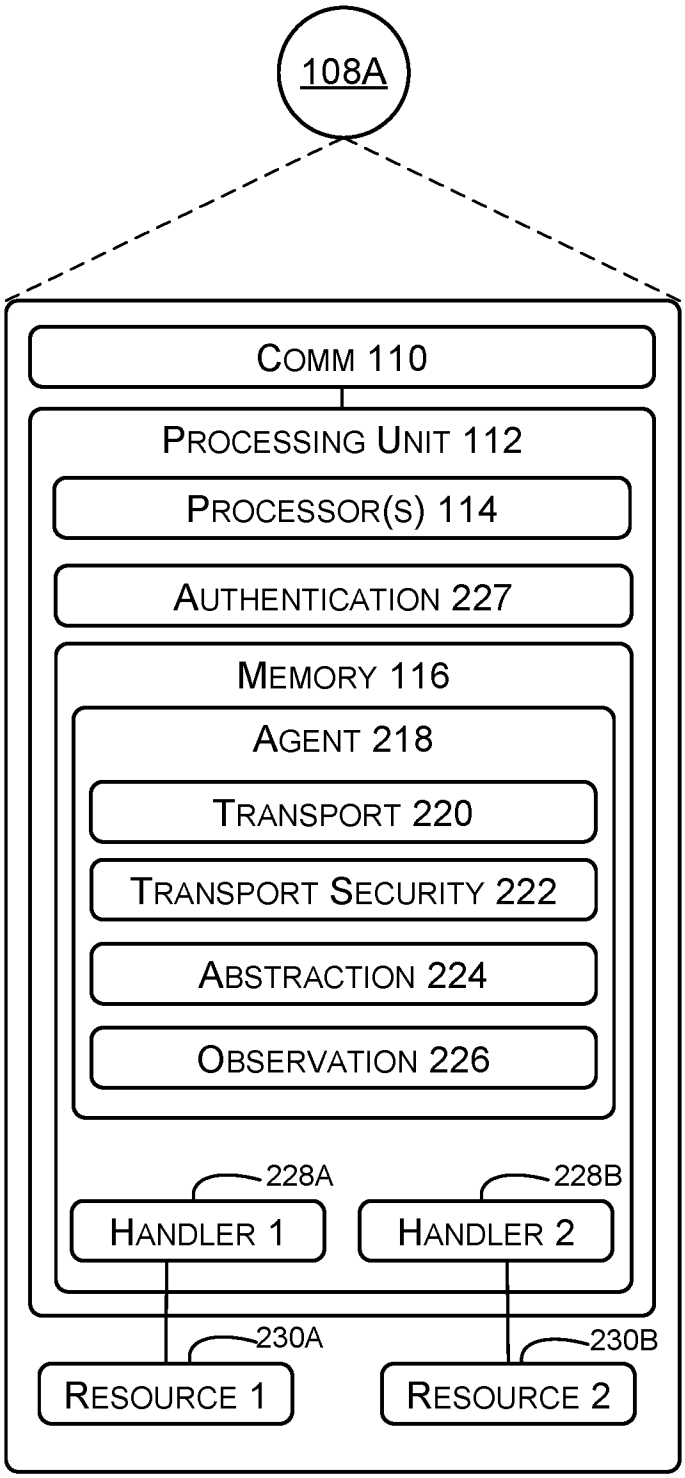

108A
COMM 110
PROCESSING UNIT 112
PROCESSOR(S) 114
AUTHENTICATION 227
MEMORY 116
AGENT 218
TRANSPORT 220
TRANSPORT SECURITY 222
ABSTRACTION 224
OBSERVATION 226
228A
228B
HANDLER 1
HANDLER 2
230A
230B
RESOURCE 1
RESOURCE 2

FIG. 2

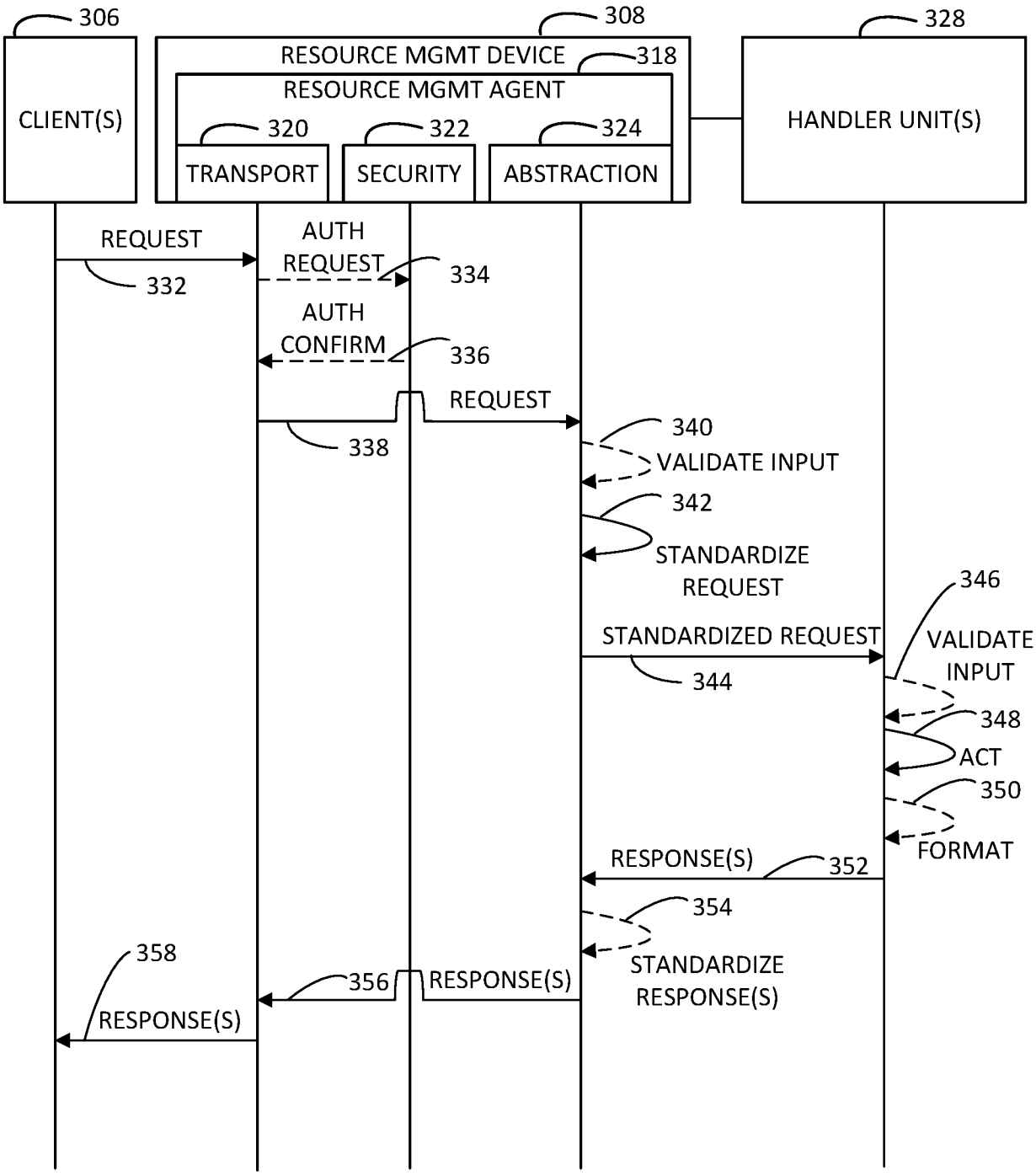

306
CLIENT(S)
308
RESOURCE MGMT DEVICE
318
RESOURCE MGMT AGENT
320
TRANSPORT
322
SECURITY
324
ABSTRACTION
328
HANDLER UNIT(S)
REQUEST
332
AUTH REQUEST
334
AUTH CONFIRM
336
REQUEST
338
340
VALIDATE INPUT
342
STANDARDIZE REQUEST
346
STANDARDIZED REQUEST
344
VALIDATE INPUT
348
ACT
350
FORMAT
RESPONSE(S)
352
354
STANDARDIZE RESPONSE(S)
358
RESPONSE(S)
356
RESPONSE(S)

FIG. 3

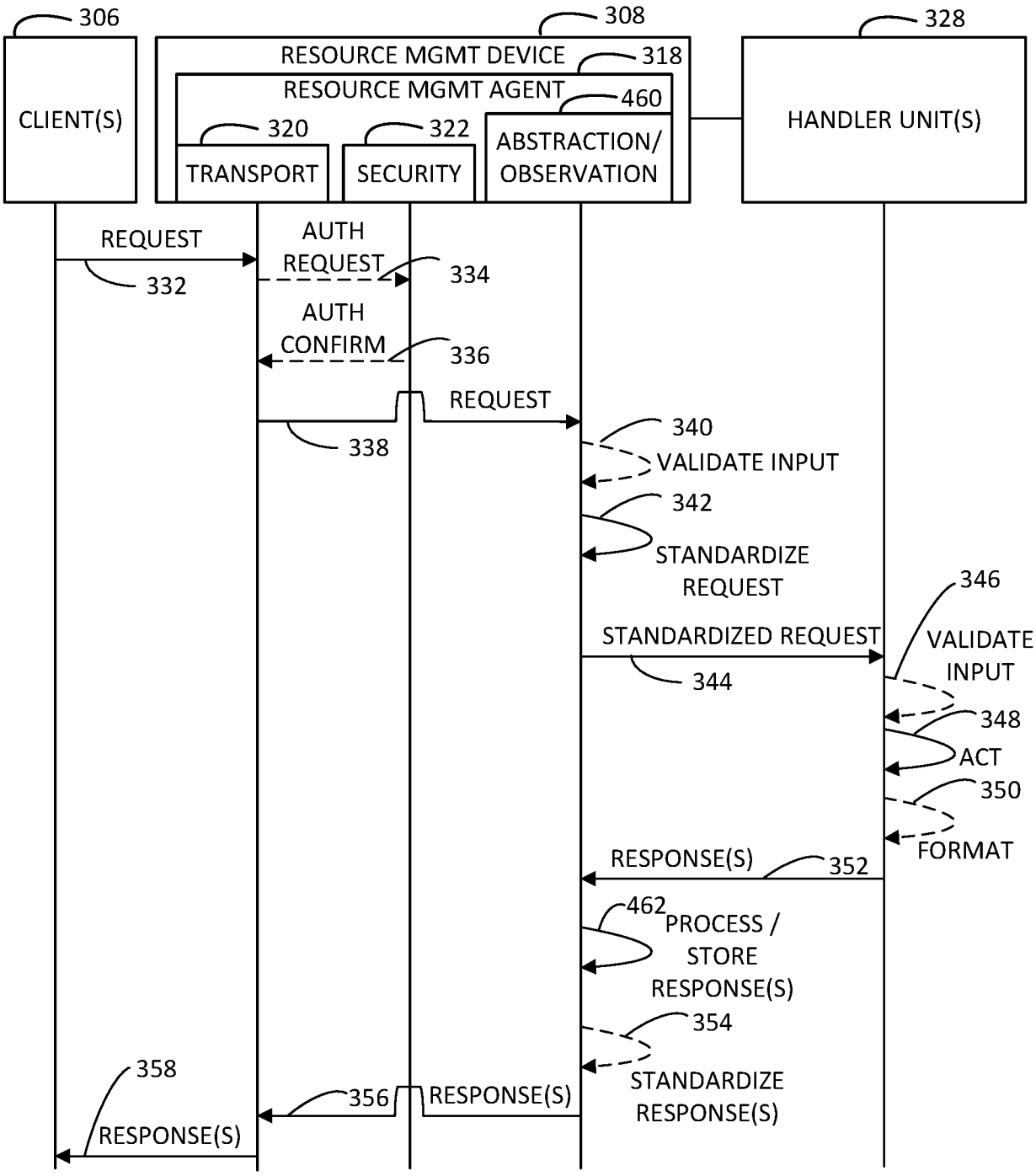

306
CLIENT(S)
308
RESOURCE MGMT DEVICE
318
RESOURCE MGMT AGENT
460
320
TRANSPORT
322
SECURITY
ABSTRACTION/
OBSERVATION
328
HANDLER UNIT(S)
REQUEST
332
AUTH
REQUEST
334
AUTH
CONFIRM
336
REQUEST
338
340
VALIDATE INPUT
342
STANDARDIZE
REQUEST
346
STANDARDIZED REQUEST
344
VALIDATE
INPUT
348
ACT
350
FORMAT
RESPONSE(S)
352
462
PROCESS /
STORE
RESPONSE(S)
354
STANDARDIZE
RESPONSE(S)
358
RESPONSE(S)
356
RESPONSE(S)

FIG. 4

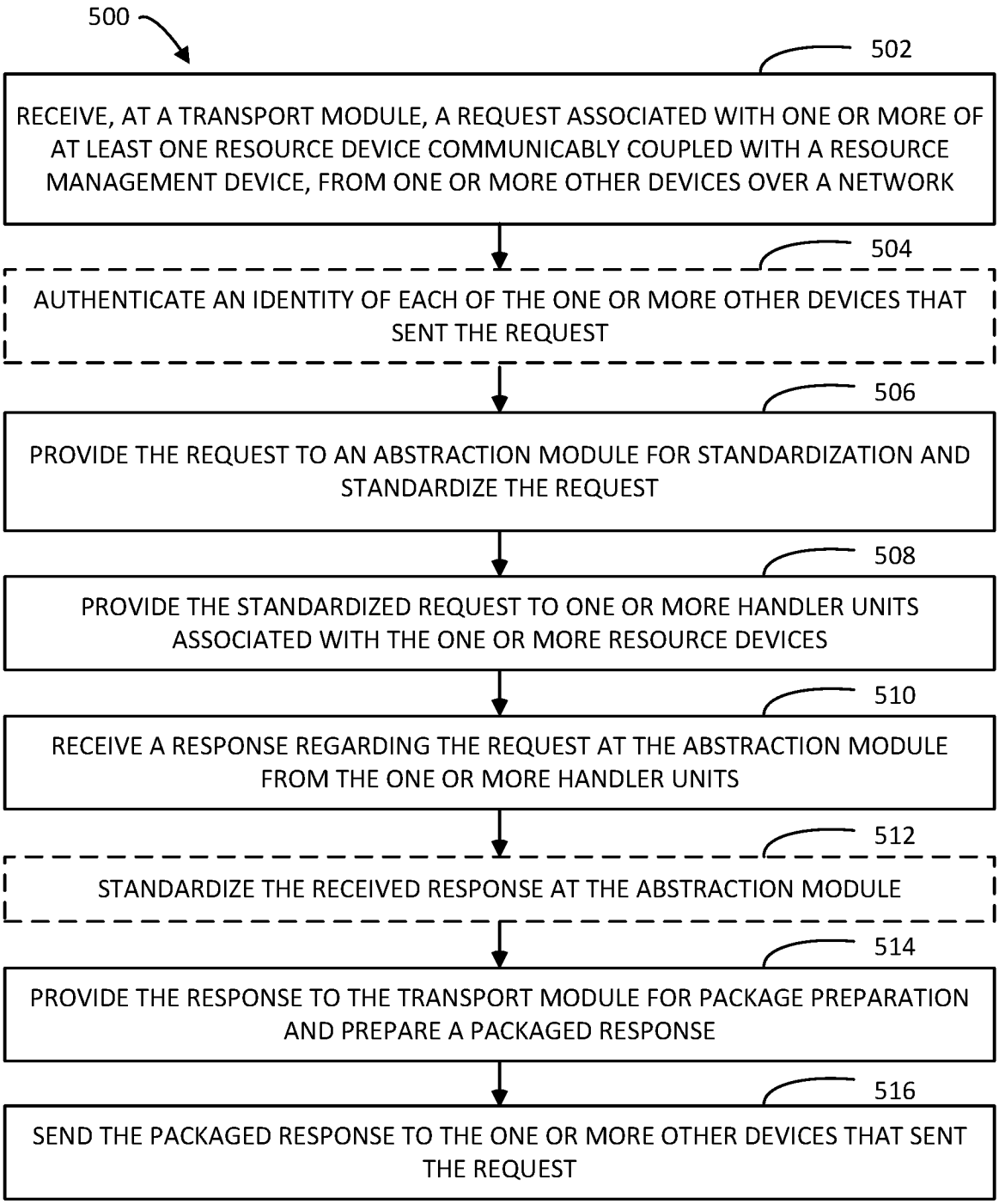

500
502
RECEIVE, AT A TRANSPORT MODULE, A REQUEST ASSOCIATED WITH ONE OR MORE OF AT LEAST ONE RESOURCE DEVICE COMMUNICABLY COUPLED WITH A RESOURCE MANAGEMENT DEVICE, FROM ONE OR MORE OTHER DEVICES OVER A NETWORK
504
AUTHENTICATE AN IDENTITY OF EACH OF THE ONE OR MORE OTHER DEVICES THAT SENT THE REQUEST
506
PROVIDE THE REQUEST TO AN ABSTRACTION MODULE FOR STANDARDIZATION AND STANDARDIZE THE REQUEST
508
PROVIDE THE STANDARDIZED REQUEST TO ONE OR MORE HANDLER UNITS ASSOCIATED WITH THE ONE OR MORE RESOURCE DEVICES
510
RECEIVE A RESPONSE REGARDING THE REQUEST AT THE ABSTRACTION MODULE FROM THE ONE OR MORE HANDLER UNITS
512
STANDARDIZE THE RECEIVED RESPONSE AT THE ABSTRACTION MODULE
514
PROVIDE THE RESPONSE TO THE TRANSPORT MODULE FOR PACKAGE PREPARATION AND PREPARE A PACKAGED RESPONSE
516
SEND THE PACKAGED RESPONSE TO THE ONE OR MORE OTHER DEVICES THAT SENT THE REQUEST

FIG. 5

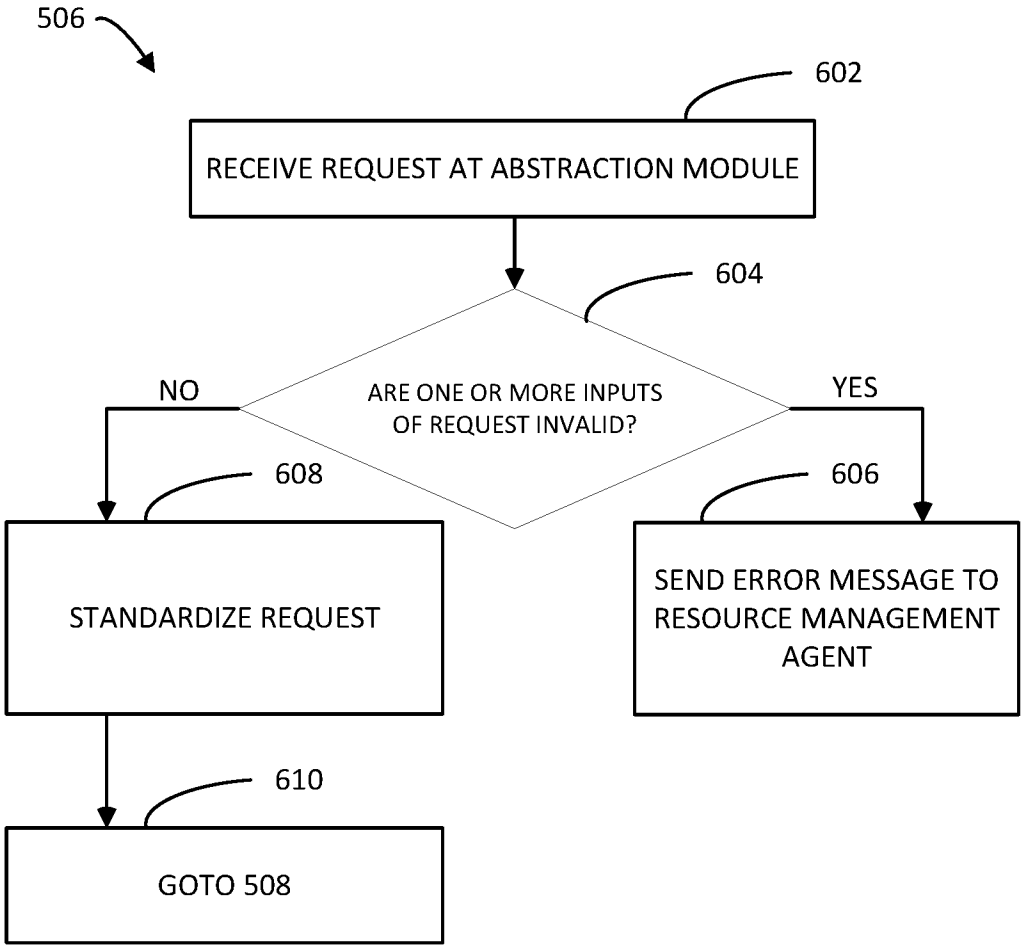

506
602
RECEIVE REQUEST AT ABSTRACTION MODULE
604
ARE ONE OR MORE INPUTS OF REQUEST INVALID?
NO
YES
608
606
STANDARDIZE REQUEST
SEND ERROR MESSAGE TO RESOURCE MANAGEMENT AGENT
610
GOTO 508

FIG. 6

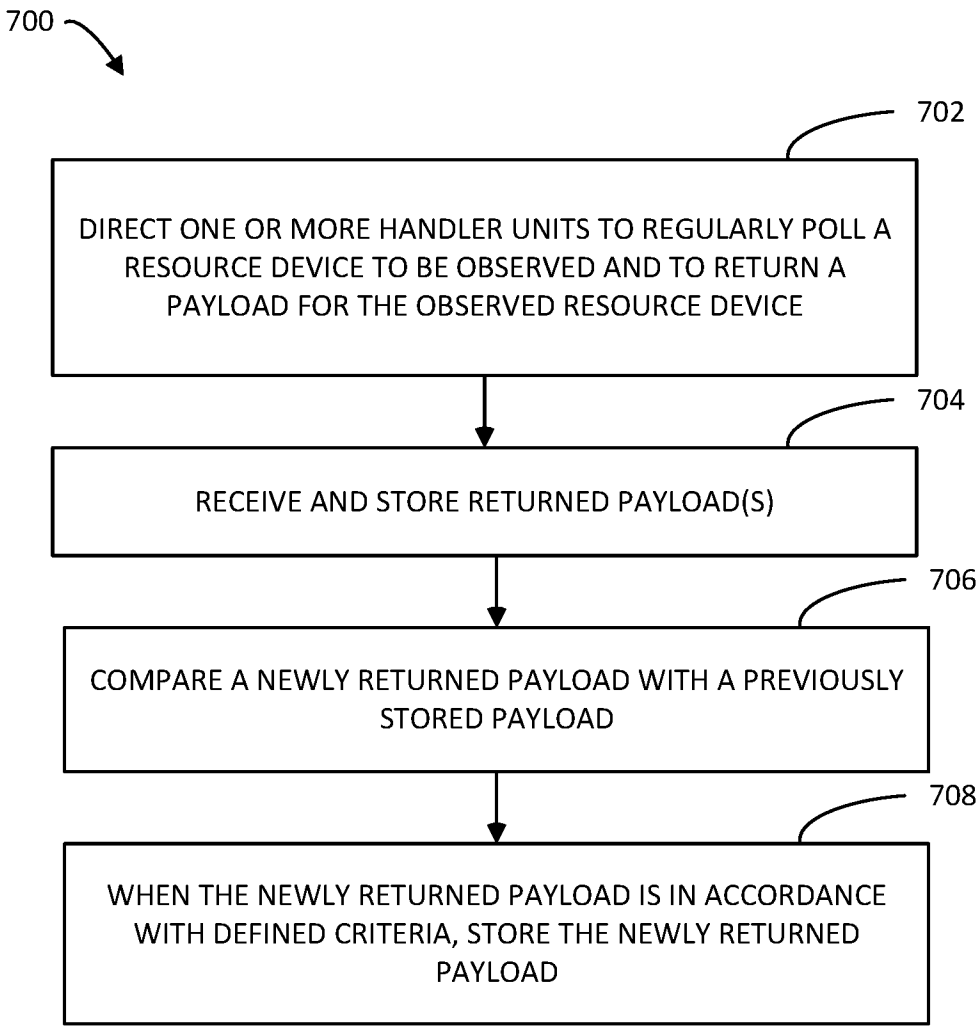

700
702
DIRECT ONE OR MORE HANDLER UNITS TO REGULARLY POLL A RESOURCE DEVICE TO BE OBSERVED AND TO RETURN A PAYLOAD FOR THE OBSERVED RESOURCE DEVICE
704
RECEIVE AND STORE RETURNED PAYLOAD(S)
706
COMPARE A NEWLY RETURNED PAYLOAD WITH A PREVIOUSLY STORED PAYLOAD
708
WHEN THE NEWLY RETURNED PAYLOAD IS IN ACCORDANCE WITH DEFINED CRITERIA, STORE THE NEWLY RETURNED PAYLOAD

FIG. 7

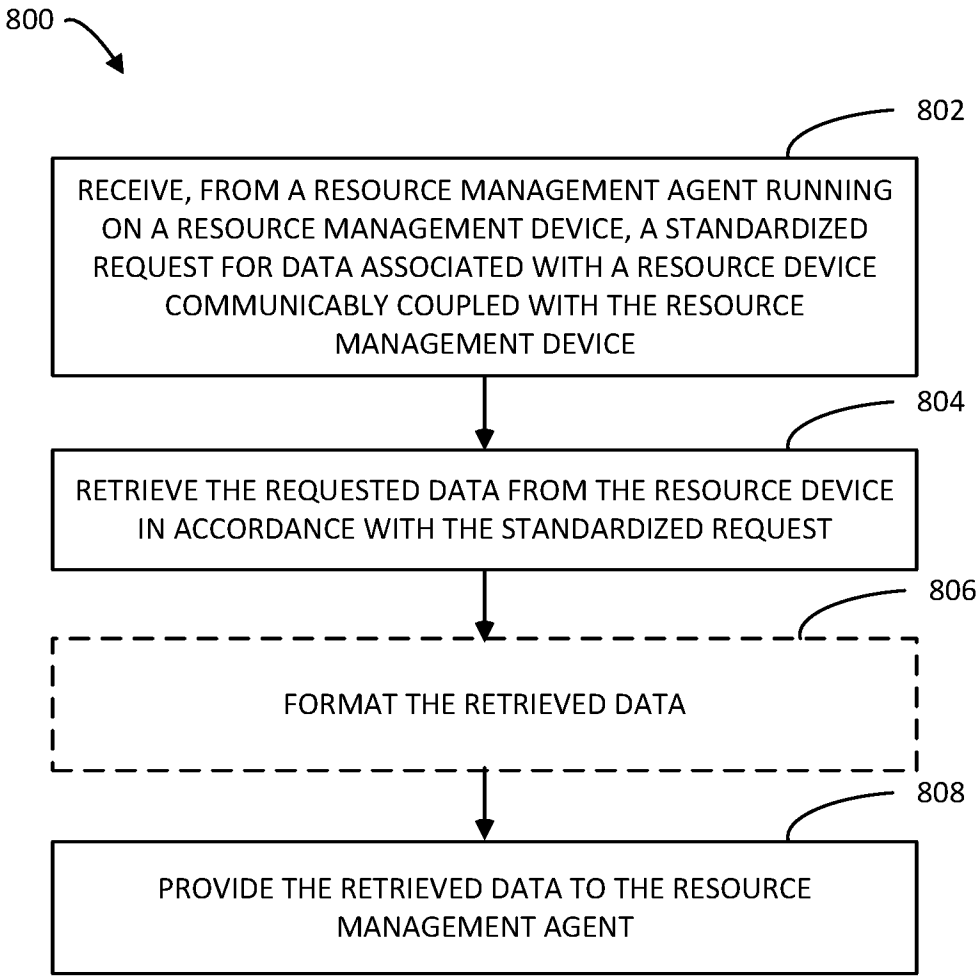

800
802
RECEIVE, FROM A RESOURCE MANAGEMENT AGENT RUNNING ON A RESOURCE MANAGEMENT DEVICE, A STANDARDIZED REQUEST FOR DATA ASSOCIATED WITH A RESOURCE DEVICE COMMUNICABLY COUPLED WITH THE RESOURCE MANAGEMENT DEVICE
804
RETRIEVE THE REQUESTED DATA FROM THE RESOURCE DEVICE IN ACCORDANCE WITH THE STANDARDIZED REQUEST
806
FORMAT THE RETRIEVED DATA
808
PROVIDE THE RETRIEVED DATA TO THE RESOURCE MANAGEMENT AGENT

FIG. 8

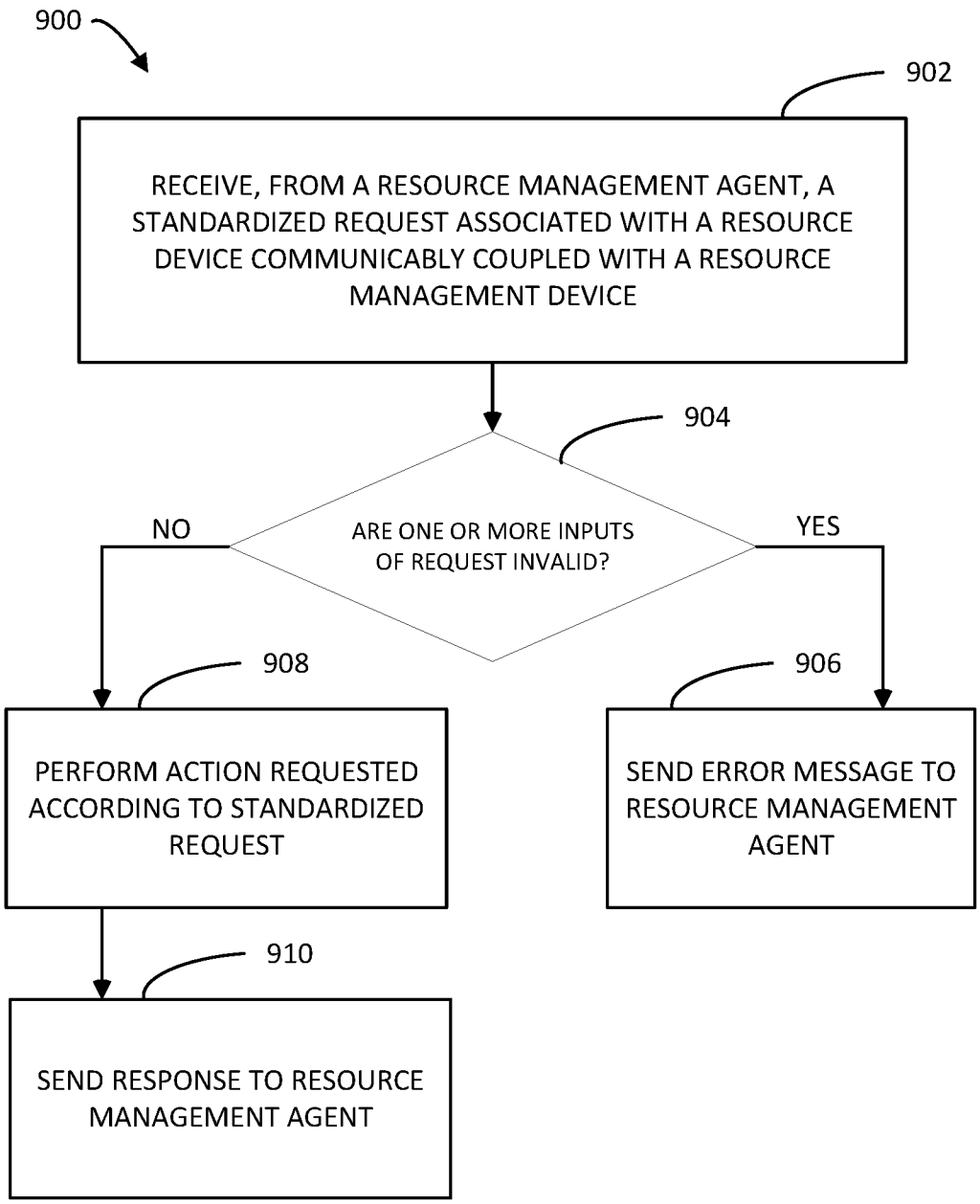

900
902
RECEIVE, FROM A RESOURCE MANAGEMENT AGENT, A STANDARDIZED REQUEST ASSOCIATED WITH A RESOURCE DEVICE COMMUNICABLY COUPLED WITH A RESOURCE MANAGEMENT DEVICE
904
ARE ONE OR MORE INPUTS OF REQUEST INVALID?
NO
YES
908
PERFORM ACTION REQUESTED ACCORDING TO STANDARDIZED REQUEST
906
SEND ERROR MESSAGE TO RESOURCE MANAGEMENT AGENT
910
SEND RESPONSE TO RESOURCE MANAGEMENT AGENT

FIG. 9

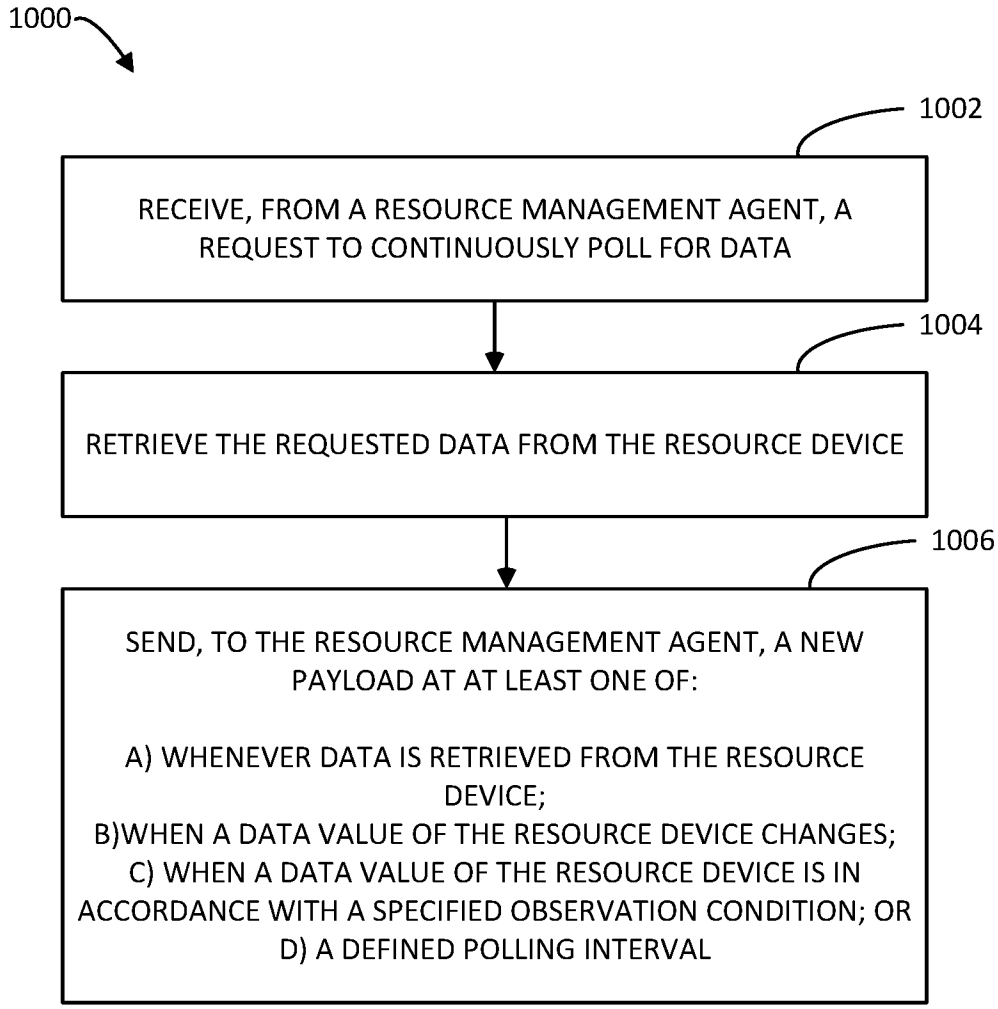

1000
1002
RECEIVE, FROM A RESOURCE MANAGEMENT AGENT, A REQUEST TO CONTINUOUSLY POLL FOR DATA
1004
RETRIEVE THE REQUESTED DATA FROM THE RESOURCE DEVICE
1006
SEND, TO THE RESOURCE MANAGEMENT AGENT, A NEW PAYLOAD AT AT LEAST ONE OF:
A) WHENEVER DATA IS RETRIEVED FROM THE RESOURCE DEVICE;
B)WHEN A DATA VALUE OF THE RESOURCE DEVICE CHANGES;
C) WHEN A DATA VALUE OF THE RESOURCE DEVICE IS IN ACCORDANCE WITH A SPECIFIED OBSERVATION CONDITION; OR
D) A DEFINED POLLING INTERVAL

FIG. 10

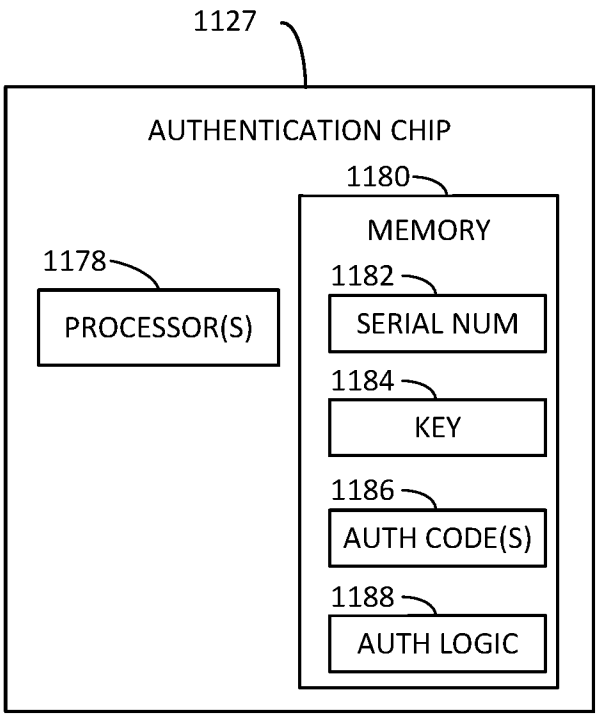

1127
AUTHENTICATION CHIP
1180
MEMORY
1178
PROCESSOR(S)
1182
SERIAL NUM
1184
KEY
1186
AUTH CODE(S)
1188
AUTH LOGIC

FIG. 11

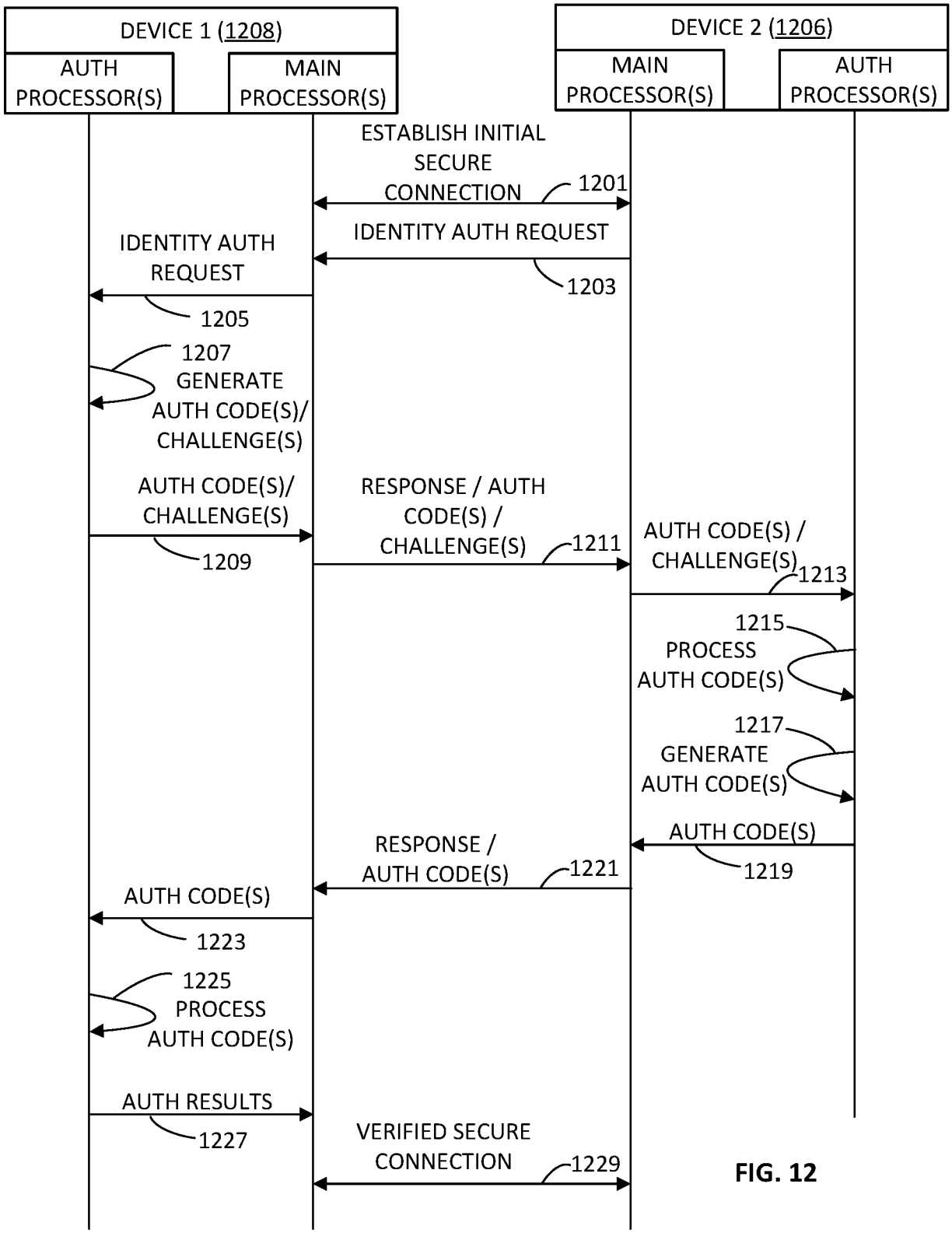

DEVICE 1 (1208)
AUTH PROCESSOR(S)
MAIN PROCESSOR(S)
DEVICE 2 (1206)
MAIN PROCESSOR(S)
AUTH PROCESSOR(S)
ESTABLISH INITIAL SECURE CONNECTION
1201
IDENTITY AUTH REQUEST
1203
IDENTITY AUTH REQUEST
1205
1207
GENERATE AUTH CODE(S)/ CHALLENGE(S)
AUTH CODE(S)/ CHALLENGE(S)
1209
RESPONSE / AUTH CODE(S) / CHALLENGE(S)
1211
AUTH CODE(S) / CHALLENGE(S)
1213
1215
PROCESS AUTH CODE(S)
1217
GENERATE AUTH CODE(S)
AUTH CODE(S)
1219
RESPONSE / AUTH CODE(S)
1221
AUTH CODE(S)
1223
1225
PROCESS AUTH CODE(S)
AUTH RESULTS
1227
VERIFIED SECURE CONNECTION
1229

FIG. 12

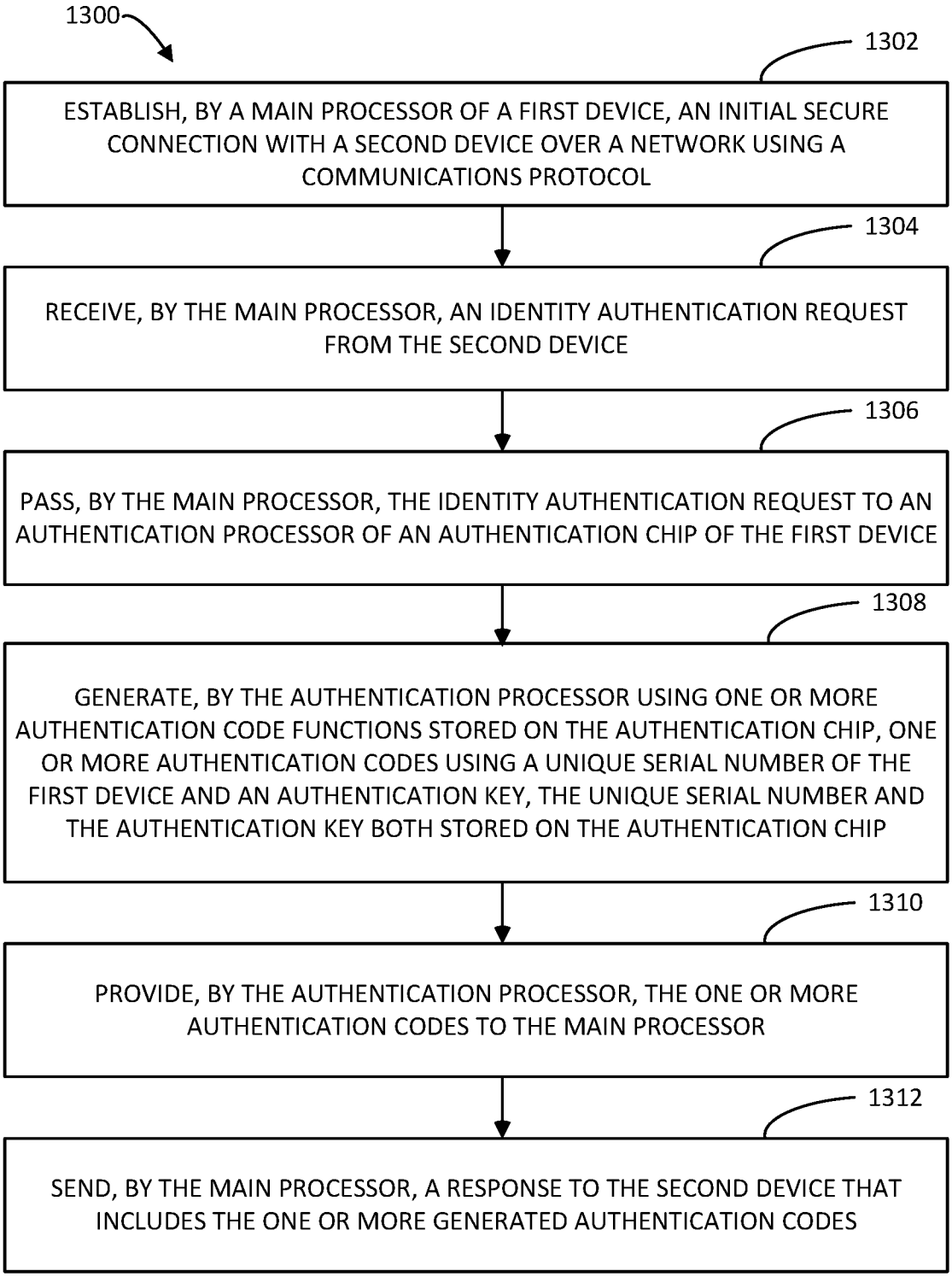

1300
1302
ESTABLISH, BY A MAIN PROCESSOR OF A FIRST DEVICE, AN INITIAL SECURE CONNECTION WITH A SECOND DEVICE OVER A NETWORK USING A COMMUNICATIONS PROTOCOL
1304
RECEIVE, BY THE MAIN PROCESSOR, AN IDENTITY AUTHENTICATION REQUEST FROM THE SECOND DEVICE
1306
PASS, BY THE MAIN PROCESSOR, THE IDENTITY AUTHENTICATION REQUEST TO AN AUTHENTICATION PROCESSOR OF AN AUTHENTICATION CHIP OF THE FIRST DEVICE
1308
GENERATE, BY THE AUTHENTICATION PROCESSOR USING ONE OR MORE AUTHENTICATION CODE FUNCTIONS STORED ON THE AUTHENTICATION CHIP, ONE OR MORE AUTHENTICATION CODES USING A UNIQUE SERIAL NUMBER OF THE FIRST DEVICE AND AN AUTHENTICATION KEY, THE UNIQUE SERIAL NUMBER AND THE AUTHENTICATION KEY BOTH STORED ON THE AUTHENTICATION CHIP
1310
PROVIDE, BY THE AUTHENTICATION PROCESSOR, THE ONE OR MORE AUTHENTICATION CODES TO THE MAIN PROCESSOR
1312
SEND, BY THE MAIN PROCESSOR, A RESPONSE TO THE SECOND DEVICE THAT INCLUDES THE ONE OR MORE GENERATED AUTHENTICATION CODES

FIG. 13

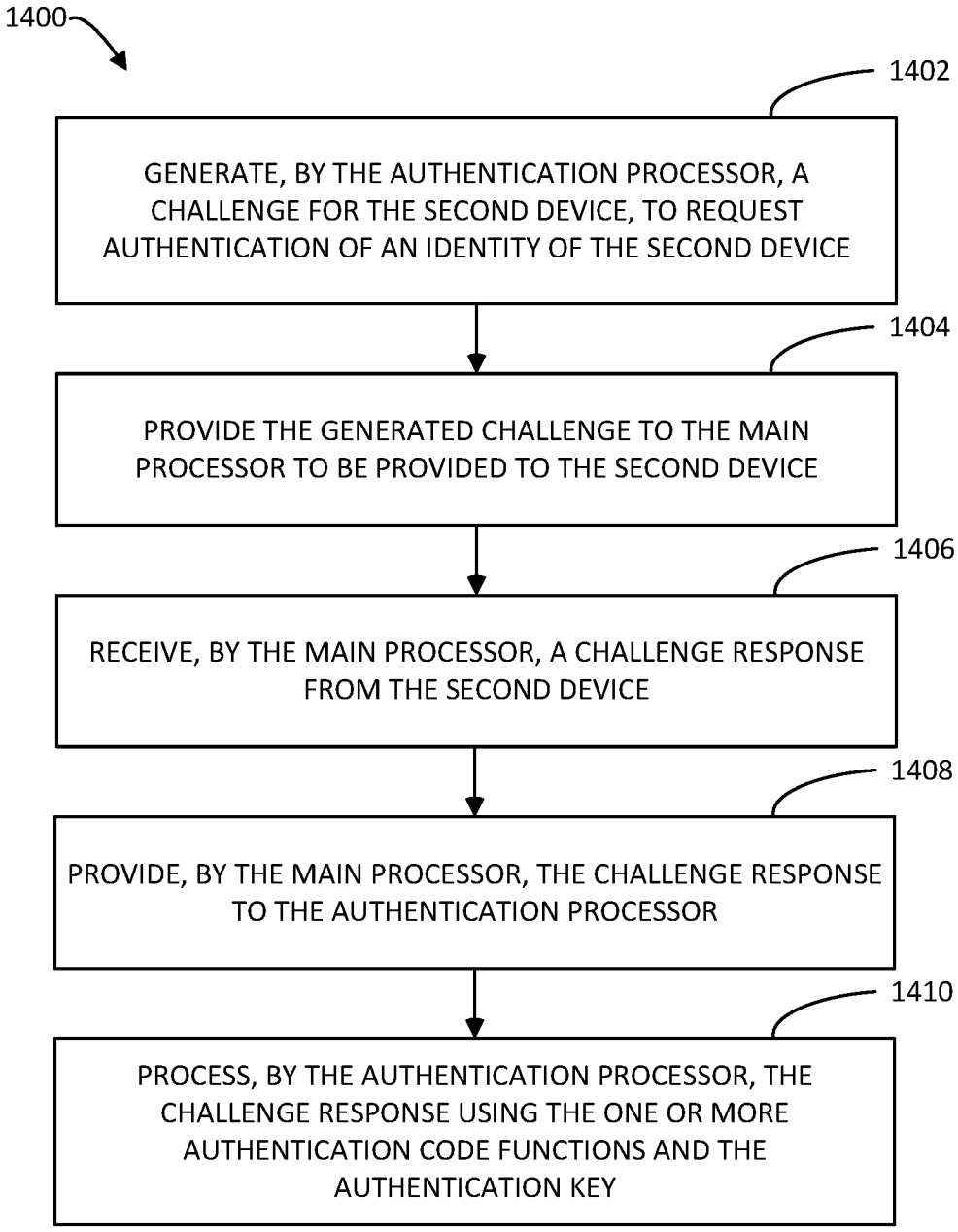

1400
1402
GENERATE, BY THE AUTHENTICATION PROCESSOR, A CHALLENGE FOR THE SECOND DEVICE, TO REQUEST AUTHENTICATION OF AN IDENTITY OF THE SECOND DEVICE
1404
PROVIDE THE GENERATED CHALLENGE TO THE MAIN PROCESSOR TO BE PROVIDED TO THE SECOND DEVICE
1406
RECEIVE, BY THE MAIN PROCESSOR, A CHALLENGE RESPONSE FROM THE SECOND DEVICE
1408
PROVIDE, BY THE MAIN PROCESSOR, THE CHALLENGE RESPONSE TO THE AUTHENTICATION PROCESSOR
1410
PROCESS, BY THE AUTHENTICATION PROCESSOR, THE CHALLENGE RESPONSE USING THE ONE OR MORE AUTHENTICATION CODE FUNCTIONS AND THE AUTHENTICATION KEY

FIG. 14

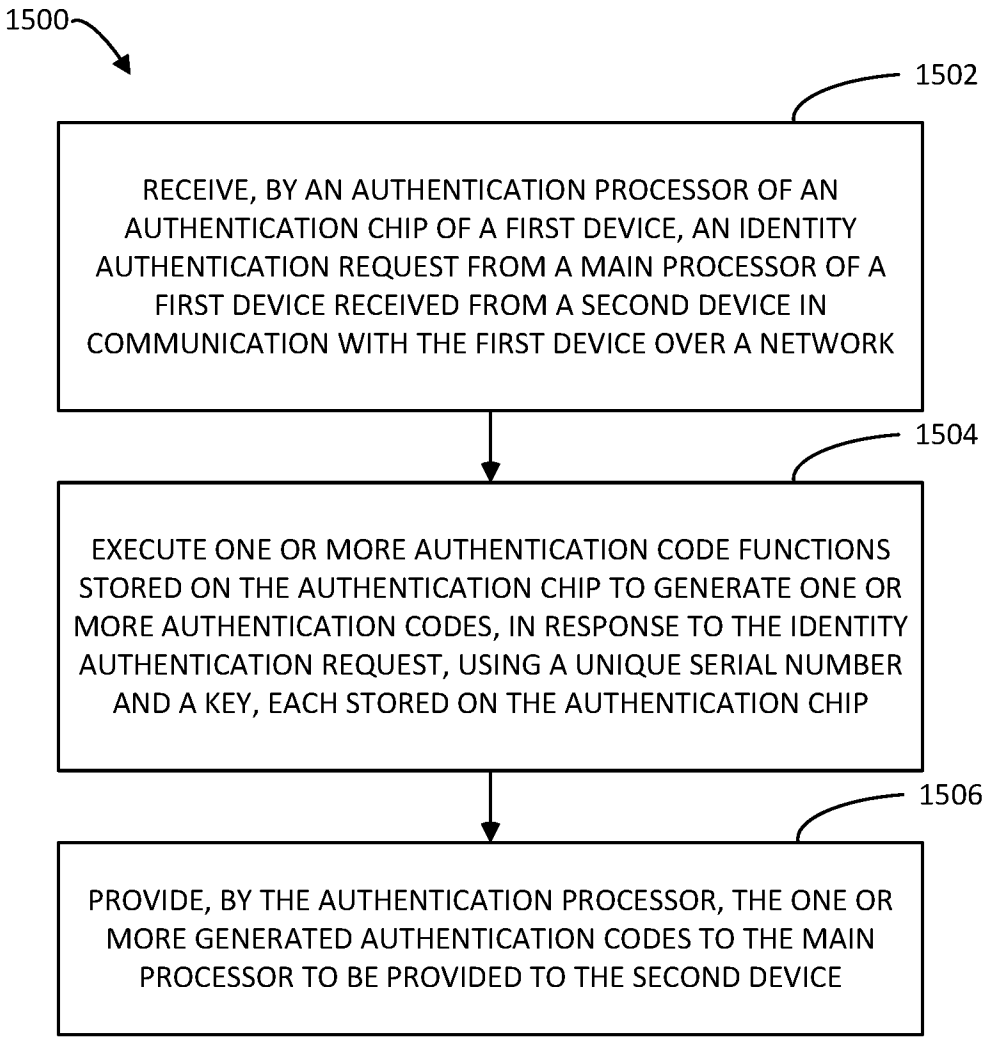

1500
1502
RECEIVE, BY AN AUTHENTICATION PROCESSOR OF AN AUTHENTICATION CHIP OF A FIRST DEVICE, AN IDENTITY AUTHENTICATION REQUEST FROM A MAIN PROCESSOR OF A FIRST DEVICE RECEIVED FROM A SECOND DEVICE IN COMMUNICATION WITH THE FIRST DEVICE OVER A NETWORK
1504
EXECUTE ONE OR MORE AUTHENTICATION CODE FUNCTIONS STORED ON THE AUTHENTICATION CHIP TO GENERATE ONE OR MORE AUTHENTICATION CODES, IN RESPONSE TO THE IDENTITY AUTHENTICATION REQUEST, USING A UNIQUE SERIAL NUMBER AND A KEY, EACH STORED ON THE AUTHENTICATION CHIP
1506
PROVIDE, BY THE AUTHENTICATION PROCESSOR, THE ONE OR MORE GENERATED AUTHENTICATION CODES TO THE MAIN PROCESSOR TO BE PROVIDED TO THE SECOND DEVICE

FIG. 15

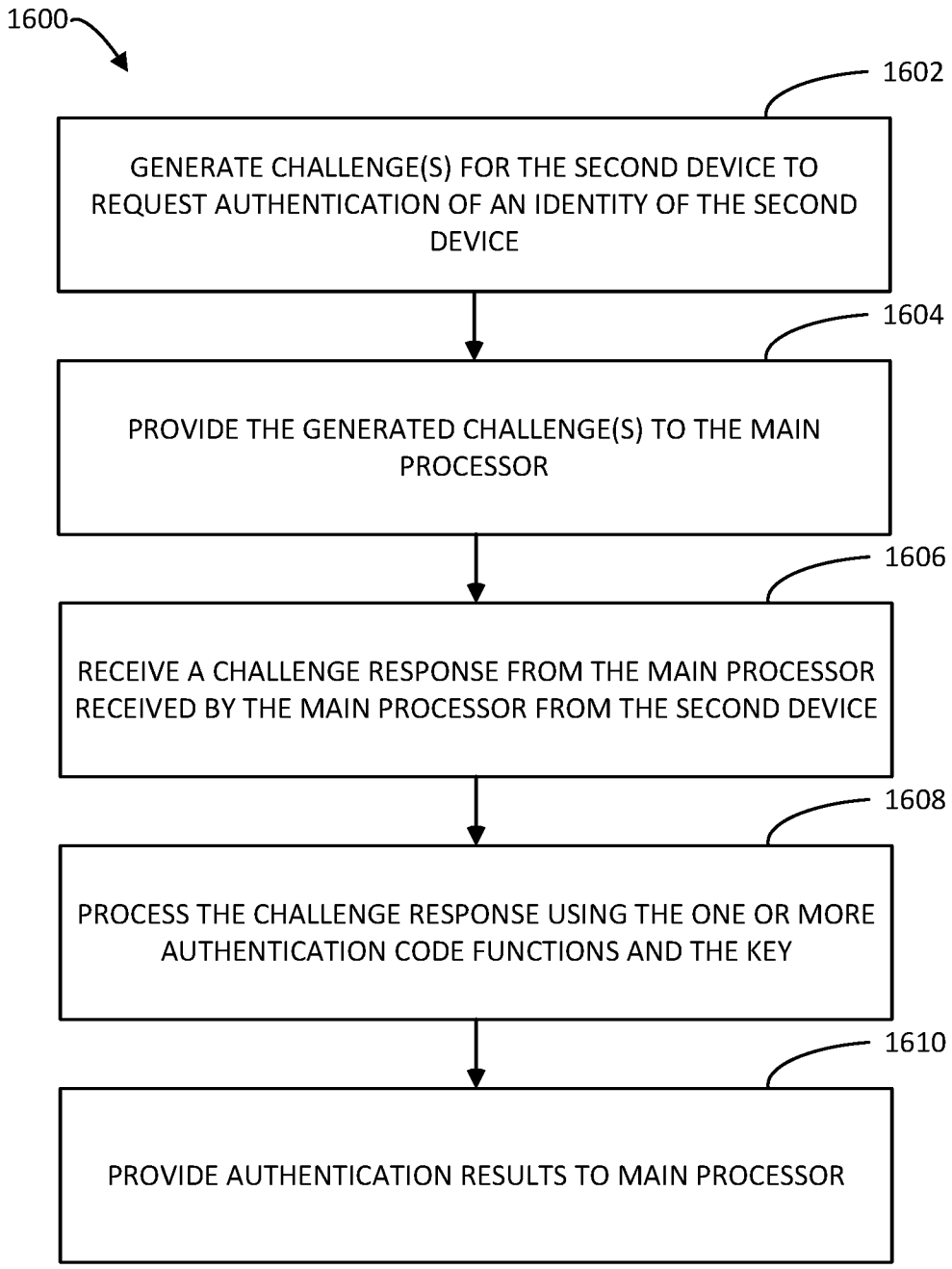

1600
1602
GENERATE CHALLENGE(S) FOR THE SECOND DEVICE TO REQUEST AUTHENTICATION OF AN IDENTITY OF THE SECOND DEVICE
1604
PROVIDE THE GENERATED CHALLENGE(S) TO THE MAIN PROCESSOR
1606
RECEIVE A CHALLENGE RESPONSE FROM THE MAIN PROCESSOR RECEIVED BY THE MAIN PROCESSOR FROM THE SECOND DEVICE
1608
PROCESS THE CHALLENGE RESPONSE USING THE ONE OR MORE AUTHENTICATION CODE FUNCTIONS AND THE KEY
1610
PROVIDE AUTHENTICATION RESULTS TO MAIN PROCESSOR

FIG. 16

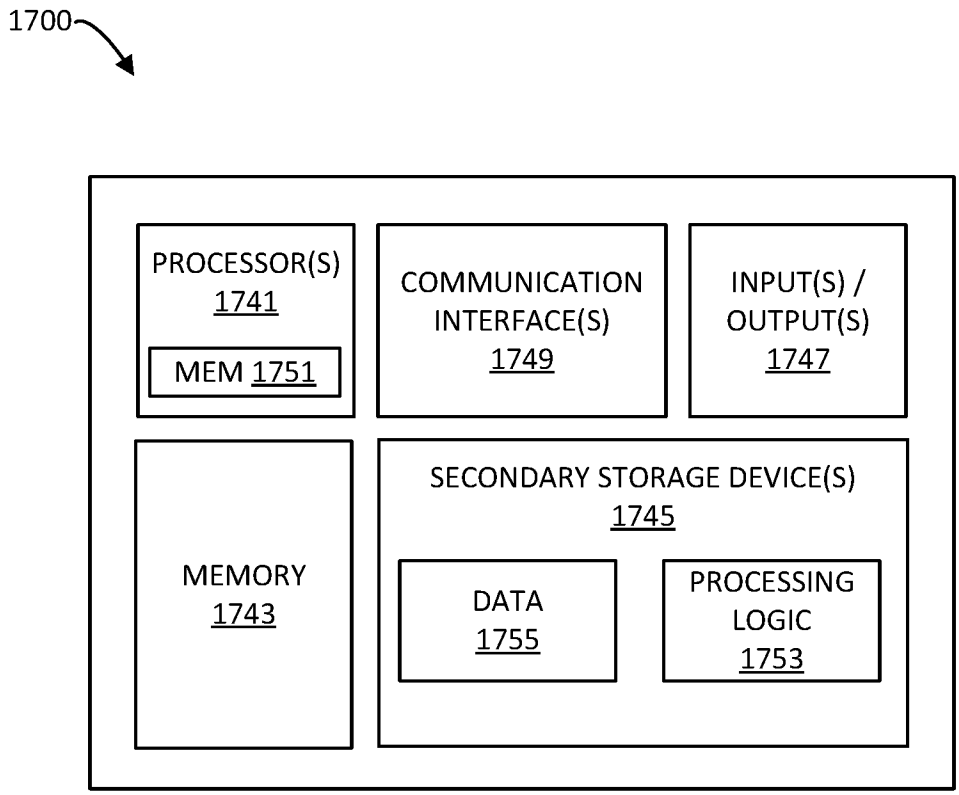

1700
PROCESSOR(S)
1741
MEM 1751
COMMUNICATION
INTERFACE(S)
1749
INPUT(S) /
OUTPUT(S)
1747
MEMORY
1743
SECONDARY STORAGE DEVICE(S)
1745
DATA
1755
PROCESSING
LOGIC
1753

FIG. 17

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2016/067545

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L12/24 G06F9/44 H04L29/08
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	VAN DER STOK CONSULTANT A BIERMAN YUMAWORKS J SCHOENWAELDER JACOBS UNIVERSITY A SEHGAL CONSULTANT P: "CoAP Management Interface; draft-vanderstok-core-comi-08.txt", COAP MANAGEMENT INTERFACE; DRAFT-VANDERSTOK-CORE-COMI-08.TXT, INTERNET ENGINEERING TASK FORCE, IETF; STANDARDWORKINGDRAFT, INTERNET SOCIETY (ISOC) 4, RUE DES FALAISES CH- 1205 GENEVA, SWITZERLAND, 16 October 2015 (2015-10-16), pages 1-74, XP015109073, [retrieved on 2015-10-16] page 3 pages 8-9 pages 15-17 page 51 ----- -/--	1-49

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
13 March 2017	24/03/2017

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Kamps, Stefan

1

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2016/067545

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2004/025173 A1 (LEVONAI GIL [US] ET AL) 5 February 2004 (2004-02-05) paragraph [0139]; figure 3 paragraph [0152]; figure 4	1-49
Y	Anonymous: "Android Sensor Integration Part 2: Sensor Readings ¦ inovex-Blog", , 11 December 2015 (2015-12-11), XP055353487, Retrieved from the Internet: URL:https://www.inovex.de/blog/android-sensor-integration-part-2-sensor-readings/ [retrieved on 2017-03-10] page 3	25
Y	Anonymous: "Raspberry Pi - View topic - Sharing my use of I2C GPIOs", , 5 May 2014 (2014-05-05), XP055353506, Retrieved from the Internet: URL:https://www.raspberrypi.org/forums/viewtopic.php?f=44&t=76468 [retrieved on 2017-03-10] page 2	25
A	Anonymous: "Constrained Application Protocol - Wikipedia", , 19 November 2015 (2015-11-19), XP055353067, Retrieved from the Internet: URL:https://en.wikipedia.org/w/index.php?title=Constrained_Application_Protocol&oldid=691395715 [retrieved on 2017-03-09] the whole document	1-49

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2016/067545

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004025173 A1	05-02-2004	NONE	