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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 July 2019 (11.07.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/135973 A1

(51) International Patent Classification:
H04L 29/08 (2006.01)

(21) International Application Number:
PCT/US2018/067624

(22) International Filing Date:
27 December 2018 (27.12.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/613,312 03 January 2018 (03.01.2018) US

(71) Applicant: **CONVIDA WIRELESS, LLC** [US/US]; 200 Bellevue Parkway, Suite 300, Wilmington, DE 19809-3727 (US).

(72) Inventors: **MILADIN, Catalina, Mihaela**; 200 Bellevue Parkway, Suite 300, Wilmington, DE 19809-3727 (US). **SEED, Dale, N.**; 200 Bellevue Parkway, Suite 300, Wilmington, DE 19809-3727 (US). **FLYNN, William, Robert IV**; 200 Bellevue Parkway, Suite 300, Wilmington, DE 19809-3727 (US). **DI GIROLAMO, Rocco**; 200 Bellevue Parkway, Suite 300, Wilmington, DE 19809-3727 (US). **CHEN, Zhuo**; 200 Bellevue Parkway, Suite 300, Wilmington, DE 19809-3727 (US). **LY, Quang**; 200 Bellevue Parkway, Suite 300, Wilmington, DE 19809-3727 (US). **LOEB, Shoshana**; 127 West Chestnut Hill Avenue, Philadelphia, PA 19118 (US).

(74) Agent: **SAMUELS, Steven, B.** et al.; Baker & Hostetler LLP, 2929 Arch Street, Cira Centre, 12th Floor, Philadelphia, PA 19104-2891 (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, JO, 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,

(54) Title: FRAMEWORK FOR INTEGRATION OF PASSIVE OBJECTS IN THE IOT/M2M SERVICE LAYER

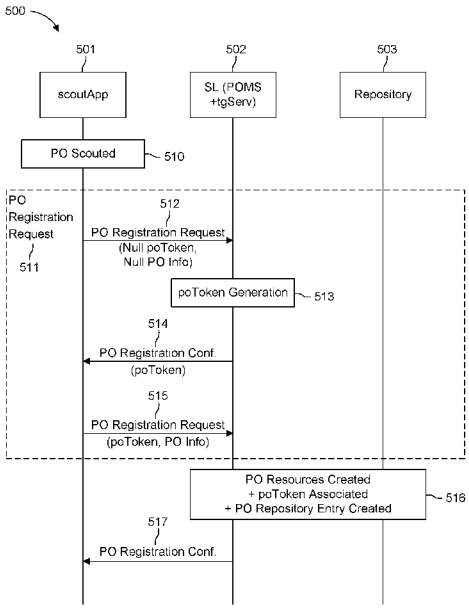

FIG. 5

(57) Abstract: Methods and apparatuses for recognizing, identifying, registering, and arbitrating passive objects (POs) are described herein. In accordance with one embodiment, an apparatus may receive, from a computing device, a passive object registration request message that includes an identifier associated with a passive object, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities. The apparatus may generate a passive object token that includes the identifier associated with the passive object and information to enable control and management of the passive object by a plurality of service layer applications. The apparatus may store, in a database, the passive object token with an associated resource to enable access by the plurality of service layer applications. The apparatus may transmit, to the computing device, a registration confirmation message that includes the passive object token.

WO 2019/135973 A1

[Continued on next page]

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

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

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

FRAMEWORK FOR INTEGRATION OF PASSIVE OBJECTS IN THE IOT/M2M SERVICE LAYER

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Patent Application No. 62/613,312, filed January 3, 2018, which is hereby incorporated by reference in its entirety.

BACKGROUND

[0002] Objects not defined by (e.g. without) computational, telecommunications, and data sharing capabilities (termed herein passive objects (POs)) are not entities of Internet of Things (IoT) systems. Rather POs are recognized only from the perspective of their relationships to other devices and are employed to characterize or augment the entities of the platform, i.e. the other devices.

[0003] Accordingly, there is a need to be able to exchange PO information and PO representations between applications and devices in order to enable services that are designed around POs. Examples of such services that may be designed around POs include but are not limited to the following: smart cameras with object recognition overseeing an industrial floor, robots active in a space that use material within reach, warehouse inventories, and transportation logs.

SUMMARY

[0004] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter. Furthermore, the claimed subject matter is not limited to limitations that solve any or all disadvantages noted in any part of this disclosure.

[0005] Methods and apparatuses for recognizing, identifying, and arbitrating passive objects (POs) are described herein. In accordance with one embodiment, an apparatus may receive, from a computing device, a passive object registration request message that includes an identifier associated with a passive object, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities. The apparatus may generate a passive object token that includes the identifier associated with the passive object and information to enable control and management of the passive object by a plurality of service layer applications. The apparatus may store, in a database, the passive object token with an associated resource to enable access by the plurality of service layer applications. The apparatus may transmit, to the computing device, a registration confirmation message that includes the passive object token.

[0006] In accordance with another embodiment, an apparatus may determine permissions and parameters associated with a passive object, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities. The apparatus may receive, from a plurality of service layer applications, a plurality of service requests that each include parameters associated with a service type and each include a permission identifier that includes a passive object token. The apparatus may select a service request of the received plurality of service requests to execute based on the parameters and the permission identifier included in the selected service request. The apparatus may transmit, to a service layer application of the plurality of service layer applications associated with the selected service request, a service request response that includes information associated with the executed service request.

[0007] In accordance with another embodiment, an apparatus may receive, from a service layer application of a plurality of service layer applications, a service request that includes information associated with a service type. The apparatus may determine, based on the service request, a plurality of policies to be used to determine how to identify a passive object of a type of passive objects, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities. The apparatus may identify the passive object based on the plurality of policies. The apparatus may execute the service request based on the identified passive object. The apparatus may transmit, to the service layer application, a service request response that includes information associated with the executed service request

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] In order to facilitate a more robust understanding of the application, reference is now made to the accompanying drawings, in which like elements are referenced with like numerals. These drawings should not be construed to limit the application and are intended only to be illustrative.

[0009] Fig. 1 is a diagram of an example protocol stack supporting an M2M/IoT service layer;

[0010] Fig. 2 is a diagram of an example oneM2M Service Layer that supports an initial set of common service functions (CSF);

[0011] Fig. 3 is a diagram of an example system providing a high-level perspective of the relationship between the POMS and PO-related services (management related or auxiliary) in the context of a service layer;

[0012] Fig. 4 is a diagram of an example procedure for PO registration in accordance with one embodiment;

[0013] Fig. 5 is a diagram of an example procedure for PO registration in accordance with another embodiment;

[0014] Fig. 6 is a diagram of an example procedure for pre-defined PO provisioning and registration in accordance with another embodiment;

[0015] Fig. 7 is a diagram of an example procedure for pre-defined PO provisioning and automatic PO registration in accordance with another embodiment;

[0016] Fig. 8 is a diagram of an example procedure for a PO service request in accordance with another embodiment;

[0017] Fig. 9 is a diagram of an example procedure for an auxiliary PO-related service request preceded by a poToken request in accordance with another embodiment;

[0018] Fig. 10 is a diagram of an example procedure for a *poToken* service using *poTokenReq* in accordance with another embodiment;

[0019] Fig. 11 is a diagram of an example a resource in accordance with one embodiment;

[0020] Fig. 12 is a diagram of an example procedure for remote scouting and GW *poToken* generation in accordance with one embodiment;

[0021] Fig. 13 is a diagram of an example procedure for remote scouting and GW *poToken* generation where the PO is pre-defined and provided to the platform via the repository in accordance with another embodiment;

[0022] Fig. 14 is a diagram of an example graphical user interface (GUI) for entering pre-defined POs in a repository;

[0023] Fig. 15 is a diagram of another example GUI that may be used for searching for the representations of a PO on a service layer platform or associated database;

[0024] Fig. 16A is a system diagram of an example machine-to-machine (M2M) or Internet of Things (IoT) communication system in which one or more disclosed embodiments may be implemented;

[0025] Fig. 16B is a system diagram of an example architecture that may be used within the M2M / IoT communications system illustrated in Fig. 16A;

[0026] Fig. 16C is a system diagram of an example M2M/IoT terminal or gateway device that may be used within the communications system illustrated in Fig. 16A; and

[0027] Fig. 16D is a block diagram of an example computing system in which aspects of the communication system of Fig. 16A may be embodied.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0028] A Passive Object Management Service (POMS), which an IoT/M2M service layer may provide to its registrants, is described in the examples and embodiments herein. The POMS may enable service layer platforms to exchange passive object (PO) information and PO representations between applications and devices to enable services designed around POs between applications.

[0029] The following is a list of definitions of terms as used herein:

[0030] Access Control Policy: a set of privileges that are represented by access control rules.

[0031] Access decision: authorization reached when an entity's Privileges are evaluated.

[0032] Augmented reality: Technology that simulates a virtual or imaginary environment by superimposing a computer-generated image on a user's view of the real world, thus providing a composite view. AR aims to make the image more meaningful through the ability to interact with it.

[0033] Host: An entity that hosts various resources.

[0034] M2M Service: M2M oriented capabilities that are provided to applications, which may be through Application Program Interfaces.

[0035] M2M Service Node: A network node hosting a service layer supporting one or more M2M Services for M2M communication.

[0036] Object disambiguation: The process of evaluating whether two PO representations are associated with the same real physical object, subject to policies and implementation choices. The disambiguation process may trigger other actions, e.g. merging of information, deleting one of the representations, etc.

[0037] Object recognition: The process of detecting an object and the decision to classify it as such. Object recognition may rely on matching, learning, or pattern recognition algorithms, e.g. based on vision (video or photo) or may be based on other logic (e.g. semantics). Both recognition and identification processes are often iterative, with the result depending on settings indicative of granularity, confidence level, etc. as well as the input data. The following is an example of recognition and identification processes based on the same video input: A first algorithm (recognition) determines that there are 3 Passive Objects within a scene, and then a second algorithm (identification) processes the information pertaining to each of the 3 Passive Objects and identifies: "cat", "dog", "table."

[0038] Object identification: Object identification refers to the process of cataloguing an object. Object identification may use matching, learning, or pattern recognition algorithms or other logic such as semantic inference, etc. This process may be similar to the Object recognition process, but Object identification goes further in classifying an object. To distinguish Object identification from the Object recognition process, the Object identification process is referred to herein as the process following Object recognition (and the decision that an object has been detected).

[0039] Originator: An entity that creates a request and transmits it to another entity (i.e. to a receiver).

[0040] Passive object (PO): In this context, any physical object (as defined herein) that is not defined by computational, telecommunication or data sharing capabilities. Moreover, passive objects may not have any computational, telecommunication or data sharing capabilities. Passive objects may be distinguished from the "Things" that are entities in the Internet of Things (IoT) domain at large (including e.g. WoT, M2M, etc.) as not being

computing devices capable of communicating, hosting software or data, or having any computational, telecommunication or data sharing capabilities, etc. Examples of such differentiations include humans versus their smartphones, fruit versus a Wi-Fi enabled fruit shipping container, etc.

[0041] Physical object: In this context, any real material object (e.g. human, other animate or lifeless object; static or mobile; palpable or intangible) characterized by its physical aspects.

[0042] PO-related service: Any service or application that understands/implements/uses the concept of PO and PO representations.

[0043] PO representation is used herein to refer to the digital counterparts of passive objects.

[0044] Receiver: An entity receiving a request from another entity (i.e. from an Originator).

[0045] Registrar: A service node where an application or another service node has registered.

[0046] Registree: An entity that registers to a service node (i.e. a Registrar) for its services.

[0047] RESTful: Systems, operations, etc. designed based on REST (Representational State Transfer) principles. Create, Retrieve, Update, Delete (CRUD) operations are an example of RESTful operations.

[0048] Virtual object: The digital counterpart of any physical object that has the role of bridging the gap between the physical and the virtual world. As noted above, PO representation is used herein to refer to the digital counterparts of passive objects.

[0049] Virtual Reality: technology that simulates the user's physical presence in a virtual or imaginary environment that may be completely computer generated and driven. The emphasis may be on visual simulation using specialized headsets, sometimes in combination with physical spaces or multi-projected environments and with audio input. Haptic simulation is also being researched and developed.

[0050] Virtualization of passive objects: There are two commonly encountered concepts using this terminology, which may be disambiguated as used herein as follows:

[0051] (1) The process of creating a digital counterpart, i.e. PO representation from information about the real object. More specifically, it refers to the process of creating a

digital counterpart which has sufficient information to be used by specialized services such as tracking, Augmented Reality, etc., which may be referred to herein using the terminology "creating an object (or PO) representation" or "creating a PO resource."

[0052] (2) The process of using the digital representation of a passive object to render it graphically (e.g. in VR or AR scenarios) together with a complementary environment, with the purpose of making the object appear to be part of the "real-life environment," which may be referred to herein using the terminology "visual rendering for virtualization of Objects" or simply "visual rendering."

[0053] Examples from M2M/IoT definitions of the entities available in the system are described in the following examples. While terminology such as "thing" may be widely employed, the example definitions below show that in many cases "thing" in IoT may stand for entities with computing and data sharing capabilities, and their use may rely upon their connectivity to other things on the internet. Below are example definitions of things as used in IoT:

[0054] (1) ITU-T Y.2060: with regards to the IoT, a "thing" is an object of the physical world (physical things) or the information world (virtual things), which is capable of being identified and integrated into communication networks.

[0055] (2) Techpublic: the "thing" commonly referred to by the concept of the IoT is any item that may contain an embedded, connected computing device. A "thing" in the IoT may be a shipping container with an RFID tag or a consumer's watch with a WIFI chip that sends fitness data or short messages to a server somewhere on the Internet.

[0056] (3) Micrium: the definition of a "thing" in the IoT varies a lot, but may be defined as an embedded computing device (or embedded system) that transmits and receives information over a network for the purpose of controlling another device or interacting with a user. A thing in this context may also be a microcontroller or microprocessor based device. The capabilities of these embedded devices have been expanding at the speed of Moore's law.

[0057] (4) Wikipedia: "Things," in the IoT, may refer to a wide variety of devices such as heart monitoring implants, biochip transponders on farm animals, electric clams in coastal waters, automobiles with built-in sensors, or field operation devices that assist fire-fighters in search and rescue. These devices may collect useful data with the help of various existing technologies and then autonomously flow the data between other devices.

[0058] (5) SWOT model: "Thing" in Web of Things (WoT) may be any object that has a unique identifier and that may send/receive web resources (including information and data) over a network. WoT devices may be connected to any device in the Web and provide information about themselves or about their surroundings (e.g. information sensed by the connected sensors) over a network (to other devices or servers/storage) or allow actuation upon the physical entities/environment around them remotely.

[0059] oneM2M: "Thing" may be an element that is individually identifiable in the oneM2M system.

[0060] Other sources define a "thing" in IoT as any object that has a unique identifier and which can send/receive data (including user data) over a network (e.g., smart phone, smart TV, computer, refrigerator, car, etc.).

[0061] Below are example definitions of devices as used in IoT:

[0062] (1) ITU-T Y.2060: a device may be a piece of equipment with the mandatory capabilities of communication and the optional capabilities of sensing, actuation, data capture, data storage and data processing.

[0063] oneM2M: Node may be defined as a logical entity that is identifiable in the oneM2M System. M2M device may be defined as physical equipment with communication capabilities, providing computing and/or sensing and/or actuation services. An M2M device may host one or more M2M Applications or other applications and may contain implementations of CSE functionalities. For example, an M2M Device contains a physical mapping associated with an Application Service Node or an Application Dedicated Node.

[0064] (2) Other sources define device as being connected to the Internet and sending information about themselves or about their surroundings (e.g. information sensed by the connected sensors) over a network (to other devices or servers/storage) or allow actuation upon the physical entities/environment around them remotely. Examples of IoT devices include but are not limited to: a home automation device that allows remotely monitoring the status of appliances and controlling appliances; an industrial machine that sends information about its operation and health monitoring data to a server; a car that sends information about its location to a cloud-based service; and a wireless-enabled wearable device that measures data about a person such as the number of steps walked and sends the data to a cloud-based service.

[0065] The following is a list of acronyms relating to technologies that may be used in the examples described herein:

ACP	Access Control Policy
AE	Application Entity
AR	Augmented Reality
App	Application
CSE	Common Service Entity
CSF	Common Service Function
CRUD	Create, Retrieve, Update, Delete
EAN	European Article Numbering association
EPC	Electronic Product Code
GDSN	Global Data Synchronisation Network
IoT	Internet of Things
IP	Internet Protocol
M2M	Machine-to-Machine
OCF	Open Connectivity Foundation
PO	Passive Object
QR	Quick Response (code)
REST	Representational State Transfer
RFID	Radio Frequency Identification
RPC	Remote Procedure Call
UCC	Uniform Code Council
UPC	Universal Product Code
URI	Uniform Resource Identifier
VR	Virtual Reality
WoT	Web of Things

[0066] Fig. 1 is a diagram of an example protocol stack 100 supporting an M2M/IoT service layer. From a protocol stack perspective, middleware service layers are typically layered on top of existing network protocol stacks and provide value added services to client applications as well as other services. Hence, service layers are often categorized as "middleware" service layers. For example, Fig. 1 shows a service layer 102 located in between an IP networking stack and applications. As shown in the example of Fig. 1, this protocol stack 100 may include an applications layer 101, application protocols layer 103 (e.g. HTTP or OCAP), transport protocols layer 104 (e.g. TCP or UDP), network protocols layer 105 (e.g. IPv4 or IPv6) and access network protocols layer 106 (e.g. Ethernet, Cellular, Wi-Fi) in addition to the service layer 102. Service layer 102 instances may be deployed on various network nodes (gateways and servers) and may provide value-added services to network applications, device applications, and to the network nodes themselves.

[0067] An M2M/IoT service layer is an example of one type of middleware service layer 102 specifically targeted towards providing value-added services for M2M/IoT type

devices and applications. Several industry standards bodies (e.g. oneM2M, OCF, ETSI M2M, and OMA LWM2M) have been developing M2M/IoT service layers to address the challenges associated with the integration of M2M/IoT types of devices and applications into deployments such as the Internet/Web and cellular, enterprise, and home networks.

[0068] An M2M service layer may provide applications and devices access to a collection of M2M centric capabilities supported by the service layer, which may include but are not limited to the following examples: security, charging, data management, device management, discovery, provisioning, and connectivity management. These capabilities may be made available to applications via application programming interfaces (APIs) that make use of message formats, resource structures, and resource representations that are supported by the M2M service layer.

[0069] Fig. 2 is a diagram of an example oneM2M Service Layer that supports an initial set of common service functions (CSF) 200. The purpose and goal of oneM2M is to develop technical specifications that address the need for a common M2M service layer that may be readily embedded within various hardware and software platforms and that may be relied upon to connect a wide variety of devices in the field with M2M application servers worldwide.

[0070] As shown in the example of Fig. 2, the oneM2M service layer may support a set of CSFs (i.e., service capabilities). An instantiation of a set of one or more particular types of CSFs may be referred to as a Common Services Entity (CSE) 201 (i.e., service layer) that may be hosted on different types of network nodes (e.g. infrastructure node, middle node, application-specific node). Common functions exposed by CSE 201 may include but are not limited to addressing and identification 210, data management & repository 211, location 212, security 213, communication management/delivery handling 214, registration 215, group management 216, device management 217, subscription notification 218, service charging & accounting 219, discovery 220, and network service exposure/service ex & triggering 221.

[0071] These common functions may be exposed via the Mca reference point 204, Mcc reference point 205, and Mcn reference point 206. The Mca reference point 204 may designate communication flows between an Application Entity (AE) 202 and the CSE 201, while the Mcc reference point 205 may designate communication flows between two CSEs 201 in the same M2M Service Provider domain. Communications across the Mca reference

point 204 and Mcc reference point 205 may take place via paired Request/Response messages, wherein each request may perform a specific RESTful operation (e.g., CRUD) upon a resource hosted on the targeted CSE 201. The Mcc reference point 205 may be used between CSEs 201 located in the Infrastructure Domain of different M2M SPs. The Mcn reference point 206 may be used between the CSE 201 and an underlying Network Services Entity (NSE) 203 for services other than transport and connectivity. A particular CSE 201 implementation may not support every function shown in the example of Fig. 2, but a complete implementation would include all the functions shown in Fig. 2.

[0072] Per the oneM2M RESTful architecture, CSFs may be represented as a set of resources. A resource may be a uniquely addressable entity in the architecture having a representation that can be manipulated via RESTful methods such as CRUD. These resources may be made addressable using Universal Resource Identifiers (URIs). A resource may contain child resources and attributes. A child resource may be a resource that has a containment relationship with a parent resource. The parent resource representation may contain references to its child resources. The lifetime of a child resource may be limited by the parent's resource lifetime. Each resource may support a set of "attributes" that store information about the resource.

[0073] Impacto is an example of a device designed to render the haptic sensation of hitting and being hit in virtual reality and an example of the research done on simulating interactions in the physical world. The device stimulates hits or punches by decomposing the stimulus: it renders the tactile aspect of being hit by tapping the skin using a solenoid; it adds impulse to the hit by thrusting the user's arm backwards using electrical muscle stimulation. The device is self-contained, wireless, and small enough for wearable use, and thus leaves the user unencumbered and able to walk around freely in a virtual environment and enhancing the experience of kicking a virtual soccer ball.

[0074] There are several initiatives for providing global identification standards such as GS1, EPC and others. This work has been accelerated by developments in fields such as barcodes, QR, etc. and aim to provide globally unique identifiers. oneM2M recognizes these developments by addressing Heterogeneous Identification services in a corresponding Work Item. To support applications from different providers or the third-party entities, this work aims to make the information related to the physical entities and the M2M service platform compatible with the various identification systems.

[0075] GS1 standards create a common foundation for business by uniquely identifying, accurately capturing, and automatically sharing vital information about products, locations, assets and more. Businesses may also combine different GS1 standards to streamline business processes such as traceability. Currently GS1 standards are focused on identification, capture and sharing.

[0076] GS1 identification standards include standards that define unique identification codes (called GS1 identification keys), which may be used by an information system to refer unambiguously to real-world entities such as: trade items, logistics units, physical location, documents, service relationships, etc.

[0077] GS1 data capture standards include definitions of bar code and radio-frequency identification (RFID) data carriers, which allow GS1 Identification Keys and supplementary data to be affixed directly to a physical object, and standards that specify consistent interfaces to readers, printers, and other hardware and software components that connect the data carriers to business applications.

[0078] GS1 standards for information sharing include data standards for master data, business transaction data, and physical event data, as well as communication standards for sharing this data between applications and trading partners. Other information sharing standards include discovery standards that help locate where relevant data resides across a supply chain and trust standards that help establish the conditions for sharing data with adequate security.

[0079] As shown through the various definitions for “thing” and “device” noted above, objects not defined by (e.g. without) computational, telecommunications, and data sharing capabilities (termed here passive objects (POs)) are not entities of IoT systems. Rather POs are recognized only from the perspective of their relationship to other devices and are employed to characterize or augment the entities of the platform, i.e. the other devices.

[0080] Recognition of objects without computing and data sharing capabilities has been explored by Augmented or Virtual Reality (AR/VR) applications. AR systems may merge views of the physical world with computer-generated graphics or other types of media, but sometimes the latter may need to be based on physical objects that are not part of the working physical world view. For accurate alignment, physical world knowledge, such as spatial relationships between objects or between objects and sensors may be required. In many real-world situations producing results with enough accuracy may require large data

sets and enough computational power to process the information in a short time to support meaningful feedback. This may be difficult to achieve when physical world recognition, tracking, event generation, and event use are encapsulated within the same application. Special arbitration techniques may be required for physical world representations to be shared between AR/VR applications in order to accurately represent physical world interactions.

[0081] The IoT/M2M domain is evolving to include computationally heterogeneous environments with wild extremes between devices. On one side, there are computationally rich devices, with capabilities such as advanced video processing leading to the fast recognition of objects or people, on the other are simple devices constantly delivering information difficult to obtain otherwise. Being able to share PO information between such devices would provide an important value-add for this domain.

[0082] For example, haptic devices may allow users to feel simulations of mechanical impacts, for example the hit of a soccer ball in the Impacto system. In this example, currently two or more users cannot share the same virtual soccer ball unless the system is specifically re-designed for several players. Assuming such personal devices would become widely available, it would be of great benefit if users with two different devices could play a game together because both devices simply recognize a common virtual soccer ball.

[0083] A similar case occurs even when a field player and a goalkeeper want to train together, but they are at different locations. The goalkeeper may be equipped with haptic devices such as Impacto, so that she is able to receive feedback on the arms and legs when her body is on the path of a virtual ball. Meanwhile, the field player uses a real ball that is aimed at a wall simulating the goal area. In this case the issue is not that of interworking of multiple devices using a common PO representation. Rather, the field player needs to be able to input her real soccer ball into the system, such that a tracking application monitoring her kicks transforms the information about the real object into a virtual one, which may be used by the haptic application used by the goalkeeper. Currently, this use case requires the two players to use a single/integrated system, which may recognize the soccer ball a-priori.

[0084] In another example, it would be useful to exchange PO information (e.g. the representation of construction materials) between applications such as smart cameras with object recognition overseeing an industrial floor, robots active in the space that use the material when in reach, warehouse inventories, and transportation logs. There is a need to

find out what services related to physical aspects may be provided by the platform, e.g. what application might be used to measure the material.

[0085] Recognizing these issues, more and more AR/VR applications are turning towards becoming platforms, enabling ecosystems for other applications to plug-in. However, without a broader service layer solution, interoperability is limited and cannot be taken advantage of in the IoT realm.

[0086] Being able to share PO representations between applications and devices may enable services designed around POs as well as to augment those designed around devices, in part by enhancing the meaning, usability, and functionality provided by some existing IoT/M2M system concepts. For example, the concept of a "device" (i.e. thing, node) when seen as a PO may allow the opportunity for new services to be designed around their physical characteristics (non-computing, non-communicating) and augment the services based on their smart/functional capabilities. Similarly, the concept of "user" may be enhanced by addressing the physical aspect of a person.

[0087] Moreover, without the capability to recognize physical objects as entities within an IoT system, some systems have introduced assumptions and approximations which have become more apparent as the applications have become more sophisticated. For example, what is recognized as "John's presence" in a certain context may really be described as "the presence of John's smart phone, with connectivity available;" the grocery item "soup can" may be "barcode sticker of a soup can."

[0088] For those items that may benefit from services such as tracking, semantic annotation, inventory, and management a service layer method of recognizing them as entities in the system may also provide the benefit of interoperability between applications.

[0089] A Passive Object Management Service (POMS), which an IoT/M2M service layer may provide to its registrants, is described in the examples and embodiments herein. The POMS may allow service layer platforms to provide services related to POs. The POMS may receive and collect information about POs, register them using the attributes/information provided/received, and add them to local or external databases for the purpose of management and data sharing. The POMS may generate a unique passive object token (*poToken*) to identify and authenticate the PO and to arbitrate data-sharing and service requests. The *poToken* may comprise identifiers and PO attributes necessary to perform functionality

including but not limited to: recognition and identification, registration, indexing, inventory, control, and management of the passive object.

[0090] The POMS may enable the execution of specialized PO-related services on PO representations shared among multiple service layer applications and entities by using policies including but not limited to: recognition, identification, location, correlation, control, and management policies, etc. Policies may enable the POMS to determine how to recognize a PO and what to do once a PO has been recognized, i.e. how to process and identify the PO.

[0091] The services may maintain correlations between POs and mappings to external identifiers, may maintain associations and mappings between different PO representations, and may link to associated services. The services may be able to discover and propose new services to be associated with and provided to the POs.

[0092] Methods to enable applications that may provide functionality such as object recognition, identification, registration, tracking, semantic annotation, inventory, and management to interwork. This framework may allow different applications and entities to exchange PO-related information, interpret it, and use it in a cooperative and harmonious manner.

[0093] Figs. 3 to 15 (described hereinafter) illustrate various embodiments associated with the POMS. In these figures, various steps or operations are shown being performed by one or more nodes, apparatuses, devices, servers, functions, or networks. For example, the apparatuses may operate singly or in combination with each other to effect the methods described herein. As used herein, the terms apparatus, network apparatus, node, server, device, entity, network function, and network node may be used interchangeably. It is understood that the nodes, devices, servers, functions, or networks illustrated in these figures may represent logical entities in a communication network and may be implemented in the form of software (e.g., computer-executable instructions) stored in a memory of, and executing on a processor of, a node of such network, which may comprise one of the general architectures illustrated in Figs. 16A or 16B described below. That is, the methods illustrated in Figs. 3 to 15 may be implemented in the form of software (e.g., computer-executable instructions) stored in a memory of a network node, such as for example the node or computer system illustrated in Figs. 16C or 16D, which computer executable instructions, when executed by a processor of the node, perform the steps illustrated in the figures. It is also understood that any transmitting and receiving steps illustrated in these figures may be

performed by communication circuitry (e.g., circuitry 34 or 97 of Figs. 16C and 16D, respectively) of the node under control of the processor of the node and the computer-executable instructions (e.g., software) that it executes. It is further understood that the nodes, devices, and functions described herein may be implemented as virtualized network functions.

[0094] The POMS described in the examples and embodiments herein may be implemented as a standalone service or as a sub-service of an IoT Service Layer. The POMS service may be hosted on various types of M2M service layer nodes such as IoT servers, gateways, and devices. The POMS may manage the resources corresponding to PO resources hosted on that node or its registrars.

[0095] POMS functionality maintains PO-specific information, metadata, and policies using a PO representation. Services or applications that understand/implement/use the concept of PO and PO representations are termed "PO-related services." Services for creating and managing PO representations, which are fundamental to the integration POs into the system, are logically distinguished from auxiliary PO-related services, which use the PO representations to offer additional functionality.

[0096] Fig. 3 is diagram of an example system 300 providing a high-level perspective of the relationship between the POMS and PO-related services (management related or auxiliary) in the context of a service layer. The example system 300 of Fig. 3 reflects the fact that different implementations may choose to implement some functions (e.g. identification) within the POMS, on the service layer platform but external to POMS, or external to the service layer platform. Referring to Fig. 3, the example system 300 includes the M2M/IoT Service Layer & POMS 301, external applications 302, services for PO representation creation and management 306, PO-related auxiliary services 307, and POs screw1 303a, chair1 304a, and Tom 305a, and their representations in the system 303b, 304b, and 305b, respectively.

[0097] POs are represented in the service layer via one or more resources and/or sub-resources. The example system 300 of Fig. 3 also provides an overview of the PO representation 308 and its components, the poDescription 309, assocEntities 310, assocPolicies 311, poToken 312, and auxServRequest List 313.

[0098] The PO description (poDescription 309) is the resource containing PO identification elements and higher-level or descriptors. It is meant to be used for the

identification of the PO and provide the least granular information about it. As such, it also references other resources that provide additional information about the PO, such as the two informational blocks below.

[0099] Information about other system entities that may be related or associated with the PO (assocEntities 310) may be maintained by the service layer in order to provide meaningful services. This may include an owner, or auxiliary PO-related platform applications known to provide specialized services such as tracking for the given PO. Associated entities may also be external databases containing PO related information. Information about PO associated policies (assocPolicies 311) as well as policies associated with the applications/service layer providing the service may enable providing PO-related services in the service layer. Each PO may be provided with specific rules indicating, for example, that only applications with photo capabilities are allowed to be associated, not those with video.

[0100] In addition to the information above, the PO Representation in the service layer may be augmented several enablers such as poToken 312 and auxServRequestList 313. Given that the POs representation describes actual physical objects, some services closely linked to the physical realm require special arbitration that may be enabled by service arbitration information related to each PO (poToken 312). Consider the example of a bicycle that may be managed (e.g. rented, tracked) through two different applications which do not interwork. The service layer manages the bicycle as a PO and is able to arbitrate (i.e. select or choose) which application has control at each moment using the service arbitration information included in the *poToken*.

[0101] In order to be able to process PO-related service requests, the service layer provides methods for applications to request these services, such as auxiliary service request representations (auxServRequestList 313). The requests may be issued in a variety of ways such as via CRUD operations on resources in a RESTful environment or Remote Procedure Calls (RPCs). For this purpose representations of the auxiliary service requests may be implemented stand-alone or within the PO representation.

[0102] The following tables include attributes describing a PO and are grouped in blocks in one logical manner/embodiment as depicted in Fig. 3, but other groupings (and resource structure respectively) may be envisioned. Some attributes may be implemented by using more than one parameter. Table 1 shows the PO representation with elements

corresponding to each of these blocks. Storage for each of these elements relative to each other is implementation dependent.

Element	Description
poDescription	Link to a sub-resource or information block for entities associated with this PO
assocEntities	Link to a sub-resource or information block for entities associated with this PO
assocPolicies	Link to a sub-resource or information block for policies associated with this PO
poToken	Attribute used by applications for retrieving a *poToken* with service permissions for the associated PO
auxServRequestList	Resources for auxiliary PO-related service requests (*auxServRequest*) applicable to this PO. Auxiliary PO-related services may, for example include confirming, augmenting, or tracking the PO.

Table 1 PO representation

[0103] Each PO is described by several common parameters of the poDescription and identified by a unique system ID (poID). A PO may also be characterized by identifiers used in other systems or be associated with other POs within the same system. poID and other parameters are described in the table below.

Element	Description
poID	System-wide unique ID identifying the PO in the Service Layer. The element might include information about the system in which this identifier is applicable, as well as ID type, e.g. temporary vs. persistent. Information about linking to other temp-poIDs may be provided.
externalIDs	Other IDs identifying the PO in external systems or databases. Each element might include information about the system in which the respective identifier is applicable. For example, GTIN (Global Trade Item Number), SSCC (Serial Shipping Container Code), etc.
assocPoIDsList	List of other POs associated with this PO, e.g. POs which are candidates for disambiguation or component POs, etc. The relationship between POs in this list and the described PO is further detailed in the assocPoIDsContextList. For example, the list may contain two POs identified as components, one as disambiguation candidate (to be further compared as it may represent the same real physical object) and two disambiguation confirmations (already confirmed that they represent similar but distinct real physical objects). This list might be split into separate lists based on relationship (as exemplified above) or be provided within assocEntities.

assocPoIDsContextList	Context information about how other POs associated with this PO (from assocPoIDsList) relate with the described PO. For example, PO in assocPoIDsList may be exact matches of the same PO, parts of the same PO, templates, etc.
poDefType	Parameter indicating the type of PO described. The type is a high-level characterization of the PO based on the results of recognition and identification. For example, in some implementation a "tempPO" designation might be used for all POs for which recognition and identification levels fall under a threshold (see *poRecognitionLevel* and *poIdentificationLevel* for information about the respective algorithm status). A "preDefinedPO" designation might be used for all POs created based on pre-provisioned information. A "POrepresentation" designation might be used for POs which have been well recognized and identified. The poDefType may be used for example by tracking applications which subscribe for notifications at creation of PO representations, limited to fully identified POs only. It may also be used by disambiguation or other services that share "tempPOs" needing a common, platform-wide definition of which POs are pre-defined, which are fully identified, etc. Parameters such as *poRecognitionLevel* and *poIdentificationLevel* can provide further context and granularity.
poRecognitionLevel	High-level description of the level of PO definition or recognition, such as: pre-defined PO, fully recognized PO, recognition-pending-PO, unrecognized-PO, recognition-not-allowed-PO, recognition-not-enabled etc. These qualifiers reflect PO status based on the object being pre-defined or the recognition algorithm outcomes (respective meaning for the terms above: "recognition algorithm finalized successfully", "recognition algorithm returned partial results and running further", "recognition algorithm finalized unsuccessfully", "use of recognition algorithms not allowed", "recognition algorithm not enabled", etc. The poDefType may be used for example by tracking applications which subscribe for notifications at creation of PO representations, limited to fully identified POs only. It may also be used by disambiguation or other services that share "tempPOs" needing a common, platform-wide definition of which POs are pre-defined, which are fully identified, etc. Parameters such as *poRecognitionLevel* and *poIdentificationLevel* can provide further context and granularity.

poIdentificationLevel	High-level description of the identification for the PO might include categories such as: pre-defined PO, identified PO, provisionally identified PO, unidentified PO, identification-not-allowed-PO etc. Further granularity might be defined via complex or additional parameters as qualifiers, e.g. "provisionally identified high-confidence" vs. "provisionally identified low-confidence"
poRegistrationPurpose	Registration purpose provides a high-level description of the types of services which are proposed to be provided by the platform, e.g. platform tracking. Although tracking services can also be specifically requested, this enables the platform to propose suitable applications in the absence of explicit calls.
poSpecificDataModel	Indicates if any specialized data model is used for this PO. For example, in some industrial systems data models for specific products may be used.
poIdentificationInfo	High level identification such as person/animate/lifeless, or might have greater granularity, e.g. in an industrial system might distinguish end product/ raw material/ component. This may also contain a list of parameters describing the object, or links to where PO identification information is contained. Note that similar information, e.g. container with raw visual data and a semantic description resource) might be linked to the PO via the *poResourceList*.
poSemanticDescriptor	This parameter contains a semantic description of the PO, e.g. RDF description. It may include other semantics-related information about the PO, e.g. associated ontology
poTokenReq	Attribute used by applications for retrieving a *poToken* with service permissions for the associated PO. See [0111] and **Error! Reference source not found.** for more information on *poToken*. . The use of *poTokenReq* for obtaining service permissions is an alternative to using *auxServRequest*.

Table 2 poDescription

[0104] Each PO may be associated with other entities in the system (*assocEntities*). For example, one or more owners may be specified, similarly to ownership of smart nodes. For objects possessing universal identifiers such as GTIN, the databases containing related information are associated entities. Similarly, other resources in the system containing related information may be associated, as well as system applications known to provide specialized services. The following table exemplifies information regarding the entities associated with the given PO (*assocEntities*).

Element	Description
ownerList	Identifies the owners of the PO and may be used in conjunction with other mechanisms to provide access control for PO related resources.
databaseList	List of databases containing external information about the PO. For example, for GS1: https://www.gs1us.org/, http://gepir.gs1.org/ The databases may be local. The databases may contain information such as templates or full objects descriptions to be used for PO recognition, therefore the list may contain indications of each database type as well.
tgServList	List of services which may provide PO token generation for this PO. There should be at least one tgServ which initially registered this PO. For more information on PO token generation and tgServ.
scoutAppList	List of applications which may provide scouting services for this PO. Scouting uses the processes of recognition and identification of POs in order to enable their integration in the SL via a registration procedure. In some implementations, a list of potential scoutApps may also be maintained (see also appAssociationPolicy).
auxServiceList	List of auxiliary services for the PO, e.g. semantic services, location services, tracking services, visual rendering services, etc. Each service type may be listed separately or the entries may be complex parameters indicating both the application and its service type. In some implementations, this list may contain only categories of applications, e.g. scaling and measurement services.
poResourceList	List of resources associated with this PO and providing the description of the PO, e.g. its measurements (for a table), containers with media files (table picture), etc.

Table 3 assocEntities

[0105] Each PO may be associated with policies (*assocPolicies)* provided to inform PO handling, as described in the following table. POs may be provided with only a subset of these policies. The handling of the PO may be determined by the explicitly provided policies enumerated here, as well as by the service policies provided to the service layer or applications using the PO representation. Those service policies are provided per service layer or application entity, but they may have in scope an entire group of entities, a domain, etc.

Element	Description
recognitionPolicy	Describes rules for PO recognition, e.g. if the PO is allowed to be recognized at all, specific types of recognition allowed (e.g. semantic matching vs. video processing), if specific entities or users are not allowed to initiate recognition, etc.

correlationPolicy	Describes rules for applying correlations between PO information in order to determine relationships between POs in the process of identification. For example, only POs registered by the same tgServ may be allowed to be used for correlation purposes, to identify duplicates or parts.
locationPolicy	Describes rules for determining and tracking location of the PO. For example, only POs registered by the same tgServ may be allowed to have location information provided.
visualRenderingPolicy	Describes rules for the visual rendering of the PO. For example, a visual rendering policy for people might allow only superimposition of photos over other editable pictures or real-time video, and it might further limit this type of operation to people in a contact list. Another policy might limit visual rendering only to pre-defined POs with 3D information.
auxServPolicy	Describes rules for how auxiliary PO-related service applications may be enabled. For example, the policy may allow POMS to create representations of such services (*auxServRequest*) for the PO or it may allow only explicit creation by another entity. The rules may be dependent on service type. This policy may be used in conjunction with access control and *poToken* to arbitrate PO-related services.
appAssociationPolicy	Describes rules on how to associate tgServs and scoutApps with the PO. For example, only scoutApps with photo capabilities are allowed to be associated, not those with video. Another type of rule may concern how the applications may be associated, i.e. they can be proposed by the POMS to be associated or only based on their own registration.
discoveryPolicy	Describes rules for the discovery of this PO. These rules may be used in conjunction with access control mechanisms to determine which applications may be allowed to discover the PO.
poTokenPolicy	Describes rules for the generation of the poToken which is to be used by the POMS to provide access control and arbitration of services provided to the PO. Describes also rules for the arbitration of service requests based on the poToken. This also includes rules regarding which applications should be provided with the *poToken* when requesting it.

Table 4 assocPolicies

[0106] The POMS may rely upon a number of individual services or functions for creating and managing the PO representation in the service layer. A PO representation may be created in the service layer by performing a PO registration. Once a PO is registered the corresponding PO representation is available to the service layer or to applications. In order to be able to initiate a PO registration a physical object needs to be "scouted." PO scouting is

the term used herein to include the processes of recognition and identification of POs, which are individually described herein.

[0107] The PO recognition process consists of detecting a PO and results classifying the PO in a category of POs or as a particular type of PO. The PO recognition process may use matching, learning, or pattern recognition algorithms or other logic such as semantic inference, etc. PO recognition may be implemented based on algorithms executed by a device running a scouting application, such as a smart camera or computing device. The recognition algorithms may be iterative and may depend or certain settings e.g. granularity, confidence level.

[0108] PO identification is the process following recognition, through which a detected object is subject to further classification. Similarly to recognition, the identification functionality may use matching, learning, or pattern recognition algorithms or other logic such as semantic inference, etc. In these descriptions PO recognition and identification may be implemented jointly by a platform application or service, or they may be implemented separately and used jointly. At the end of the identification process the PO representation may be given a unique ID, which may be the *poID* described above or may be correlated with the *poID* during the registration process. PO identification may be implemented based on algorithms executed by a device running a scouting application, such as a smart camera or computing device. The identification algorithms may also be iterative and may depend or certain settings e.g. granularity, confidence level.

[0109] For example, a smart camera may perform a recognition process to recognize a PO entering a room as a person. The smart camera may then perform an identification process to identify the person as a specific individual by name.

[0110] As noted above, PO scouting is the term used herein to include both recognition and identification. Given the reliance of recognition and identification algorithms on similar input information, the following descriptions assume that the two algorithms are implemented jointly, but this they may also be implemented separately. Both algorithms may be iterative, so there might be a phase where the joint algorithm generates a "candidate PO" with a classification given based on a lower confidence level. Later, further information or algorithm iterations might result in a higher confidence level and possibly a different classification.

[0111] A function or application implementing PO scouting may be referred to herein as a scoutApp, to be distinguished from applications that simply provide input. For example, a "smart" camera that processes its own feed and is able to provide triggers when objects are recognized and identified may be considered a scoutApp (the input functionality is implied). A "regular" camera that provides only video feeds is an input application to a platform. A gateway receiving feeds from several regular cameras (input applications) uses a local or cloud-based application as a scoutApp in order to provide PO-related services.

[0112] The *poToken* may contain information used to arbitrate service requests affecting the PO. The *poToken* is unique to the PO and associated with physical aspects of the PO. For example, a PO such as a soccer ball may only be in one place at one time so an arbitration process of the *poToken* of the soccer ball would ensure that only one application at a time executes operations using the *poToken* of the soccer ball.

[0113] The *poToken* is linked to the corresponding PO representation and the *poDescription* information via the poID. Given that POs are representations of actual physical objects, the *poToken* may be used to provide arbitration between services closely linked to the physical realm. For example, the platform may be used to implement a system through which a robot may be controlled through two different applications on the same service layer that do not interwork. The service layer manages the robot as a PO and is able to arbitrate which application has control at each moment using the *poToken.* The *poToken* parameters are shown in the table below.

Element	**Description**
poID	Unique ID identifying the PO in the system. The element might include information about the system in which this identifier is applicable, as well as ID type.
issuerID	ID of the poToken issuer
timestamp	Time of the poToken issuance
servicePermissionIDs	Identifiers used for controlling the PO access by services for the associated PO. The identifiers may be provided per service or service type, e.g. scouting permission ID.
servicePermissionContext	The parameter indicates context for the use of the servicePermissionIDs, e.g. service type allowed, if more than one entity may be performing the service simultaneously, time window or number of operations limits, etc.
servicePermissionAudience	Optional list of entities allowed to request the *poToken*. See clause 5.1.4 for description of PO-related Service Requests using the *poToken*.

refID	A reference identifier to be used to identify the *poToken*. Depending on PO type and PO-related services needed, more than one *poToken* may be associated with a poID. This reference identifier may be exchanged when requesting service permission.

Table 5 *poToken*

[0114] *poToken* generation is a service enabling specialized PO management by the POMS, which is termed tgServ herein. *poToken* generation services may be implemented by an application (e.g. scoutApp) or be included in the POMS functionality. Generating the *poToken* requires the capability to generate poIDs unique within the system (either via assignment or negotiation). It also requires availability of the *poTokenPolicy* or implementation of similar rules for how the service permissions included in the *poToken* should be generated. Once the *poToken* is generated, POMS is able to provide arbitration of services which may access or use the PO representation, if needed.

[0115] The service layer relies upon other individual services to provide PO-related functionality, in addition to scouting and *poToken* generation needed for management. These services may be used to confirm, augment, or track the PO and may be implemented in either the service layer or as application layer functionality interacting through a service layer. These service may be considered individual, atomic functions external to the POMS for ease of description, but some may be employed differently, e.g. merged or implemented jointly, included in the POMS, etc. High-level, exemplary descriptions of such services are provided below.

[0116] POMS relies upon external services for deciding if two POs represent the same physical object or for confirming PO information. This process may use the PO information managed by the service layer and the POMS. PO disambiguation may include execution of algorithms to resolve conflicting information about POs between service layer data and external database information. The result of the disambiguation and confirmation processes is that POs in a system may be used as separate individual entries.

[0117] The purpose of augmentation is to provide additional information about permanent characteristics of the PO. For example, the information about a bicycle identified in a night-time photo might be augmented with color after a day-time picture is provided or after special processing of the night-time picture. The PO augmentation process may employ inputs and algorithms similar to those available for recognition and identification, or may be different in nature.

[0118] The purpose of the PO tracking process is similar to that of augmentation, in that it aims to provide additional information about the PO. However, tracking refers mainly to transitory or semi-permanent characteristics of an object, e.g. location. In the examples and embodiments described herein, these processes are treated together as the differentiation between PO characteristics depends greatly on the system implementation. For example, in a factory the color of an object might be tracked as it goes through painting procedures, rather than being considered a single update or augmentation procedure. Algorithms used for augmentation and tracking may also be based on semantics, information discovery, location services, etc.

[0119] In order to be able to process auxiliary PO-related service requests (*auxServRequest*), the service layer provides methods for applications to request these services. The resources in the following table may be used for these requests and may be implemented as a stand-alone resource or as a sub-resource of a PO resource.

Element	**Description**
requestingEntityID	ID identifying the requesting entity in the system
poID	poID or a list of poIDs for which the request applies. If this *auxServRequest is* a sub-resource of a *poDescription,* then the request applies specifically to that PO
requestType	Identifies the request type, which is further described by the requestParamList. The types allowed depend on implementation and may include key terms such as reserve, augmentInfo, updateInfo, trackParams, customOp, etc. A special type of request may be implemented by using a special value e.g. *requestTokenInfo* which signifies that the initiator requests the *poToken* information for subsequent operations.
requestParamList	List of complex attributes containing parameters related to the service request. For example, an *updateInfo* request might indicate *poIdentificationLevel* as a parameter which is requested to be augmented. For certain types of requests (e.g. customOp) the parameter may contain the full content of the operation to be executed. When requesting the *poToken* information (*requestType* = *requestTokenInfo*) these attributes may indicate the purpose of the request, number of operations or amount of time needed, etc.
requestContextList	List of complex attributes containing further context for the request, for example it may indicate "high" as the desired *poIdentificationLevel.*

servicePermissionID	This is a parameter corresponding to the *servicePermissionIDs* described in the *poToken* clause. It allows POMS to arbitrate PO service requests. When requesting the *poToken* information (*requestType = requestTokenInfo)* this parameter is not needed. It may also not be needed if no special arbitration is required for the auxiliary PO-related service calls.
customOpExec	Attribute used to trigger a request of type customOp or similar.
explicitAuxServRequest	Indicates if a specific auxiliary service (from auxServiceList) is to be invoked. If it is not specified, POMS selects a suitable platform service application to invoke, depending on the requestType.
serviceReqRespInfo	Attribute used to store information resulting from the execution of the auxiliary service, which may need to be provided to the initiator, in addition to the information stored in existing SL resources (e.g. specific return codes)

Table 6 auxServRequest

[0120] Procedures to enable POs to be integrated in the M2M/IoT Service Layer and to enable services to be executed using the POs SL representations are described herein. The M2M/IoT system may need to be aware of the POs related services available and the policies associated with these services. The physical devices may need to be represented as POs in the M2M/IoT system. Applications need to be able to request PO related services on these POs.

[0121] Existing Service Layer systems such as oneM2M include procedures, termed here registration, through which individual application entities provide information to the service layer about their capabilities and "sign-up" for services from the Service Layer. The application registration procedures described herein may be enhanced such that the applications provide information about their capabilities from a PO management perspective, e.g. their capability to provide scouting, assigning poIDs, etc. This information may be similar in scope with the capabilities provided at the entity by the policy, or may be finer-grained, e.g. functionality for PO identification with image-based template matching. Alternatively, this information may be provided to the service layer or other applications upon request or discovery, via announcement or notification procedures, etc. The below table includes parameters for enhanced application registration.

Element	**Description**
poRelatedCapabilities	Describes the PO-related functionalities the application can provide e.g. PO scouting, PO recognition only, etc.

poPolicies	Information about the PO related policies of the application, as described with respect to policy provisioning.

Table 7 Parameters for enhanced application registration

[0122] Service layer applications/entities and other applications need to be provided with policies regarding handling of PO functionality. The following table describes PO handling policies that may be pre-provisioned or configured for these entities. The description assumes policies to be provided per service layer or application entity, but they may have in scope an entire group of entities, a domain, etc. The table includes a large variety of policies, but not all need to be realized in a given implementation. Such services may also use local policies for implementing this functionality. Moreover, some policies apply only to specific services or procedures, e.g. scouting or PO registration. These policies may be provided to the service entity via pre-provisioning or configuration following the application registration process.

Element	**Description**
entityRole	Specifies the role of the entity for which this policy applies. Roles vis-à-vis PO handling functionality may include: - POMS: entity implementing PO management services - scoutApp: entity implementing recognition and identification services (Note: these two functions may be implemented separately, in which case scoutApp could denote recognition services only, and the identification role may be termed differently e.g. identificationApp) - tgServ: entity implementing poToken generation services - poDB: entity implementing a PO database, such as a PO Representation repository - input: entity implementing a functionality which may provide input to PO recognition algorithms, e.g. camera. - not usable: functionality may not be used for PO services - unknown: roles vis-à-vis PO handling unknown.
retentionRules	Includes rules about what PO-related information should be retained by this application or function.
scoutingTypeRules	Include rules about the type of scouting to be done by this scoutApp application. For example, the scouting may be self-initiated or on-demand only, or both approaches may be allowed.
poScopeRules	Include rules about the POs in scope of scouting. These may include: - rules about the types of objects which may be scouted (e.g. people/animals/objects, manufacturer X product only, etc.). - rules about POs in scope such as their location.
scoutingScope rules	Other rules about scouting scoping such as: - Time-based, e.g. time-of-day - Location based, e.g. only when the scouting occurs in a certain location

	- Operation control, e.g. only POs registered by a specific entity may be provided with information resulting from the scouting "operation" - User-based, e.g. specific application users may scout for specific POs
identificationRules	Include rules about the objects to be identified. These may include rules about the types of objects which may be identified (e.g. large or fine grained-categories such as people/animals/objects, manufacturer X product only, etc.). Identification refers to a process of assigning the PO to categories of various granularity, annotation of PO with labels or semantics which provide additional description, etc. Identification results are qualified by the identification scope and the identification certainty.
triggeringRules	Rules about the generation of triggers that a PO has been recognized and identified. (Here again recognition and identification may be separated, but the description assumes scouting incorporates both). The rules may include granularity or confidence levels which may be independent of which PO is analyzed. The rules may also specify if triggering of candidate POs (candidatePO) is to be performed, for the purpose of registering POs which may require further servicing by other applications. Conversely, triggering only when a PO processing has been fully finalized (according to policies) may be allowed.
IDrules	Includes rules about the POs to be ID-ed. These may include rules about the types of objects which may be assigned with temporary or persistent IDs. The rules include information about the ID scheme, ID authority, number of parallel IDs, ID-based databases, etc. as applicable. In a system with applications that can use more than one ID scheme, the rules may indicate also if this application should be the one providing any of the alternate IDs, and if provides mapping between IDs, etc. For example, a gateway may be allowed to only provide temporary IDs or IDs unique in a sub-domain, while a central server may be required to provide IDs unique in the entire domain. In some cases all the IDs used by a certain platform may be required to be issued a standardized GTIN numbers. In yet another case a platform may be required to use its own numbering system for the poID, while associating each PO with applicable GTIN (as *externalIDs*) via an external database.
registrationRules	Includes rules about triggering the process of PO registration. For example, only some categories of scouted POs such as structural elements (e.g. doors, window) may be registered to the gateway in a certain implementation. Others, such as appliances, may be registered only to the local service layer, while others such as people should not be registered at all, even if identified.

	Rules may also indicate what types of parameters to be used during the registration procedure, described later in this paper. Other rules may specify that registrations should be triggered only within a specific time frame, or at a specific location, etc.
disambiguationRules	Rules to be used in the disambiguation process. For example, specific percentage matching should be achieved for two POs to be considered the same. The rules may include specifications that certain aspects must be considered (e.g. PO location) or entirely discarded (e.g. PO color) Rules may also indicate what types of POs should be disambiguated (e.g. people only), when the process should be self-triggered and when on-demand only. Other rules may include processing context such as time (e.g. process at night only), valid inputs (e.g. which database to be used for matching, video inputs only, photos from a specific camera only)
trackingRules	Rules to be used for tracking. For example, only specific timeframes or areas should be considered in scope, or only specific PO types.

Table 8 PO-related policies for services

[0123] Fig. 4 is a diagram of an example procedure 400 for PO registration in accordance with one embodiment, which may be used in combination with any of the embodiments described herein. The PO registration procedure 400 may result in a corresponding PO resource being created at the POMS with an associated poToken. Therefore, in a RESTful system the PO registration request may be implemented using a CREATE request containing the necessary information for the PO resource. Depending on the implementation, entries may be generated for associated registries and databases.

[0124] In the example of Fig. 4, the scoutApp and tgServ 401 may perform scouting of a PO and generate a poToken 410. The scoutApp and tgServ 401 may then transmit a PO registration request message 411 including the poToken and other PO information to the service layer POMS 402. At registration time the service layer POMS 402 may generate 412 a PO resource corresponding to the information provided in PO representation clause to associate with the poToken, and a PO repository for storing the poToken and the other PO information may be generated in the repository 403. The poID may be used for linking the PO resource created with the associated poToken. The service layer POMS 402 may transmit a PO registration confirmation message 413 to the scoutApp and tgServ 401.

[0125] In the example of Fig. 4, the poToken is generated by the scoutApp and tgServ 401 and provided to the service layer POMS 402 in the PO registration request message 411. In other embodiments, the poToken may be generated by the POMS based on a request from the tgServ.

[0126] Fig. 5 is a diagram of an example procedure 500 for PO registration in accordance with another embodiment, which may be used in combination with any of the embodiments described herein. In this example, the registration request message is used for the purpose of requesting a poToken. Referring to Fig. 5, the scoutApp 501 may perform scouting of a PO 510. A PO registration request process 511 may then be initiated in which the scoutApp 501 may transmit a PO registration request message 512 to the service layer POMS and tgServ 502. A NULL poToken may be included in the PO registration request message 512 to signify a request for poToken generation without PO resource creation. The PO registration request message 512 may be implemented using other messages, e.g. a specific poToken generation request may be used. The service layer POMS and tgServ 502 may then generate a poToken 513 and then transmit a PO registration confirmation message 513 that includes the poToken to the scoutApp 501. The scoutApp 501 may then transmit a PO registration request message 515 including the poToken and other PO information to the service layer POMS and tgServ 502, which may then generate 516 a PO resource corresponding to the information provided in PO representation clause to associate with the poToken, and a PO repository for storing the poToken and the other PO information may be generated in the repository 503. The service layer POMS and tgServ 502 may transmit a PO registration confirmation message 517 to the scoutApp 501.

[0127] In other examples, no service permission IDs (and related information) may be generated, and only the poID is used in the registration message with the PO information necessary to create the PO resource. Therefore, the entity initiating the registration may have either the capability to generate the poID or would be able to request it from POMS using a procedure similar to the one in the example of Fig. 5.

[0128] Further, the PO registration request/confirm messages in the examples of Fig. 4 and Fig. 5 may be used also to provide for a deregistration process. When initiated externally to the POMS, the PO registration request may include a deregistration-request flag indicating that the PO representation should be made unavailable in the system, e.g. via deletion, removing external access rights, etc. Similarly, the POMS may initiate deregistration in special cases (e.g. when information has become corrupted or unavailable) by sending a PO registration confirmation message with a flag indicating deregistration and that the PO representation is no longer available.

[0129] Optimizations may also be provided by allowing multiple PO registrations, such as those in the examples of Fig. 4 and Fig. 5, by the POMS at one time.

[0130] Platforms providing support for the integration of POs may include provisioned information about pre-defined POs. A PO repository may be available in the service layer, but access to external databases may also be used for this purpose. For example, the registration message may be directed to the service layer repository or to external databases. When pre-provisioning the PO information, the *poTokenPolicy* may specify how an associated poToken should be generated.

[0131] Fig. 6 is a diagram of an example procedure 600 for pre-defined PO provisioning and registration in accordance with another embodiment, which may be used in combination with any of the embodiments described herein. In this example, when a tgServ discovers a pre-provisioned PO repository entry it can use an explicit registration request message to trigger the poToken generation, as well as the PO resource creation based on the PO information available in the repository. Referring to Fig. 6, a pre-defined PO provisioning and registration request process 610 may include the repository 603 generating a PO repository entry (poEntry) 611. ScoutApp 601 may discover the poEntry 612. scoutApp 601 may transmit a PO registration request message 613 to the service layer POMS and tgServ 602. A NULL poToken may be included in the PO registration request message 613. The service layer POMS and tgServ 602 may then create a PO resource, generate a poToken, and associate the poEntry 614. The service layer POMS and tgServ 602 may then transmit a PO registration confirmation message 615.

[0132] Fig. 7 is a diagram of an example procedure 700 for pre-defined PO provisioning and automatic PO registration in accordance with another embodiment, which may be used in combination with any of the embodiments described herein. In this example, the *poTokenPolicy* may specify that an associated poToken should be generated based on internal POMS events, e.g. after a notification about the entry or after discovering the new entry after periodic discovery requests. The POMS may perform poToken generation as well as PO resource creation based on the PO information available in the repository without the intervention of tgServ. Referring to Fig. 7, a pre-defined PO provisioning and registration request process 710 may include the repository 702 generating a PO repository entry (poEntry) 711. The service layer POMS and tgServ 701 may then receive a PO repository entry notification or other POMS event notification 712, and PO representation information

may be retrieved by the POMS 713. The service layer POMS and tgServ 701 may then create a PO resource, generate a poToken, and associate the poEntry 714.

[0133] Similar to the tgServ triggered registration, a simplified implementation with no service permission IDs may be envisioned. In this case the poID may be used in the registration entries, together with the PO information to create the entry. In this example, the application used to generate the entries may have either the capability to generate the poID or would be able to request poIDs from POMS.

[0134] Note that in the registration flows of Figs. 4-7, a first step includes the PO registration request or trigger. When an explicit request is made, such as in Figs. 4-6, the request may be implemented in RESTful systems such as oneM2M via a CREATE operation that includes the PO representation information described herein. In the example of Fig. 7, the trigger notification prompts the POMS to retrieve the information from the repository, which may be implemented via a RESTful RETRIEVE operation. Once the PO representation information is available at the POMS, the PO resource creation may occur. This means creating the PO representation as described herein based on the information provided in the request and based on the applicable policies. The *poToken* may then be either generated or, if already provided, stored. PO entries in local or external registries may then be created (as needed) or, if provided via pre-provisioning, may be associated with the newly created PO representation.

[0135] As illustrated in the examples and embodiments described herein, the POMS plays a PO administration role in that it may monitor all the applications registered to the platform and may propose them as auxiliary PO-related services for individual POs. For example, upon PO registration, the POMS may use the information in *poRegistrationPurpose* to populate *auxServiceList,* in conjunction with policies such as *appAssociationPolicy.* This process may also be enabled by the information provided by each application in their enhanced registration, via *poRelatedCapabilities* and *poPolicies.* For example, using poRegistrationPurpose of "platform tracking", the POMS may be allowed to provide additional cameras as an auxiliary service that may be used as alternatives as needed. The POMS may also create the associated *auxServRequest* representations that may be used by external applications to request services. However, policies such as *appAssociationPolicy* may indicate that the POMS is not allowed to proactively or automatically proceed with these

actions. This information may be explicitly provided by the PO registration initiator or by any other entities with the required permissions.

[0136] Fig. 8 is a diagram of an example procedure 800 for a PO service request in accordance with another embodiment, which may be used in combination with any of the embodiments described herein. PO-related service requests may originate with applications which are PO aware, which have registered with the platform, and which have become aware of POs registered with POMS. In some examples, requests are made that do not involve the auxiliary platform PO-related services. For example, an Augmented Reality (AR) application may want to retrieve photo data related to a specific PO in order to superimpose it over an image. This request may be fulfilled by a RETRIEVE of information (e.g. *poDescription*, specifically using *poResourceList* for the image container), which in the context of our RESTful examples, may proceed according to the procedure in the example of Fig. 8. Referring to Fig. 8, the PO and applications are registered 810, and a PO service request (e.g. CRUD operations other than an *auxServRequest*) may be received by the service layer POMS 802 from an application 804. The service layer POMS 802 may execute the CRUD operation 812 and may transmit a repository update as needed 813 to the repository 803. The service layer POMS 802 may then transmit a PO service response 814 to the application 804. As illustrated in the example of Fig. 8, the auxiliary service resource 801 is not used (even if it is linked to *poDescription)*, and the request is executed in the service layer similarly to other retrieve operations, and the service layer uses regular access control mechanisms to perform the request initiated.

[0137] Fig. 9 is a diagram of an example procedure 900 for an auxiliary PO-related service request preceded by a poToken request in accordance with another embodiment, which may be used in combination with any of the embodiments described herein. In the example of Fig. 9, the POMS may enable applications to request auxiliary PO-related services. Applications perform this request by targeting *auxServRequest.* In the example of Fig. 9, the initiating application 904 provides a *servicePermissionID*, and therefore it should have *poToken* information. The *poToken* request may be implemented using a PO service request message, with the *requestType* set to *poTokenInfo* and any additional information needed by POMS in *requestParamList* and *requestContextList.*

[0138] Referring to Fig. 9, the PO and applications are registered 910, and a PO service request message including a poToken request 911 may be received by the service

layer POMS 902 from an application 904. The service layer POMS 902 may evaluate the request based on the *poTokenPolicy* and determine whether the *poToken* may be transmitted 912. The POMS may also evaluates the request based on other PO parameters. For example *poTokenPolicy* may indicate that a bike rental application may be granted only when a custom attribute (e.g. *loaningAvailibility*) has a specific value (e.g. *available)*. In another example, the *poTokenPolicy* may indicate that only one request from any rental application may be processed at a time. If the service request is approved based on the evaluation, the service layer POMS 902 may then transmit a PO service response 913 that includes the *poToken* to the application 904. After receiving the *poToken,* the initiating application 904 has the *servicePermissionID* for further requests. A PO service request message including a request type 914 may then be received by the service layer POMS 902 from the application 904. The service layer POMS 902 may evaluate permissions and arbitrate 915 the service request message based on information contained in the *poToken* such as the *servicePermissionID.*

[0139] The processing by the service layer POMS 902 in step 915 exemplifies the arbitration process. Assuming multiple PO service requests messages are received or are outstanding for the same PO, the service layer POMS 902 may evaluate them based on information about the *poTokens* that have been issued for the PO, the received *servicePermissionID,* the type and parameters of the received request, policies, etc. If the policies do not allow for multiple *poTokens* to be issued or active at the same time, the service layer POMS 902 may indicate to one or more requesters that their *poTokens* are invalid, expired, etc. This may be accomplished using the PO service response 913 message, which provides a *poToken* with the additional information. If the policies do allow for multiple *poTokens* to be active, the service layer POMS 902 may check if multiple requests of a same type are allowed or not by using, e.g. *poTokenPolicy* or local policies. For example, a request from a bike rental application may be processed at the same time as a request for semantic augmentation of the PO representation of the bike. In another example, the policies may also allow multiple *poTokens* to be active, but priority should be given to requests from specific applications. For example, a request from the platform's own bike rental application may be prioritized and be completed before a request from an external bike rental application. If the bike is still available after the first application is processed, it may be rented via the second one, but from a POMS perspective both *poTokens* are valid simultaneously with a priority indicated by the policies.

[0140] The service layer POMS 902 may then store the request in the *auxServRequest* resource targeted by the service request and may translate the service request into auxiliary service 901 calls that are then executed 916. The service layer POMS 902 may transmit a repository update as needed 917 to the repository 903. The service layer POMS 902 may then transmit a PO service response 918 to the application 904, e.g. by updating *serviceReqRespInfo* or any other PO representation parameters changed by the execution of the auxiliary service.

[0141] In an example, the procedure depicted in Fig. 9 enables the AR application described above to obtain augmented information about a PO, such as location information and a photo which is to be superimposed over another image. The AR application may obtain this information by finding location services in *auxServiceList* and may request it by using the corresponding member of *auxServRequestList.* As shown in Fig. 9, the service layer POMS 902 may execute and/or call the auxiliary service 916 (in this case the location service) and may use any PO related information from the PO representation or stored locally, as needed. The type of information necessary in the request may be provided by *auxServRequest* parameters such as *requestParamList* and *requestContextList.* Alternatively or additionally, the auxiliary services such as the location service may be invoked by the service layer POMS 902 using custom operations specified by *customOpExec.* In yet another alternative, applications such as the AR application may trigger the execution explicitly by using *explicitAuxServRequest.*

[0142] Fig. 10 is a diagram of an example procedure 1000 for a *poToken* service using *poTokenReq* in accordance with another embodiment, which may be used in combination with any of the embodiments described herein. In the example of Fig. 10, the poToken request may be performed by targeting the *poTokenReq* attribute of the corresponding PO. Referring to Fig. 10, the PO and applications are registered 1010, and a PO service request message including a RETRIEVE poToken request 1011 may be received by the service layer POMS 1002 from an application 1004. The service layer POMS 1002 may evaluate the request based on the *poTokenPolicy* 1012. The service layer POMS 1002 may then transmit a PO service response 1013 that includes the *poToken* to the application 1004. Auxiliary PO-related Service requests 1001 may then be arbitrated using the *poToken.* In addition to regular access control performed by the service layer, the service layer POMS may check the service permissions based on the *servicePermissionIDs* provided, as well as

associated parameters of the *poToken* such as *servicePermissionContext.* For example, the service layer POMS may verify whether the service required is of the specified type and within the specified time window, etc. The procedure of Fig. 10 may also be implemented by requiring the service request to include the entire *poToken* information or the *refID* instead of *servicePermissionIDs.* The *poToken* information may also be augmented to include additional security functionality for authorization and authentication of the request and the issuer.

[0143] Fig. 11 is a diagram of an example a resource 1100 in accordance with one embodiment, which may be used in combination with any of the embodiments described herein. The example of Fig. 11 includes an example resource 1100 for the PO representation as <poResource> 1100, which may be used in a RESTful environment such as oneM2M. The example of Fig. 11 also includes an organization using the informational blocks <poDescription> 1111, <assocEntities> 1112, <assocPolicies> 1113, and <poToken> 1114, but other organizations of this information (e.g. flat) may also be used. The <poResource> 1110 may be created at PO registration time, for example in the procedures described herein. In oneM2M, the POMS may create this resource in a variety of ways, including but not limited to the following: as a child of the <CSEbase> root resource, as a child of an <AE> resource representing the application which initiated the registration, as a child of <container> resource which encapsulates one or more PO representations, by using a new <POrepository> resource type that includes all the PO representations available locally to POMs, etc.

[0144] Alternatively, some blocks of information may be stored remotely. For example the *poToken* information may be only associated with the <poResource> 1110 via the *refID* and *poID,* so it may be stored elsewhere. Many attributes such as those pertaining to associated entities are pointers or links to external databases or other resources. Attributes such as *poIdentificationInfo*, *servicePermissionContext,* etc. as well as the policies and the lists may be implemented as complex attributes, lists, etc. rather than single parameters.

[0145] Fig. 12 is a diagram of an example procedure 1200 for remote scouting and GW *poToken* generation in accordance with one embodiment, which may be used in combination with any of the embodiments described herein. The example procedure 1200 of Fig. 12 may be implemented in a RESTful environment (e.g. oneM2M). In this example, the PO may be registered by a scoutApp after being recognized and identified remotely. Referring to Fig. 12, PO-related policies may be provisioned 1210 on the platform for all the service layer and application entities. All applications 1204 and scoutApp(s) 1201 may perform

registration 1211, 1212 at the service layer that also hosts the POMS 1202. Registration at the service layer may provide the POMS 1202 with *poRelatedCapabilities* and *poPolicies* for the registrees through the enhancements to the existing registration procedures.

[0146] The scoutApp(s) 1201 may trigger a service layer PO registration 1213 with the POMS 1202. The scoutApp(s) 1201 may provide recognition and identification services and *poToken* generation 1214, which may be, for example, smart camera enabled recognition services, and the scoutApp(s) 1201 may store its information in the service layer. In this example the *poToken* generation is provided by scoutApp(s) 1201. For example, this may be triggered, for example, when the recognition and identification functionality in a smart camera determines that an automobile has been recognized. The triggering is subject to the local policies provided when provisioned 1210, e.g. with triggeringRules, scoutingTypeRules, poScopeRules, scoutingScope rules, identificationRules, etc. Other policies such as registrationRules are also used to determine when to trigger the process, as well as the parameters contained in the request. The scoutApp(s) 1201 may transmit a PO registration request message 1215 to the POMS 1202 during PO registration 1213. The POMS 1202 may perform the PO registration including the creation of the PO resources and information updates 1216 of the repository 1203. The POMS 1202 may also store the *poToken* for use in arbitration of services. The POMS 1202 may transmit a PO registration confirmation response 1217 to the scoutApp 1201 initiating the PO registration 1213.

[0147] After completing the PO registration 1213, PO management 1218 functionality at the POMS 1202 may include:

[0148] Maintaining *auxServiceList*, *assocAuxServiceReq*, and *auxServRequestList* 1219, for example by adding newly registered applications with auxiliary PO-related services;

[0149] Generating notifications to applications interested that a new PO has been registered, and generating announcements of the newly registered PO 1220. For example, the POMS 1202 may announce the PO to semantic annotation services in order to augment its information; and

[0150] Arbitrating PO-related service requests by using the *poToken* information as described herein, and processing the auxiliary PO-related service requests 1221 by making the corresponding function calls on behalf of the issuer. For example, a request for augmented information may result in calls to *scoutApp_n* 1201 an additional camera that may be used for tracking services.

[0151] Fig. 13 is a diagram of an example procedure 1300 for remote scouting and GW *poToken* generation where the PO is pre-defined and provided to the platform via the repository in accordance with another embodiment, which may be used in combination with any of the embodiments described herein. PO-related policies may be provisioned 1310 on the platform for all the service layer and application entities. All applications 1304 and the scoutApp(s) 1301 may perform registration at the service layer 1311, 1312 that also hosts POMS 1302. A PO repository entry may be created (*poEntry*) 1314. The POMS 1302 may be triggered by the repository 1303 with a notification about the PO entry or other POMS event 1315. The triggering may be accomplished by other means such as timer, requests from other entities, etc. The POMS 1302 may execute a RETRIEVE request 1316 of the PO-related information at the repository, e.g. *poEntry*. This information may be included in the earlier notification. Also, the POMS 1302 may retrieve other relevant information, for example may perform a discovery over other repositories/databases in order to obtain all information necessary for the PO registration 1313.

[0152] The POMS 1302 may perform the PO registration including the creation of the PO resources, *poToken* generation, and association of the *poEntry* in the repository with the created PO representation 1318, such that future updates to the PO resource may be mirrored in the repository, if applicable.

[0153] The PO management 1319 of the registered PO is performed by the POMS 1302, which may include:

[0154] Maintaining *auxServiceList*, *assocAuxServiceReq*, and *auxServRequestList* 1320, for example by adding newly registered applications with auxiliary PO-related services;

[0155] Generating notifications to applications interested that a new PO has been registered, and generating announcements of the newly registered PO 1321. For example, the POMS 1302 may announce the PO to semantic annotation services in order to augment its information; and

[0156] Arbitrating PO-related service requests by using the *poToken* information as described herein, and processing the auxiliary PO-related service requests 1322 by making the corresponding function calls on behalf of the issuer. For example, a request for augmented information may result in calls to *scoutApp_n* 1301 an additional camera that may be used for tracking services.

[0157] Fig. 14 is a diagram of an example graphical user interface (GUI) 1400 for entering pre-defined POs in a repository, which may be either local to a service layer or an external database. More or less information than that included in the example of Fig. 14 may be available for entry. The GUI 1400 may be specialized. For example, the GUI 1400 for an industrial database may include fields to detail manufacturer, measurements, etc. given the types of objects included. It may also offer the opportunity to upload 2D or 3D models, photos, videos, schematics, etc. Conversely, an employee database might include names, addresses, uploaded pictures, etc.

[0158] Fig. 15 is a diagram of another example GUI 1500 that may be used for searching for the representations of a PO on a service layer platform or associated database. The GUI 1500 in the example of Fig. 15 may also be customized and include fields characteristic to only one object type, e.g. manufacturer, measurements, etc. The GUI 1500 may be used for both pre-defined POs and scouted POs that have been registered on the service layer platform.

[0159] Fig. 16A is a diagram of an example machine-to-machine (M2M), Internet of Things (IoT), or Web of Things (WoT) communication system 10 in which one or more disclosed embodiments may be implemented. Generally, M2M technologies provide building blocks for the IoT/WoT, and any M2M device, M2M gateway or M2M service platform may be a component of the IoT/WoT as well as an IoT/WoT service layer, etc. Any of the devices, functions, nodes, or networks illustrated in any of Figs. 1 to 15 may comprise a node of a communication system such as the one illustrated in Figs. 16A-B.

[0160] As shown in Fig. 16A, the M2M/IoT/WoT communication system 10 includes a communication network 12. The communication network 12 may be a fixed network (e.g., Ethernet, Fiber, ISDN, PLC, or the like) or a wireless network (e.g., WLAN, cellular, or the like) or a network of heterogeneous networks. For example, the communication network 12 may comprise multiple access networks that provides content such as voice, data, video, messaging, broadcast, or the like to multiple users. For example, the communication network 12 may employ one or more channel access methods, such as code division multiple access (CDMA), time division multiple access (TDMA), frequency division multiple access (FDMA), orthogonal FDMA (OFDMA), single-carrier FDMA (SC-FDMA), and the like. Further, the communication network 12 may comprise other networks such as a core network, the Internet, a sensor network, an industrial control network, a

personal area network, a fused personal network, a satellite network, a home network, or an enterprise network for example.

[0161] As shown in Fig. 16A, the M2M/ IoT/WoT communication system 10 may include the Infrastructure Domain and the Field Domain. The Infrastructure Domain refers to the network side of the end-to-end M2M deployment, and the Field Domain refers to the area networks, usually behind an M2M gateway. The Field Domain and Infrastructure Domain may both comprise a variety of different nodes (e.g., servers, gateways, devices, of the network. For example, the Field Domain may include M2M gateways 14 and terminal devices 18. It will be appreciated that any number of M2M gateway devices 14 and M2M terminal devices 18 may be included in the M2M/ IoT/WoT communication system 10 as desired. Each of the M2M gateway devices 14 and M2M terminal devices 18 are configured to transmit and receive signals via the communication network 12 or direct radio link. A M2M gateway device 14 allows wireless M2M devices (e.g. cellular and non-cellular) as well as fixed network M2M devices (e.g., PLC) to communicate either through operator networks, such as the communication network 12 or direct radio link. For example, the M2M devices 18 may collect data and send the data, via the communication network 12 or direct radio link, to an M2M application 20 or M2M devices 18. The M2M devices 18 may also receive data from the M2M application 20 or an M2M device 18. Further, data and signals may be sent to and received from the M2M application 20 via an M2M service layer 22, as described below. M2M devices 18 and gateways 14 may communicate via various networks including, cellular, WLAN, WPAN (e.g., Zigbee, 6LoWPAN, Bluetooth), direct radio link, and wireline for example. Exemplary M2M devices include, but are not limited to, tablets, smart phones, medical devices, temperature and weather monitors, connected cars, smart meters, game consoles personal digital assistants, health and fitness monitors, lights, thermostats, appliances, garage doors and other actuator-based devices, security devices, and smart outlets.

[0162] Referring to Fig. 16B, the illustrated M2M service layer 22 in the field domain provides services for the M2M application 20, M2M gateway devices 14, and M2M terminal devices 18 and the communication network 12. It will be understood that the M2M service layer 22 may communicate with any number of M2M applications, M2M gateway devices 14, M2M terminal devices 18, and communication networks 12 as desired. The M2M service layer 22 may be implemented by one or more servers, computers, or the like. The M2M service layer 22 provides service capabilities that apply to M2M terminal devices 18,

M2M gateway devices 14 and M2M applications 20. The functions of the M2M service layer 22 may be implemented in a variety of ways, for example as a web server, in the cellular core network, in the cloud, etc.

[0163] Similar to the illustrated M2M service layer 22, there is the M2M service layer 22' in the Infrastructure Domain. M2M service layer 22' provides services for the M2M application 20' and the underlying communication network 12' in the infrastructure domain. M2M service layer 22' also provides services for the M2M gateway devices 14 and M2M terminal devices 18 in the field domain. It will be understood that the M2M service layer 22' may communicate with any number of M2M applications, M2M gateway devices and M2M terminal devices. The M2M service layer 22' may interact with a service layer by a different service provider. The M2M service layer 22' may be implemented by one or more servers, computers, virtual machines (e.g., cloud/compute/storage farms, etc.) or the like.

[0164] Still referring to Fig. 16B, the M2M service layer 22 and 22' provide a core set of service delivery capabilities that diverse applications and verticals can leverage. These service capabilities enable M2M applications 20 and 20' to interact with devices and perform functions such as data collection, data analysis, device management, security, billing, service/device discovery, etc. Essentially, these service capabilities free the applications of the burden of implementing these functionalities, thus simplifying application development and reducing cost and time to market. The service layer 22 and 22' also enables M2M applications 20 and 20' to communicate through various networks 12 and 12' in connection with the services that the service layer 22 and 22' provide.

[0165] The M2M applications 20 and 20' may include applications in various industries such as, without limitation, transportation, health and wellness, connected home, energy management, asset tracking, and security and surveillance. As mentioned above, the M2M service layer, running across the devices, gateways, and other servers of the system, supports functions such as, for example, data collection, device management, security, billing, location tracking/geofencing, device/service discovery, and legacy systems integration, and provides these functions as services to the M2M applications 20 and 20'.

[0166] Generally, a service layer (SL), such as the service layers 22 and 22' illustrated in Figs. 16A and 16B, defines a software middleware layer that supports value-added service capabilities through a set of application programming interfaces (APIs) and underlying networking interfaces. Both the ETSI M2M and oneM2M architectures define a

service layer. ETSI M2M's service layer is referred to as the Service Capability Layer (SCL). The SCL may be implemented in a variety of different nodes of the ETSI M2M architecture. For example, an instance of the service layer may be implemented within an M2M device (where it is referred to as a device SCL (DSCL)), a gateway (where it is referred to as a gateway SCL (GSCL)) and/or a network node (where it is referred to as a network SCL (NSCL)). The oneM2M service layer supports a set of Common Service Functions (CSFs) (i.e. service capabilities). An instantiation of a set of one or more particular types of CSFs is referred to as a Common Services Entity (CSE), which can be hosted on different types of network nodes (e.g. infrastructure node, middle node, application-specific node). The Third Generation Partnership Project (3GPP) has also defined an architecture for machine-type communications (MTC). In that architecture, the service layer, and the service capabilities it provides, are implemented as part of a Service Capability Server (SCS). Whether embodied in a DSCL, GSCL, or NSCL of the ETSI M2M architecture, in a Service Capability Server (SCS) of the 3GPP MTC architecture, in a CSF or CSE of the oneM2M architecture, or in some other node of a network, an instance of the service layer may be implemented in a logical entity (e.g., software, computer-executable instructions, and the like) executing either on one or more standalone nodes in the network, including servers, computers, and other computing devices or nodes, or as part of one or more existing nodes. As an example, an instance of a service layer or component thereof may be implemented in the form of software running on a network node (e.g., server, computer, gateway, device, or the like) having the general architecture illustrated in Fig. 16C or 16D described below.

[0167] Further, the methods and functionalities described herein may be implemented as part of an M2M network that uses a Service Oriented Architecture (SOA) and/or a resource-oriented architecture (ROA) to access services, such as the above-described Network and Application Management Service for example.

[0168] Fig. 16C is a block diagram of an example hardware/software architecture of a node of a network, such as one of the nodes, devices, functions, or networks illustrated in Figs. 1 to 7, which may operate as an M2M server, gateway, device, or other node in an M2M network such as that illustrated in Figs. 16A and 16B. As shown in Fig. 16C, the node 30 may include a processor 32, a transceiver 34, a transmit/receive element 36, a speaker/microphone 38, a keypad 40, a display/touchpad 42, non-removable memory 44, removable memory 46, a power source 48, a global positioning system (GPS) chipset 50, and

other peripherals 52. The node 30 may also include communication circuitry, such as a transceiver 34 and a transmit/receive element 36. It will be appreciated that the node 30 may include any sub-combination of the foregoing elements while remaining consistent with an embodiment. This node may be a node that implements the notifications and triggers related thereto described herein.

[0169] The processor 32 may be a general purpose processor, a special purpose processor, a conventional processor, a digital signal processor (DSP), a plurality of microprocessors, one or more microprocessors in association with a DSP core, a controller, a microcontroller, Application Specific Integrated Circuits (ASICs), Field Programmable Gate Array (FPGAs) circuits, any other type of integrated circuit (IC), a state machine, and the like. The processor 32 may perform signal coding, data processing, power control, input/output processing, and/or any other functionality that enables the node 30 to operate in a wireless environment. The processor 32 may be coupled to the transceiver 34, which may be coupled to the transmit/receive element 36. While Fig. 16C depicts the processor 32 and the transceiver 34 as separate components, it will be appreciated that the processor 32 and the transceiver 34 may be integrated together in an electronic package or chip. The processor 32 may perform application-layer programs (e.g., browsers) and/or radio access-layer (RAN) programs and/or communications. The processor 32 may perform security operations such as authentication, security key agreement, and/or cryptographic operations, such as at the access-layer and/or application layer for example.

[0170] As shown in Fig. 16C, the processor 32 is coupled to its communication circuitry (e.g., transceiver 34 and transmit/receive element 36). The processor 32, through the execution of computer executable instructions, may control the communication circuitry in order to cause the node 30 to communicate with other nodes via the network to which it is connected. In particular, the processor 32 may control the communication circuitry in order to perform the transmitting and receiving steps described herein (e.g., in Figs. 1-15) and in the claims. While Fig. 16C depicts the processor 32 and the transceiver 34 as separate components, it will be appreciated that the processor 32 and the transceiver 34 may be integrated together in an electronic package or chip.

[0171] The transmit/receive element 36 may be configured to transmit signals to, or receive signals from, other nodes, including M2M servers, gateways, devices, and the like. For example, in an embodiment, the transmit/receive element 36 may be an antenna

configured to transmit and/or receive RF signals. The transmit/receive element 36 may support various networks and air interfaces, such as WLAN, WPAN, cellular, and the like. In an embodiment, the transmit/receive element 36 may be an emitter/detector configured to transmit and/or receive IR, UV, or visible light signals, for example. In yet another embodiment, the transmit/receive element 36 may be configured to transmit and receive both RF and light signals. It will be appreciated that the transmit/receive element 36 may be configured to transmit and/or receive any combination of wireless or wired signals.

[0172] In addition, although the transmit/receive element 36 is depicted in Fig. 16C as a single element, the node 30 may include any number of transmit/receive elements 36. More specifically, the node 30 may employ MIMO technology. Thus, in an embodiment, the node 30 may include two or more transmit/receive elements 36 (e.g., multiple antennas) for transmitting and receiving wireless signals.

[0173] The transceiver 34 may be configured to modulate the signals that are to be transmitted by the transmit/receive element 36 and to demodulate the signals that are received by the transmit/receive element 36. As noted above, the node 30 may have multi-mode capabilities. Thus, the transceiver 34 may include multiple transceivers for enabling the node 30 to communicate via multiple RATs, such as UTRA and IEEE 802.11, for example.

[0174] The processor 32 may access information from, and store data in, any type of suitable memory, such as the non-removable memory 44 and/or the removable memory 46. The non-removable memory 44 may include random-access memory (RAM), read-only memory (ROM), a hard disk, or any other type of memory storage device. The removable memory 46 may include a subscriber identity module (SIM) card, a memory stick, a secure digital (SD) memory card, and the like. In other embodiments, the processor 32 may access information from, and store data in, memory that is not physically located on the node 30, such as on a server or a home computer. The processor 32 may be configured to control lighting patterns, images, or colors on the display or indicators 42 to reflect the status of a node or configure a node (e.g., nodes in Figs.1-7), and in particular underlying networks, applications, or other services in communication with the UE. The processor 32 may receive power from the power source 48, and may be configured to distribute and/or control the power to the other components in the node 30. The power source 48 may be any suitable device for powering the node 30. For example, the power source 48 may include one or more dry cell

batteries (e.g., nickel-cadmium (NiCd), nickel-zinc (NiZn), nickel metal hydride (NiMH), lithium-ion (Li-ion), etc.), solar cells, fuel cells, and the like.

[0175] The processor 32 may also be coupled to the GPS chipset 50, which is configured to provide location information (e.g., longitude and latitude) regarding the current location of the node 30. It will be appreciated that the node 30 may acquire location information by way of any suitable location-determination method while remaining consistent with an embodiment.

[0176] The processor 32 may further be coupled to other peripherals 52, which may include one or more software and/or hardware modules that provide additional features, functionality, and/or wired or wireless connectivity. For example, the peripherals 52 may include various sensors such as an accelerometer, biometrics (e.g., fingerprint) sensors, an e-compass, a satellite transceiver, a digital camera (for photographs or video), a universal serial bus (USB) port or other interconnect devices, a vibration device, a television transceiver, a hands free headset, a Bluetooth® module, a frequency modulated (FM) radio unit, a digital music player, a media player, a video game player module, an Internet browser, and the like.

[0177] Fig. 16D is a block diagram of an exemplary computing system 90 which may also be used to implement one or more nodes of a network, such as nodes, devices, functions, or networks illustrated in Figs. 1-15, which may operate as an M2M server, gateway, device, or other node in an M2M network such as that illustrated in Figs. 16A and 16B. Computing system 90 may comprise a computer or server and may be controlled primarily by computer readable instructions, which may be in the form of software, wherever, or by whatever means such software is stored or accessed. Such computer readable instructions may be executed within central processing unit (CPU) 91 to cause computing system 90 to do work. In many known workstations, servers, and personal computers, central processing unit 91 is implemented by a single-chip CPU called a microprocessor. In other machines, the central processing unit 91 may comprise multiple processors. Coprocessor 81 is an optional processor, distinct from main CPU 91, which performs additional functions or assists CPU 91. CPU 91 and/or coprocessor 81 may receive, generate, and process data related to the disclosed systems and methods for security protection.

[0178] In operation, CPU 91 fetches, decodes, and executes instructions, and transfers information to and from other resources via the computer's main data-transfer path, system bus 80. Such a system bus connects the components in computing system 90 and

defines the medium for data exchange. System bus 80 typically includes data lines for sending data, address lines for sending addresses, and control lines for sending interrupts and for operating the system bus. An example of such a system bus 80 is the PCI (Peripheral Component Interconnect) bus.

[0179] Memory devices coupled to system bus 80 include random access memory (RAM) 82 and read only memory (ROM) 93. Such memories include circuitry that allows information to be stored and retrieved. ROMs 93 generally contain stored data that cannot easily be modified. Data stored in RAM 82 can be read or changed by CPU 91 or other hardware devices. Access to RAM 82 and/or ROM 93 may be controlled by memory controller 92. Memory controller 92 may provide an address translation function that translates virtual addresses into physical addresses as instructions are executed. Memory controller 92 may also provide a memory protection function that isolates processes within the system and isolates system processes from user processes. Thus, a program running in a first mode can access only memory mapped by its own process virtual address space; it cannot access memory within another process's virtual address space unless memory sharing between the processes has been set up.

[0180] In addition, computing system 90 may contain peripherals controller 83 responsible for communicating instructions from CPU 91 to peripherals, such as printer 94, keyboard 84, mouse 95, and disk drive 85.

[0181] Display 86, which is controlled by display controller 96, is used to display visual output generated by computing system 90. Such visual output may include text, graphics, animated graphics, and video. Display 86 may be implemented with a CRT-based video display, an LCD-based flat-panel display, gas plasma-based flat-panel display, or a touch-panel. Display controller 96 includes electronic components required to generate a video signal that is sent to display 86.

[0182] Further, computing system 90 may contain communication circuitry, such as for example a network adaptor 97 that may be used to connect computing system 90 to an external communications network, such as network 12 of Fig. 16A and Fig. 16B, to enable the computing system 90 to communicate with other nodes of the network. The communication circuitry, alone or in combination with the CPU 91, may be used to perform the transmitting and receiving steps described herein (e.g., in Figs. 1-15) and in the claims.

[0183] In describing preferred embodiments of the subject matter of the present disclosure, as illustrated in the Figures, specific terminology is employed for the sake of clarity. The claimed subject matter, however, is not intended to be limited to the specific terminology so selected, and it is to be understood that each specific element includes all technical equivalents that operate in a similar manner to accomplish a similar purpose.

What is Claimed:

1. An apparatus comprising a processor, a memory, and communication circuitry, the apparatus being connected to a network via its communication circuitry, the apparatus further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform operations comprising:

receiving, from a computing device, a passive object registration request message comprising an identifier associated with a passive object, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities;

storing, in a database, the identifier and an associated resource to enable access by a plurality of service layer applications;

causing, based on using the identifier, control and management of the passive object by the plurality of service layer applications; and

transmitting, to the computing device, a registration confirmation message.

2. The apparatus of claim 1, further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform further operations comprising:

generating a passive object token comprising the identifier and information to enable the control and management of the passive object by the plurality of service layer applications; and

storing, in a database, the passive object token.

3. The apparatus of claim 2, wherein the registration confirmation message comprises the passive object token.

4. The apparatus as recited in claim 1, further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform further operations comprising:

receiving, from a service layer application of the plurality of service layer applications, a service request comprising information associated with a service type and a permission identifier comprising the identifier;

on a condition that the permission identifier is valid, executing the service request; and

transmitting, to the service layer application, a service request response comprising information associated with the executed service request.

5. The apparatus as recited in claim 1, wherein the apparatus is a server.

6. The apparatus as recited in claim 1, wherein the passive object does not have computational, telecommunication, or data sharing capabilities.

7. The apparatus as recited in claim 1, wherein the passive object is associated with a plurality of policies to be used by a scouting application to handle recognition, location, identification, management, and control of the passive object.

8. The apparatus as recited in claim 7, wherein the plurality of policies comprises a policy comprising rules indicating types of information associated with the passive object to be retained by the scouting application.

9. The apparatus as recited in claim 7, wherein the plurality of policies comprises a policy comprising rules indicating types of scouting performed by the scouting application and triggers for initiating scouting by the scouting application.

10. The apparatus as recited in claim 7, wherein the plurality of policies comprises a policy comprising rules indicating categories of passive objects to identify by the scouting application.

11. The apparatus as recited in claim 4, further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform further operations comprising:

receiving, from a second service layer application of the plurality of service layer applications, a second service request; and

determining access to the passive object by the plurality of service layer applications based on service permission identifiers associated with the passive object.

12. The apparatus as recited in claim 2, further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform further operations comprising:

receiving, from the service layer application, a request for the passive object token; and

on a condition that the service layer application is permitted by a policy to receive the passive object token, transmitting, to the service layer application, the passive object token.

13. The apparatus as recited in claim 1, wherein the information to enable control and management of the passive object includes policy information.

14. An apparatus comprising a processor, a memory, and communication circuitry, the apparatus being connected to a network via its communication circuitry, the apparatus further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform operations comprising:

receiving, from a computing device, a passive object registration request message comprising an identifier associated with a passive object, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities;

generating a passive object token comprising the identifier and information to enable control and management of the passive object by a plurality of service layer applications;

storing, in a database, the passive object token with an associated resource to enable access by the plurality of service layer applications; and

transmitting, to the computing device, a registration confirmation message comprising the passive object token.

15. The apparatus as recited in claim 14, further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform further operations comprising:

receiving, from a service layer application of the plurality of service layer applications, a service request comprising information associated with a service type and a permission identifier comprising the passive object token;

on a condition that the permission identifier is valid, executing the service request; and

transmitting, to the service layer application, a service request response comprising information associated with the executed service request.

16. The apparatus as recited in claim 14, wherein the apparatus is a server.

17. The apparatus as recited in claim 14, wherein the passive object does not have computational, telecommunication, or data sharing capabilities.

18. The apparatus as recited in claim 14, wherein the passive object is associated with a plurality of policies to be used by a scouting application to handle recognition, location, identification, management, and control of the passive object.

19. The apparatus as recited in claim 18, wherein the plurality of policies comprises a policy comprising rules indicating types of information associated with the passive object to be retained by the scouting application.

20. The apparatus as recited in claim 18, wherein the plurality of policies comprises a policy comprising rules indicating types of scouting performed by the scouting application and triggers for initiating scouting by the scouting application.

21. The apparatus as recited in claim 18, wherein the plurality of policies comprises a policy comprising rules indicating categories of passive objects to identify by the scouting application.

22. The apparatus as recited in claim 15, further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform further operations comprising:

receiving, from a second service layer application of the plurality of service layer applications, a second service request; and

determining access to the passive object by the plurality of service layer applications based on service permission identifiers included in the passive object token.

23. The apparatus as recited in claim 14, further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform further operations comprising:

receiving, from the service layer application, a request for the passive object token; and

on a condition that the service layer application is permitted by a policy to receive the passive object token, transmitting, to the service layer application, the passive object token.

24. The apparatus as recited in claim 14, wherein the information to enable control and management of the passive object comprises policy information.

25. An apparatus comprising a processor, a memory, and communication circuitry, the apparatus being connected to a network via its communication circuitry, the apparatus further comprising computer-executable instructions stored in the memory of the apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform operations comprising:

determining permissions and parameters associated with a passive object, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities;

receiving, from a plurality of service layer applications, a plurality of service requests that each comprise parameters associated with a service type and each comprise a permission identifier comprising a passive object token;

selecting a service request of the received plurality of service requests to execute based on the parameters and the permission identifier included in the selected service request; and

transmitting, to a service layer application of the plurality of service layer applications associated with the selected service request, a service request response comprising information associated with the executed service request.

26. The apparatus as recited in claim 25, wherein the apparatus is a server.

27. The apparatus as recited in claim 25, wherein the passive object does not have computational, telecommunication, or data sharing capabilities.

28. The apparatus as recited in claim 25, wherein the passive object is associated with a plurality of policies to be used by a scouting application to handle recognition, location, identification, management, and control of the passive object.

29. The apparatus as recited in claim 28, wherein the plurality of policies comprises a policy comprising rules indicating types of information associated with the passive object to be retained by the scouting application.

30. The apparatus as recited in claim 28, wherein the plurality of policies comprises a policy comprising rules indicating types of scouting performed by the scouting application and triggers for initiating scouting by the scouting application.

31. The apparatus as recited in claim 28, wherein the plurality of policies comprises a policy comprising rules indicating categories of passive objects to identify by the scouting application.

32. An apparatus comprising a processor, a memory, and communication circuitry, the apparatus being connected to a network via its communication circuitry, the apparatus further comprising computer-executable instructions stored in the memory of the

apparatus which, when executed by the processor of the apparatus, cause the apparatus to perform operations comprising:

receiving, from a service layer application of a plurality of service layer applications, a service request comprising information associated with a service type;

determining, based on the service request, a plurality of policies to be used to determine how to identify a passive object of a type of passive objects, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities;

identifying the passive object based on the plurality of policies;

executing the service request based on the identified passive object; and

transmitting, to the service layer application, a service request response comprising information associated with the executed service request.

33. The apparatus as recited in claim 32, wherein the apparatus is a server.

34. The apparatus as recited in claim 32, wherein the passive object does not have computational, telecommunication, or data sharing capabilities.

35. The apparatus as recited in claim 32, wherein the plurality of policies are to be used to recognize, locate, manage, and control the passive object.

36. The apparatus as recited in claim 35, wherein the plurality of policies comprises a policy comprising rules indicating types of information associated with the passive object to retain by the apparatus.

37. The apparatus as recited in claim 32, wherein the plurality of policies comprises a policy comprising rules indicating types of scouting performed by the apparatus and triggers for initiating scouting by the apparatus.

38. The apparatus as recited in claim 32, wherein the plurality of policies comprises a policy comprising rules indicating categories of passive objects to identify by the apparatus.

39. A method comprising:

receiving, from a computing device, a passive object registration request message comprising an identifier associated with a passive object, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities;

storing, in a database, the identifier and an associated resource to enable access by a plurality of service layer applications;

causing, based on using the identifier, control and management of the passive object by the plurality of service layer applications; and

transmitting, to the computing device, a registration confirmation message.

40. A method comprising:

receiving, from a computing device, a passive object registration request message comprising an identifier associated with a passive object, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities;

generating a passive object token comprising the identifier and information to enable control and management of the passive object by a plurality of service layer applications;

storing, in a database, the passive object token with an associated resource to enable access by the plurality of service layer applications; and

transmitting, to the computing device, a registration confirmation message comprising the passive object token.

41. A method comprising:

determining permissions and parameters associated with a passive object, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities;

receiving, from a plurality of service layer applications, a plurality of service requests that each comprise parameters associated with a service type and each comprise a permission identifier comprising a passive object token;

selecting a service request of the received plurality of service requests to execute based on the parameters and the permission identifier included in the selected service request; and

transmitting, to a service layer application of the plurality of service layer applications associated with the selected service request, a service request response comprising information associated with the executed service request.

42. A method comprising:

receiving, from a service layer application of a plurality of service layer applications, a service request comprising information associated with a service type;

determining, based on the service request, a plurality of policies to be used to determine how to identify a passive object of a type of passive objects, wherein the passive object is a physical object not defined by computational, telecommunication, or data sharing capabilities;

identifying the passive object based on the plurality of policies;

executing the service request based on the identified passive object; and

transmitting, to the service layer application, a service request response comprising information associated with the executed service request.

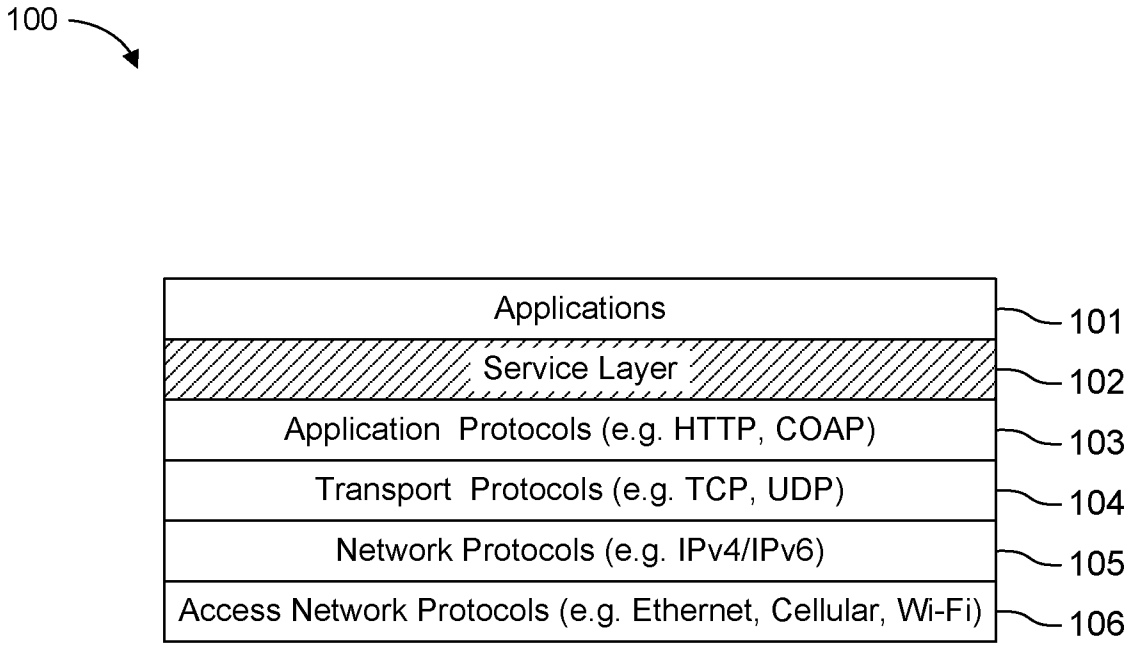
100
Applications
Service Layer
Application Protocols (e.g. HTTP, COAP)
Transport Protocols (e.g. TCP, UDP)
Network Protocols (e.g. IPv4/IPv6)
Access Network Protocols (e.g. Ethernet, Cellular, Wi-Fi)
101
102
103
104
105
106

FIG. 1

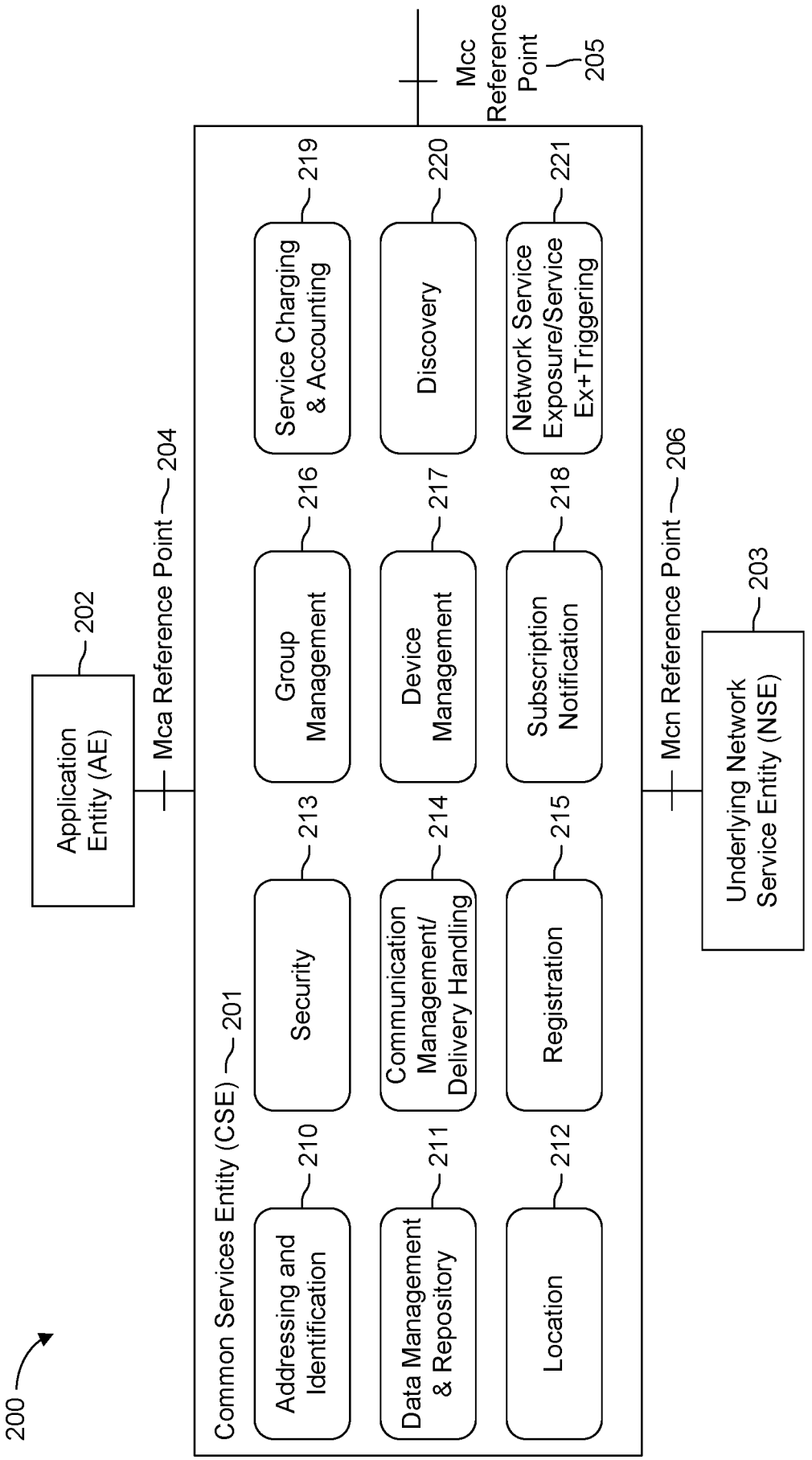
200
Application Entity (AE)
202
Mca Reference Point
204
Common Services Entity (CSE)
201
Addressing and Identification
210
Data Management & Repository
211
Location
212
Security
213
Communication Management/ Delivery Handling
214
Registration
215
Group Management
216
Device Management
217
Subscription Notification
218
Service Charging & Accounting
219
Discovery
220
Network Service Exposure/Service Ex+Triggering
221
Mcc Reference Point
205
Mcn Reference Point
206
Underlying Network Service Entity (NSE)
203

FIG. 2

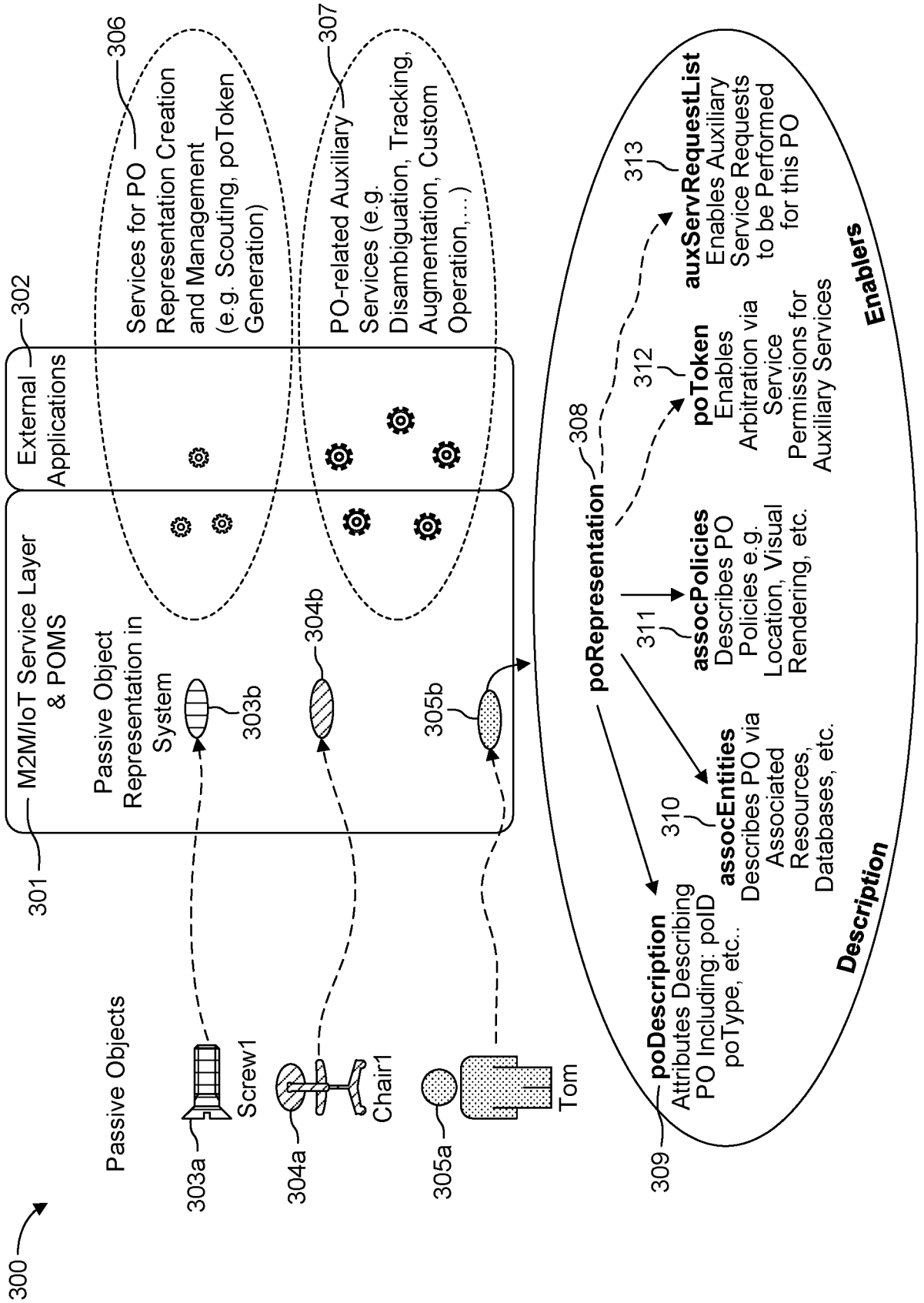
300
Passive Objects
Screw1
Chair1
Tom
303a
304a
305a
301
M2M/IoT Service Layer & POMS
External Applications
302
Passive Object Representation in System
303b
304b
305b
306
Services for PO Representation Creation and Management (e.g. Scouting, poToken Generation)
307
PO-related Auxiliary Services (e.g. Disambiguation, Tracking, Augmentation, Custom Operation,...)
Description
poDescription
Attributes Describing PO Including: poID poType, etc..
309
assocEntities
Describes PO via Associated Resources, Databases, etc.
310
poRepresentation
308
assocPolicies
Describes PO Policies e.g. Location, Visual Rendering, etc.
311
poToken
Enables Arbitration via Service Permissions for Auxiliary Services
312
auxServRequestList
Enables Auxiliary Service Requests to be Performed for this PO
313
Enablers

FIG. 3

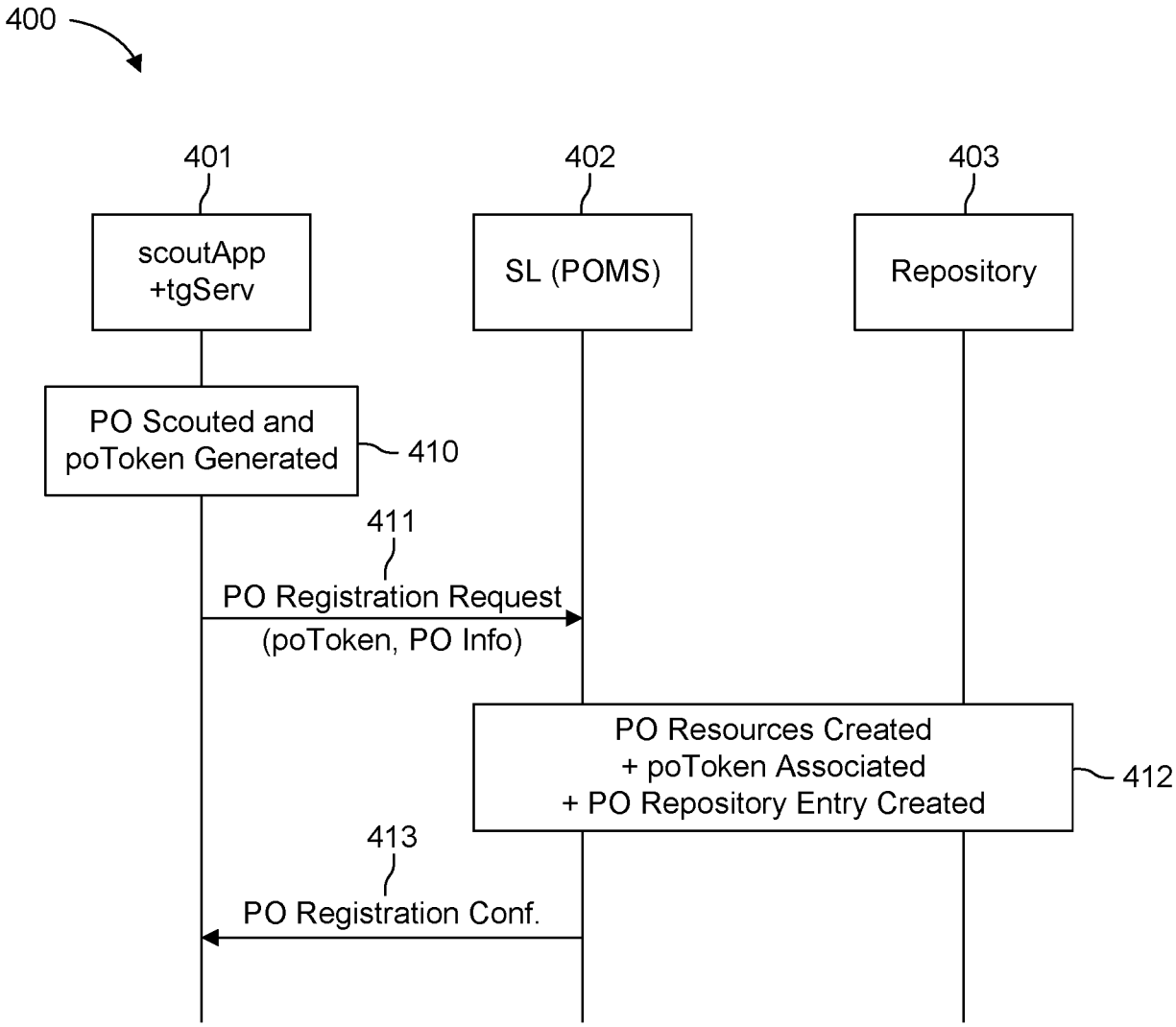
400
401
scoutApp
+tgServ
402
SL (POMS)
403
Repository
PO Scouted and
poToken Generated
410
411
PO Registration Request
(poToken, PO Info)
PO Resources Created
+ poToken Associated
+ PO Repository Entry Created
412
413
PO Registration Conf.

FIG. 4

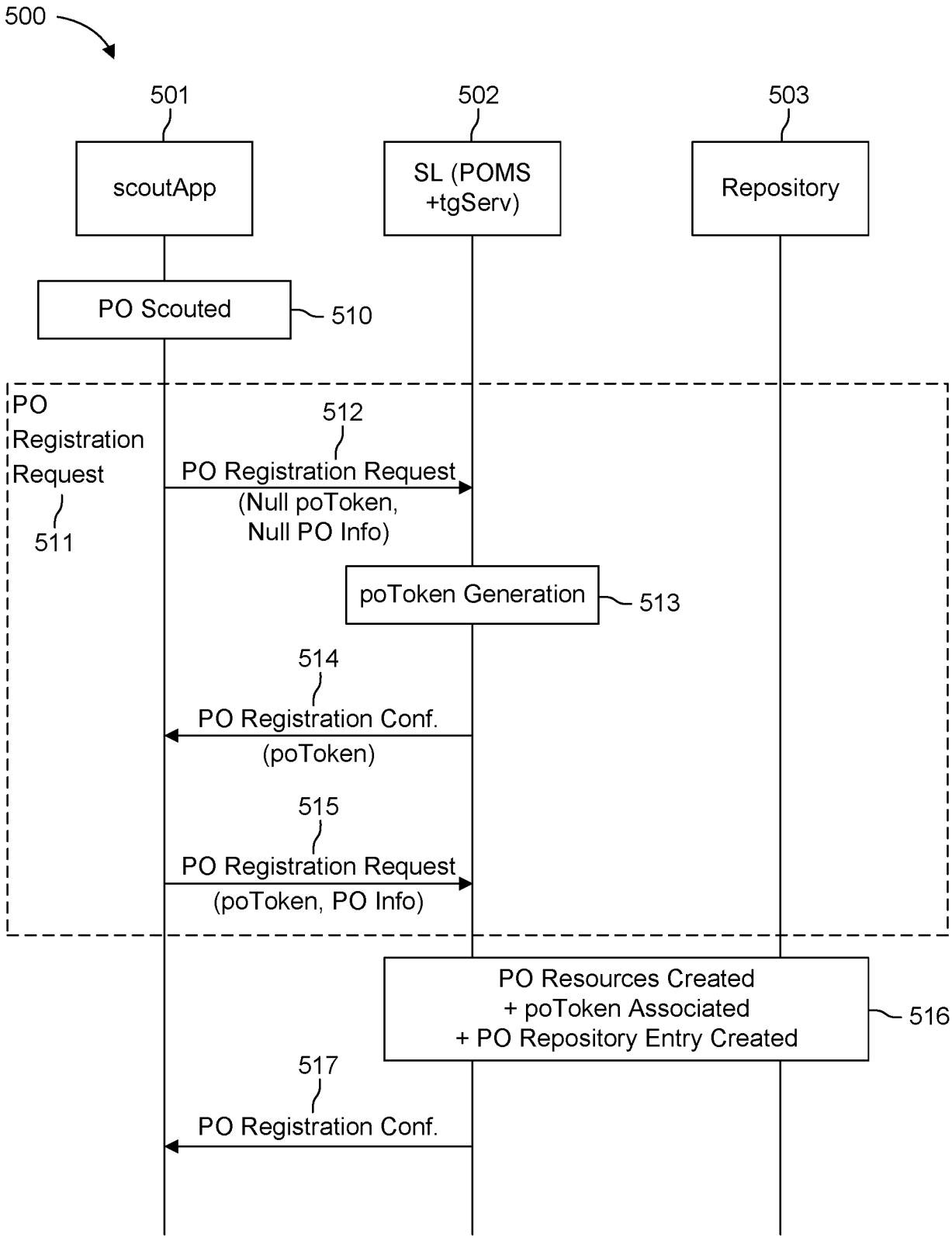
500
501
scoutApp
502
SL (POMS +tgServ)
503
Repository
PO Scouted
510
PO Registration Request
511
512
PO Registration Request
(Null poToken, Null PO Info)
poToken Generation
513
514
PO Registration Conf.
(poToken)
515
PO Registration Request
(poToken, PO Info)
PO Resources Created
+ poToken Associated
+ PO Repository Entry Created
516
517
PO Registration Conf.

FIG. 5

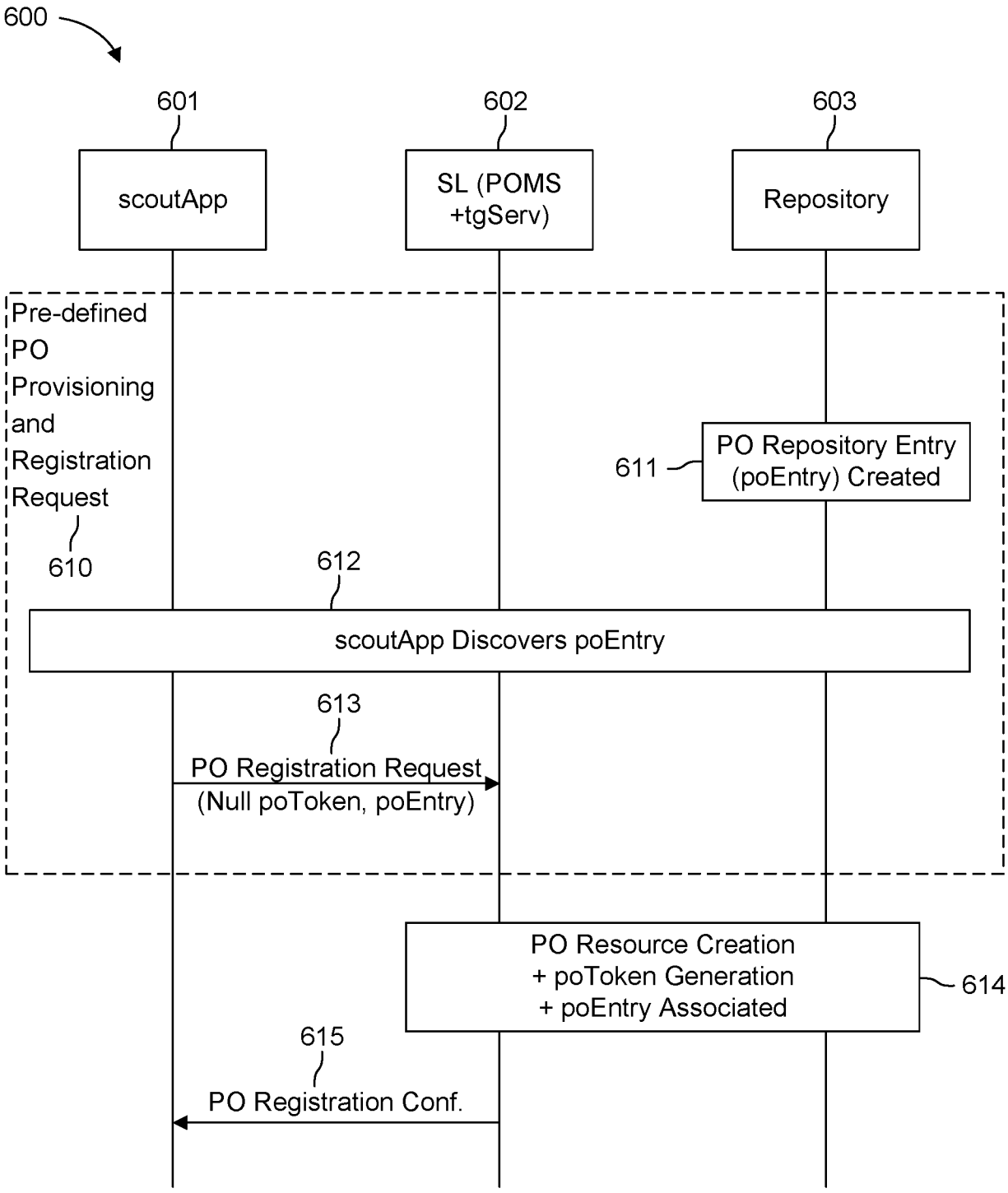
600
601
602
603
scoutApp
SL (POMS +tgServ)
Repository
Pre-defined PO Provisioning and Registration Request
610
611
PO Repository Entry (poEntry) Created
612
scoutApp Discovers poEntry
613
PO Registration Request (Null poToken, poEntry)
PO Resource Creation + poToken Generation + poEntry Associated
614
615
PO Registration Conf.

FIG. 6

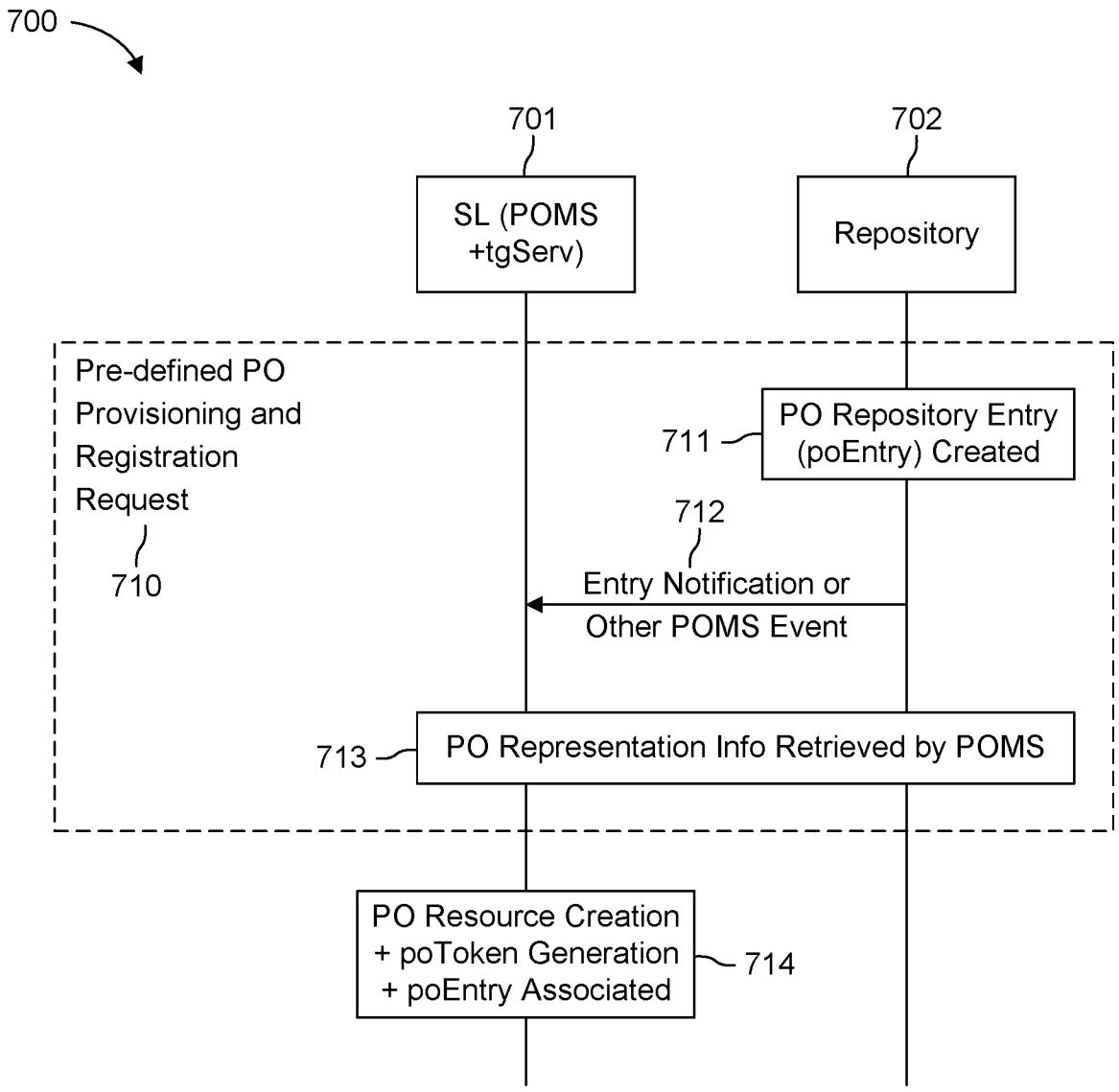
700
701
SL (POMS +tgServ)
702
Repository
Pre-defined PO Provisioning and Registration Request
710
711
PO Repository Entry (poEntry) Created
712
Entry Notification or Other POMS Event
713
PO Representation Info Retrieved by POMS
PO Resource Creation + poToken Generation + poEntry Associated
714

FIG. 7

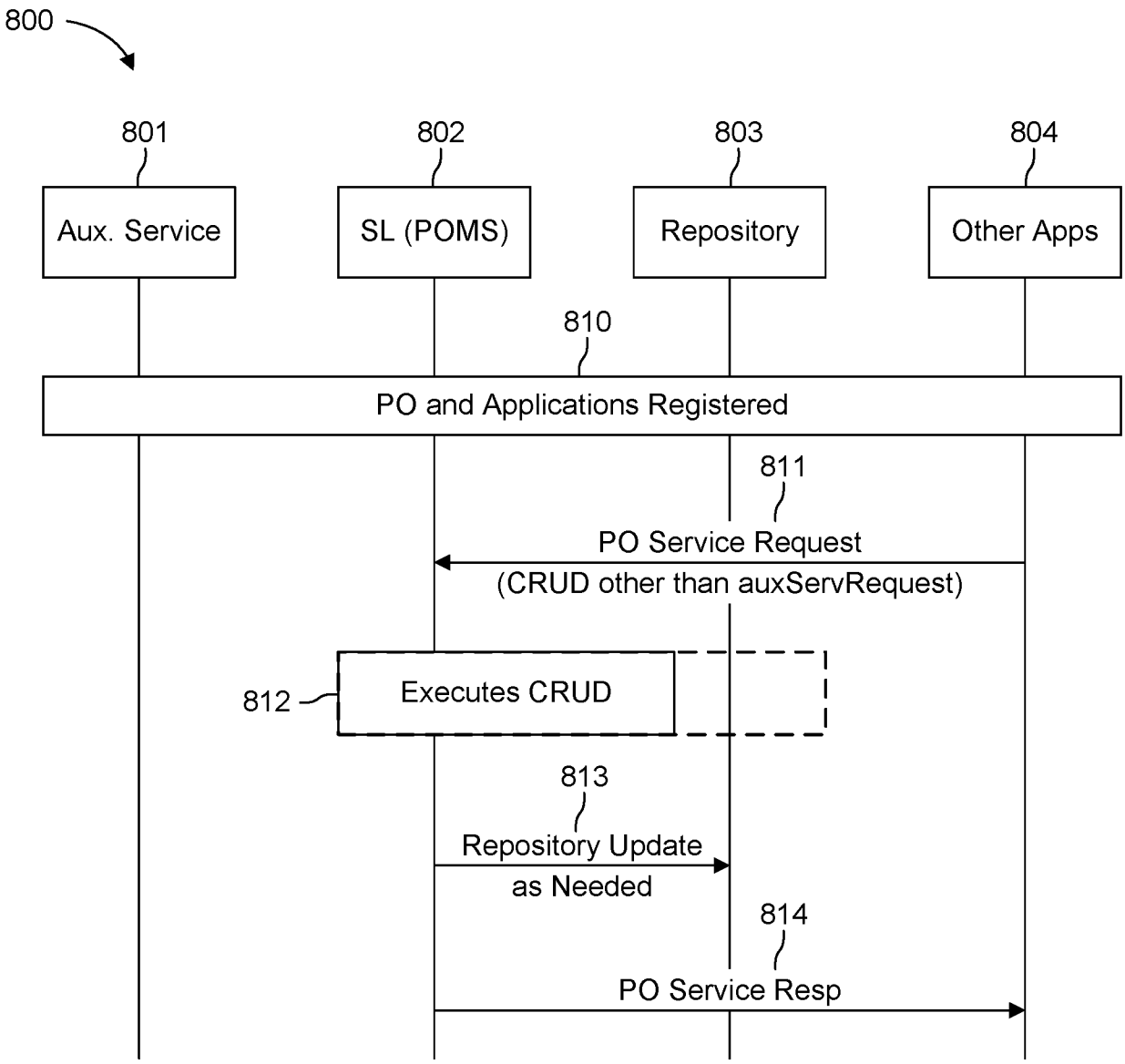
800
801
802
803
804
Aux. Service
SL (POMS)
Repository
Other Apps
810
PO and Applications Registered
811
PO Service Request
(CRUD other than auxServRequest)
812
Executes CRUD
813
Repository Update
as Needed
814
PO Service Resp

FIG. 8

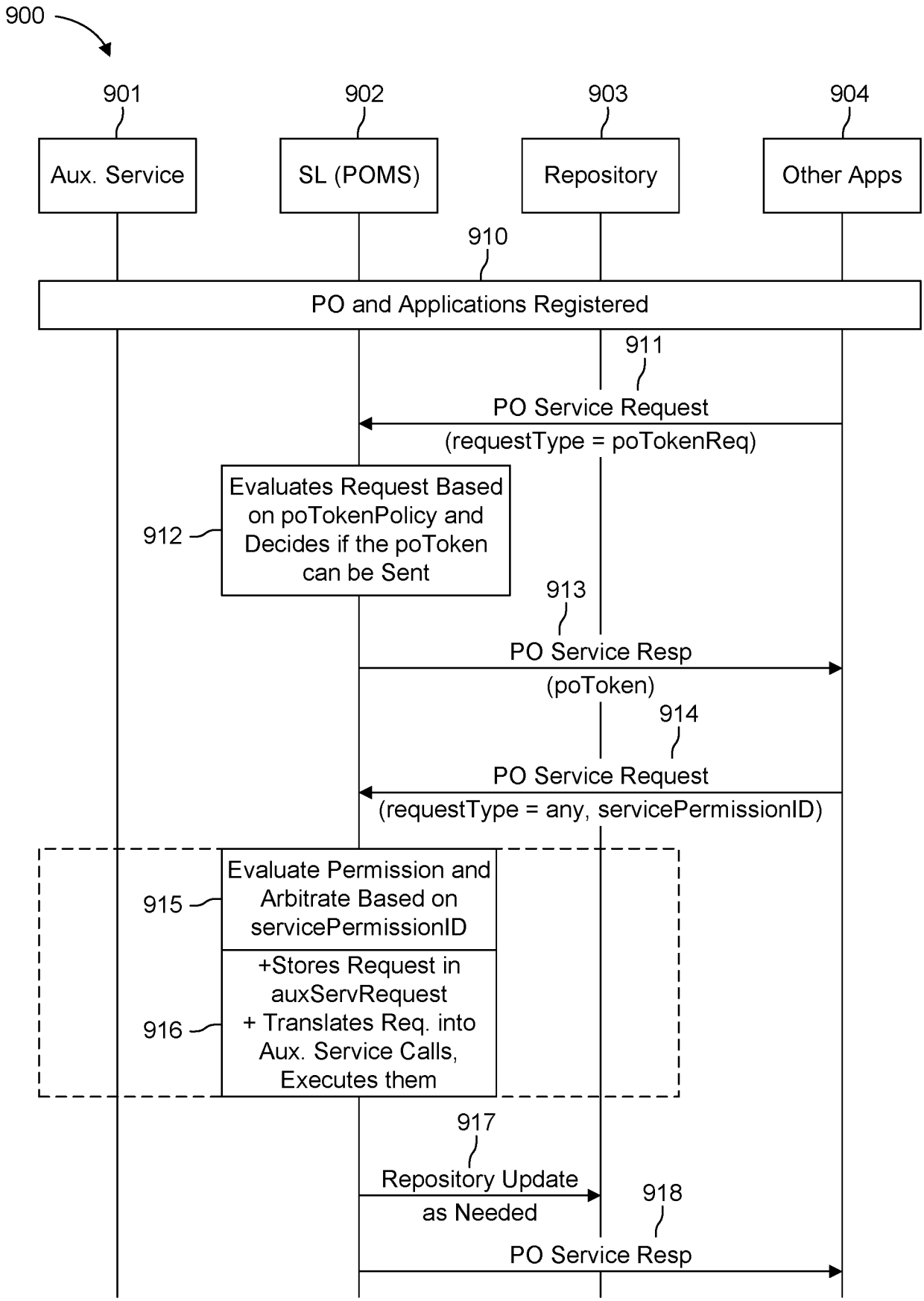
900
901
Aux. Service
902
SL (POMS)
903
Repository
904
Other Apps
910
PO and Applications Registered
911
PO Service Request
(requestType = poTokenReq)
912
Evaluates Request Based on poTokenPolicy and Decides if the poToken can be Sent
913
PO Service Resp
(poToken)
914
PO Service Request
(requestType = any, servicePermissionID)
915
Evaluate Permission and Arbitrate Based on servicePermissionID
916
+Stores Request in auxServRequest
+ Translates Req. into Aux. Service Calls, Executes them
917
Repository Update as Needed
918
PO Service Resp

FIG. 9

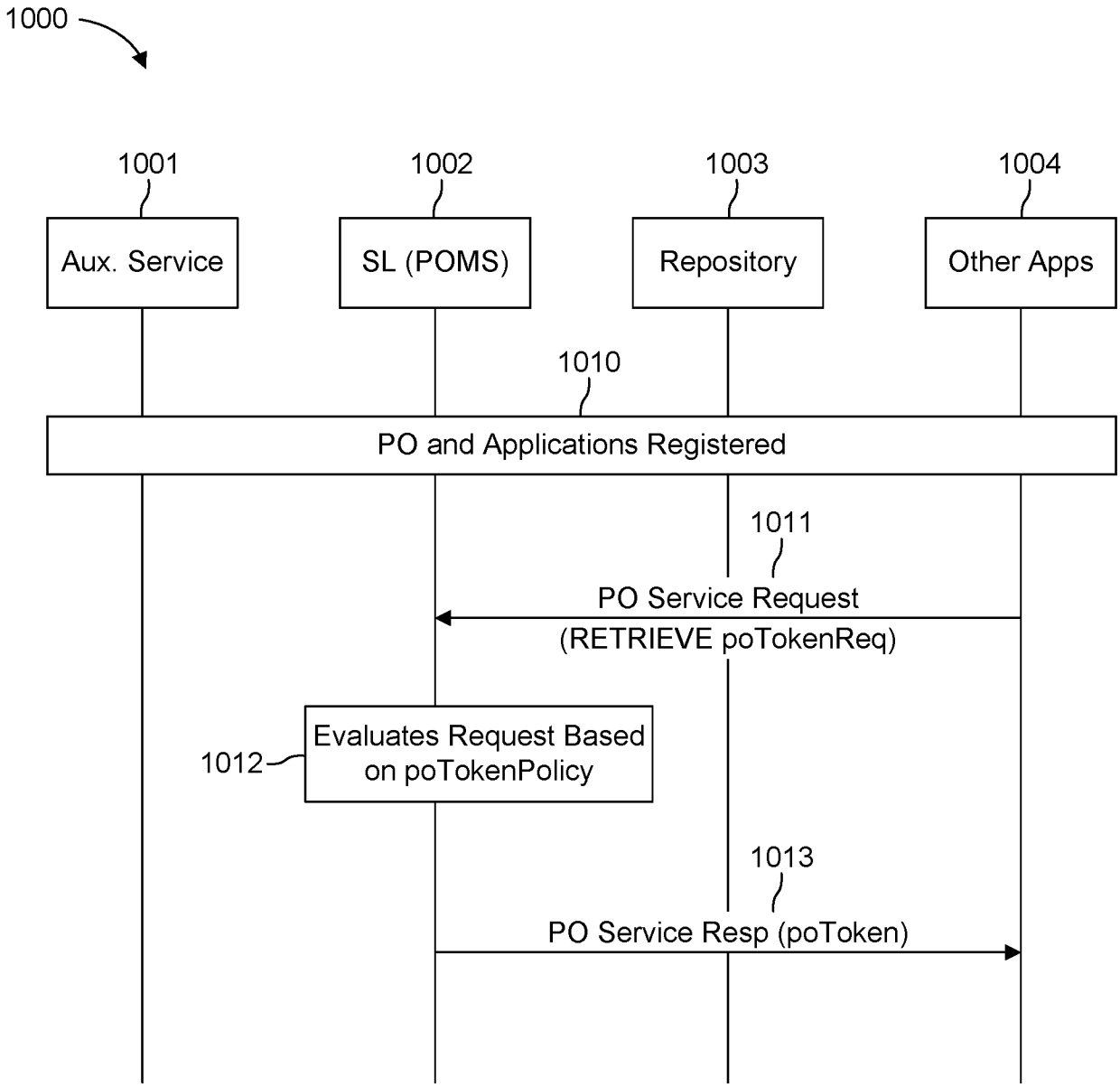
1000
1001
1002
1003
1004
Aux. Service
SL (POMS)
Repository
Other Apps
1010
PO and Applications Registered
1011
PO Service Request
(RETRIEVE poTokenReq)
1012
Evaluates Request Based
on poTokenPolicy
1013
PO Service Resp (poToken)

FIG. 10

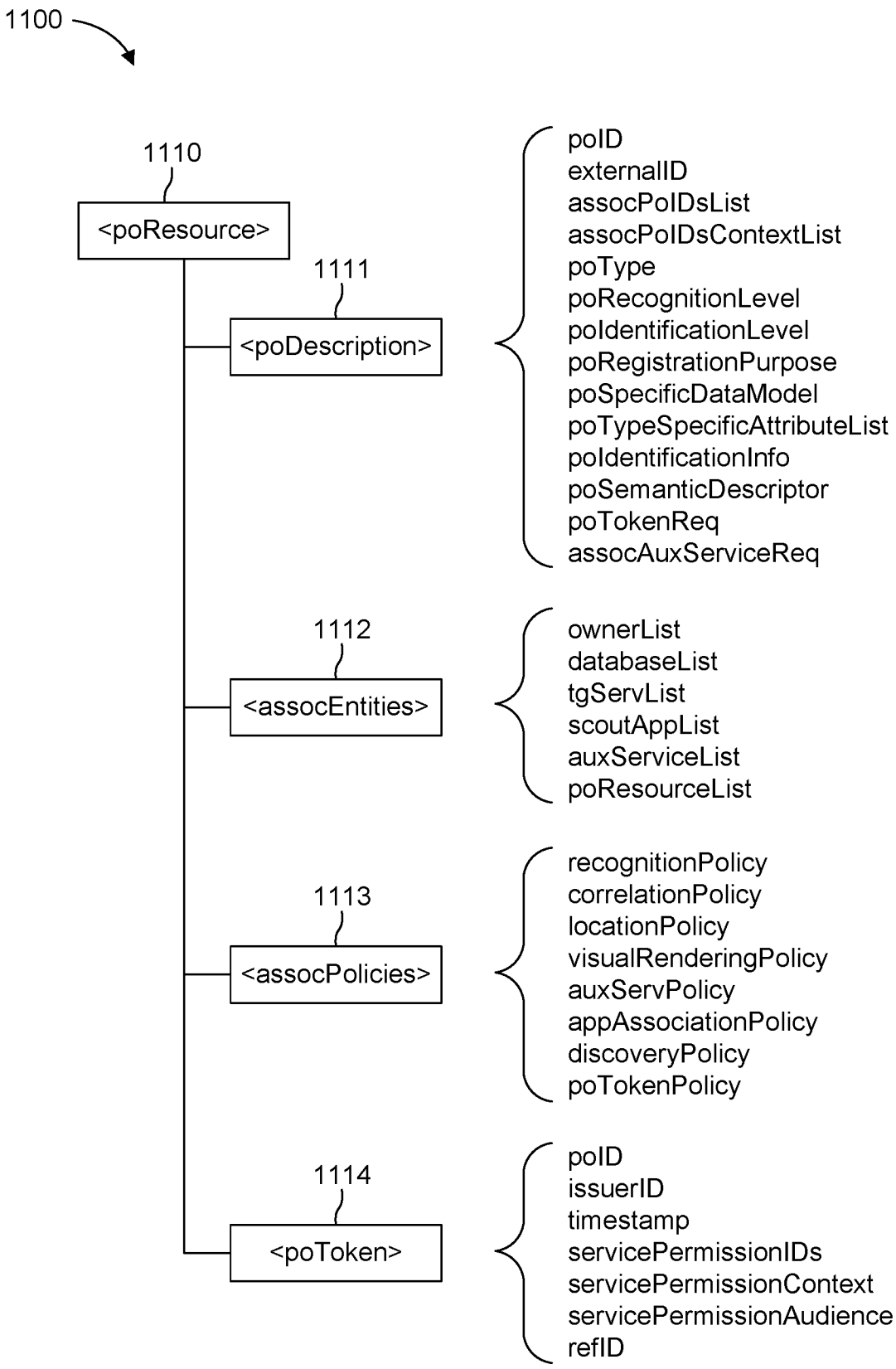
1100
1110
<poResource>
1111
<poDescription>
poID
externalID
assocPoIDsList
assocPoIDsContextList
poType
poRecognitionLevel
poIdentificationLevel
poRegistrationPurpose
poSpecificDataModel
poTypeSpecificAttributeList
poIdentificationInfo
poSemanticDescriptor
poTokenReq
assocAuxServiceReq
1112
<assocEntities>
ownerList
databaseList
tgServList
scoutAppList
auxServiceList
poResourceList
1113
<assocPolicies>
recognitionPolicy
correlationPolicy
locationPolicy
visualRenderingPolicy
auxServPolicy
appAssociationPolicy
discoveryPolicy
poTokenPolicy
1114
<poToken>
poID
issuerID
timestamp
servicePermissionIDs
servicePermissionContext
servicePermissionAudience
refID

FIG. 11

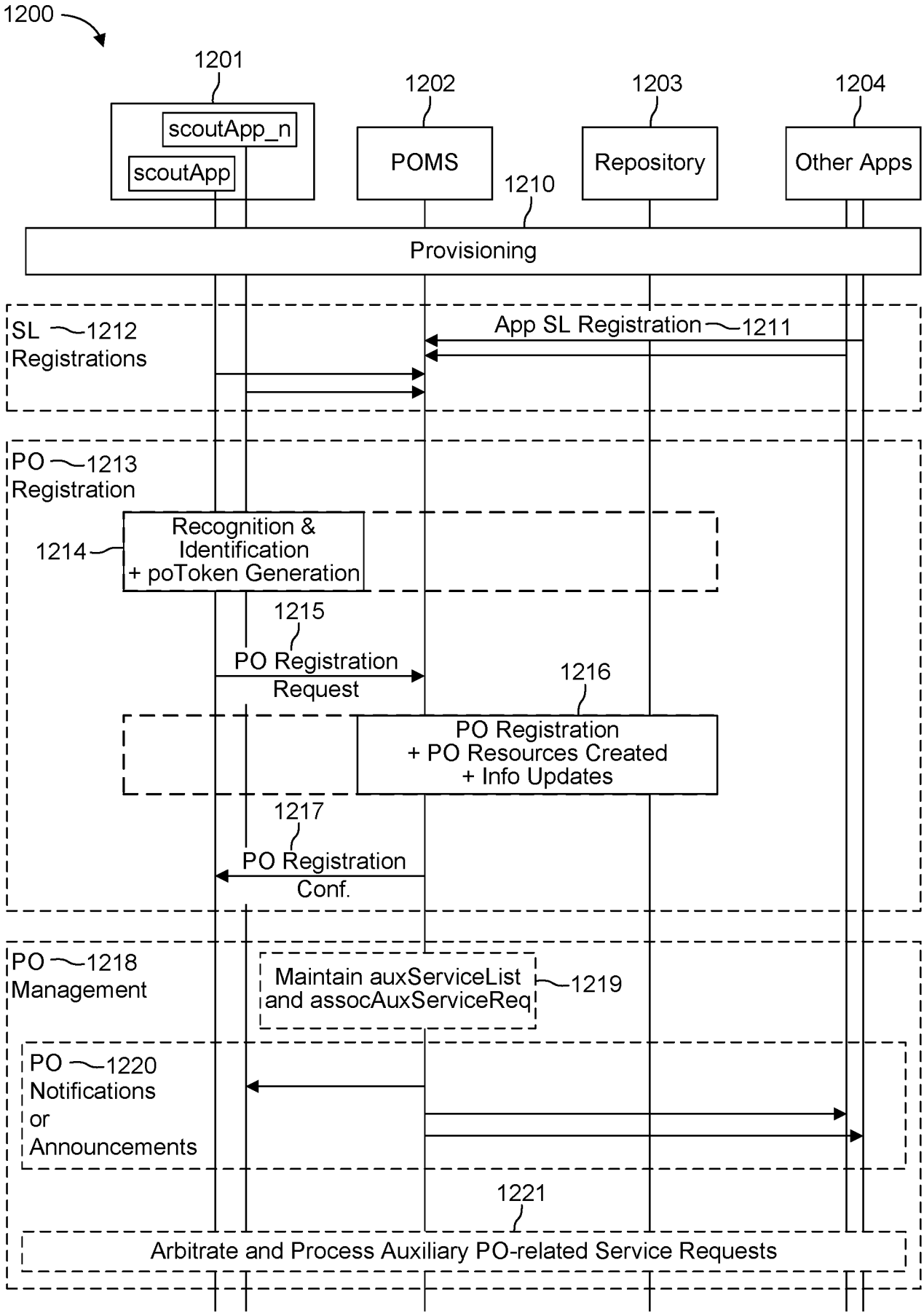
1200
1201
scoutApp_n
scoutApp
1202
POMS
1203
Repository
1204
Other Apps
1210
Provisioning
SL 1212
Registrations
App SL Registration 1211
PO 1213
Registration
1214
Recognition &
Identification
+ poToken Generation
1215
PO Registration
Request
1216
PO Registration
+ PO Resources Created
+ Info Updates
1217
PO Registration
Conf.
PO 1218
Management
Maintain auxServiceList
and assocAuxServiceReq
1219
PO 1220
Notifications
or
Announcements
1221
Arbitrate and Process Auxiliary PO-related Service Requests

FIG. 12

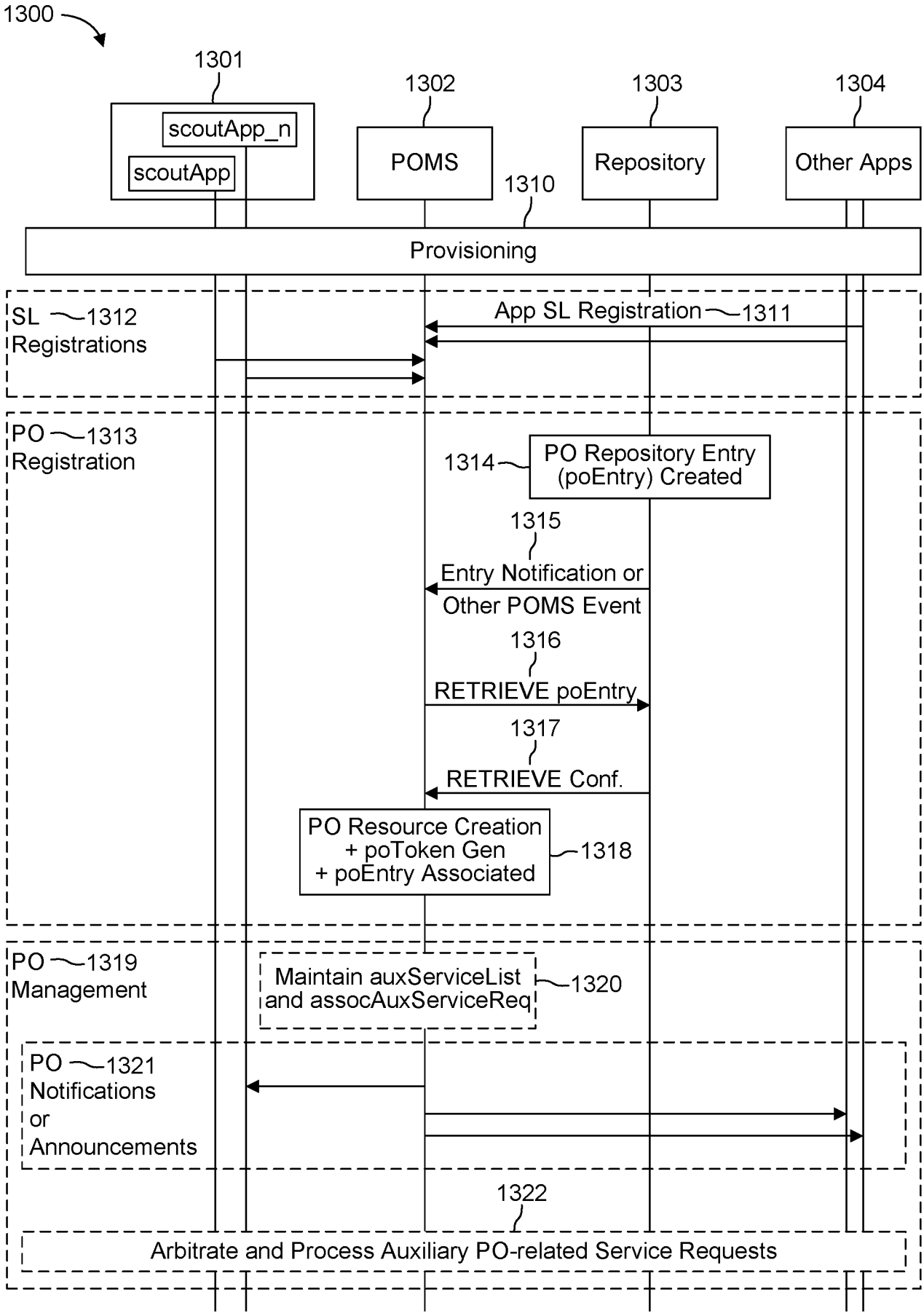
1300
1301
scoutApp_n
scoutApp
1302
POMS
1303
Repository
1304
Other Apps
1310
Provisioning
SL Registrations
1312
App SL Registration
1311
PO Registration
1313
1314
PO Repository Entry (poEntry) Created
1315
Entry Notification or Other POMS Event
1316
RETRIEVE poEntry
1317
RETRIEVE Conf.
PO Resource Creation + poToken Gen + poEntry Associated
1318
PO Management
1319
Maintain auxServiceList and assocAuxServiceReq
1320
PO Notifications or Announcements
1321
1322
Arbitrate and Process Auxiliary PO-related Service Requests

FIG. 13

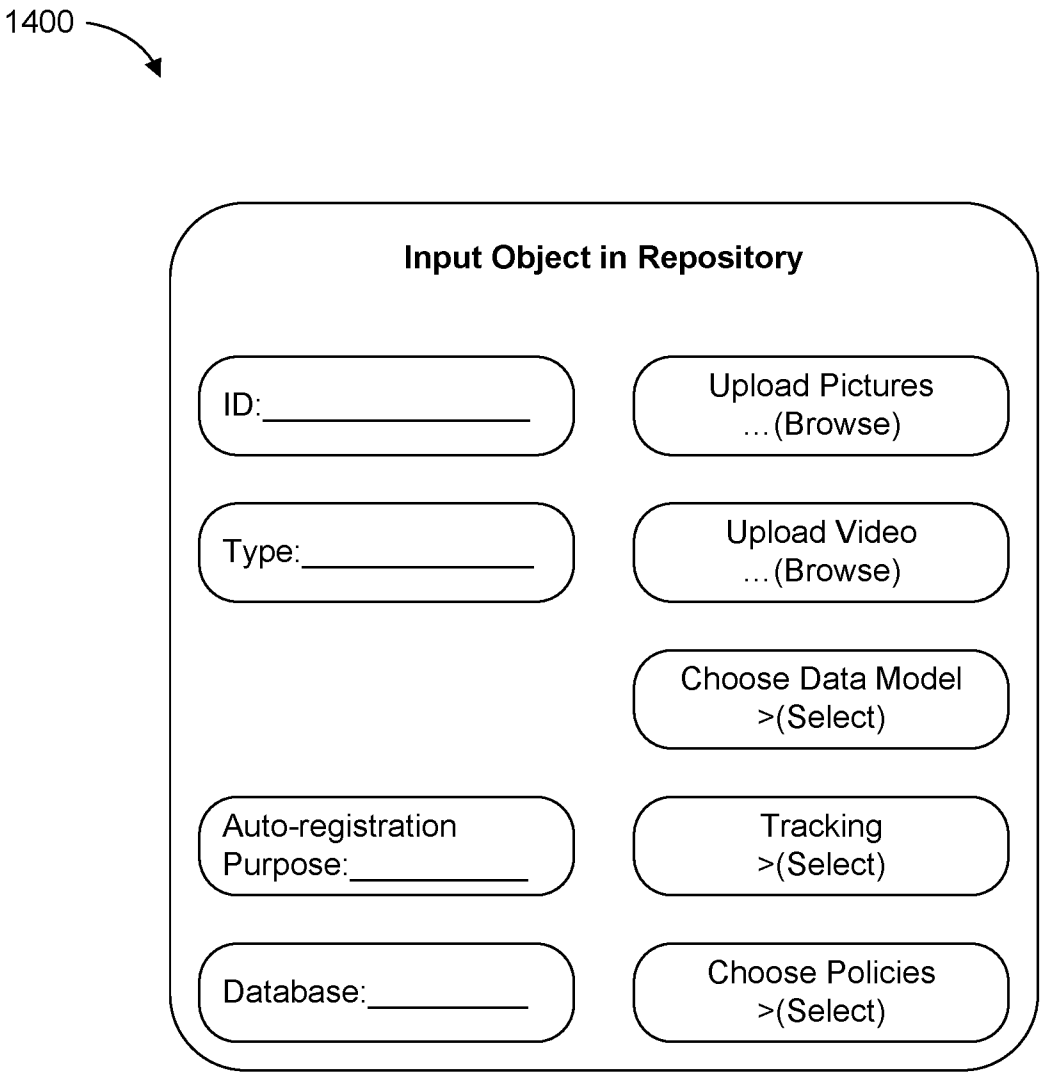
1400
Input Object in Repository
ID:_____
Upload Pictures
…(Browse)
Type:_____
Upload Video
…(Browse)
Choose Data Model
>(Select)
Auto-registration
Purpose:_____
Tracking
>(Select)
Database:_____
Choose Policies
>(Select)

FIG. 14

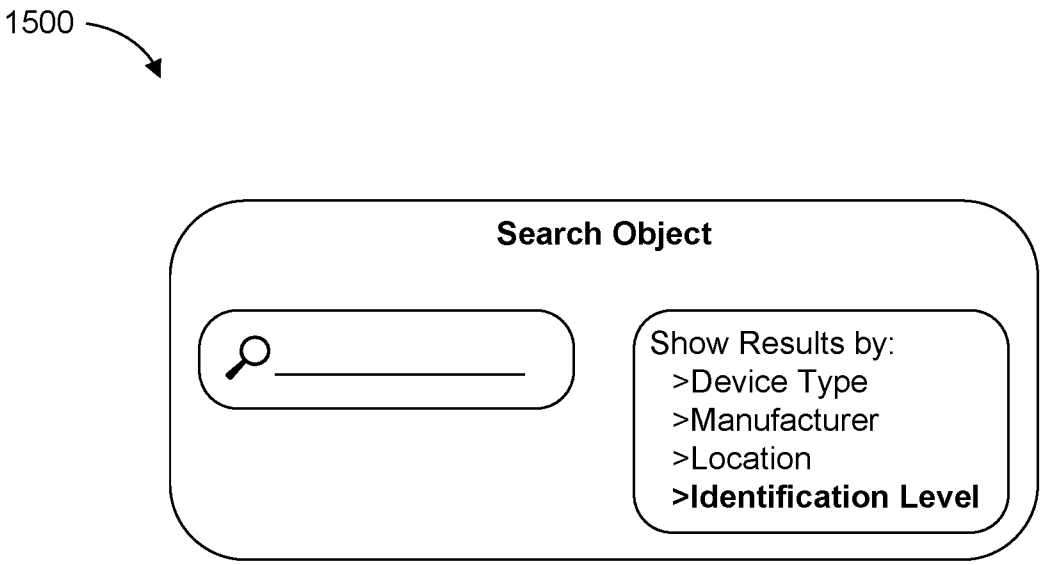
1500
Search Object
Show Results by:
>Device Type
>Manufacturer
>Location
>Identification Level

FIG. 15

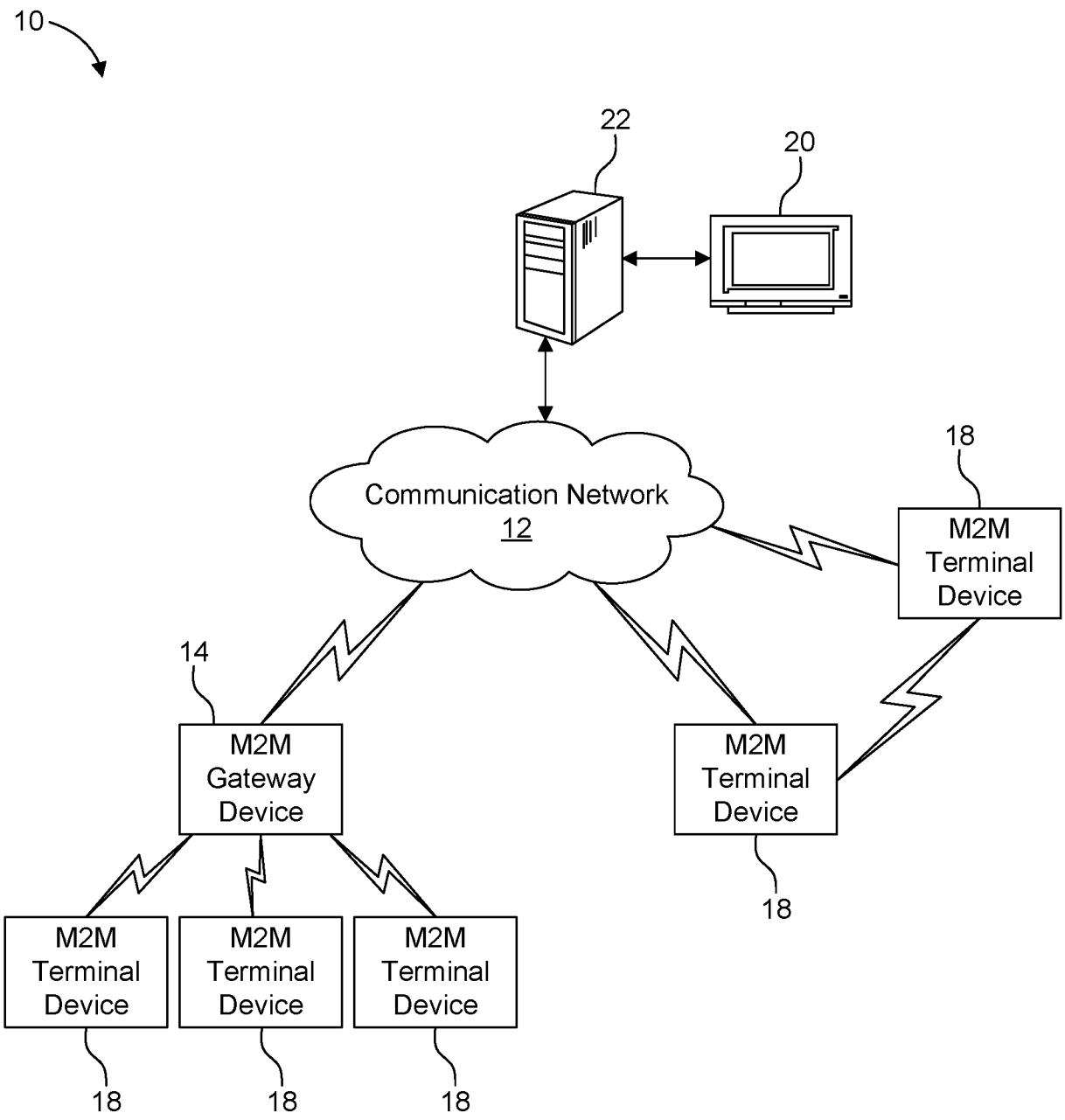
10
22
20
Communication Network
12
18
M2M
Terminal
Device
14
M2M
Gateway
Device
M2M
Terminal
Device
18
M2M
Terminal
Device
M2M
Terminal
Device
M2M
Terminal
Device
18
18
18

FIG. 16A

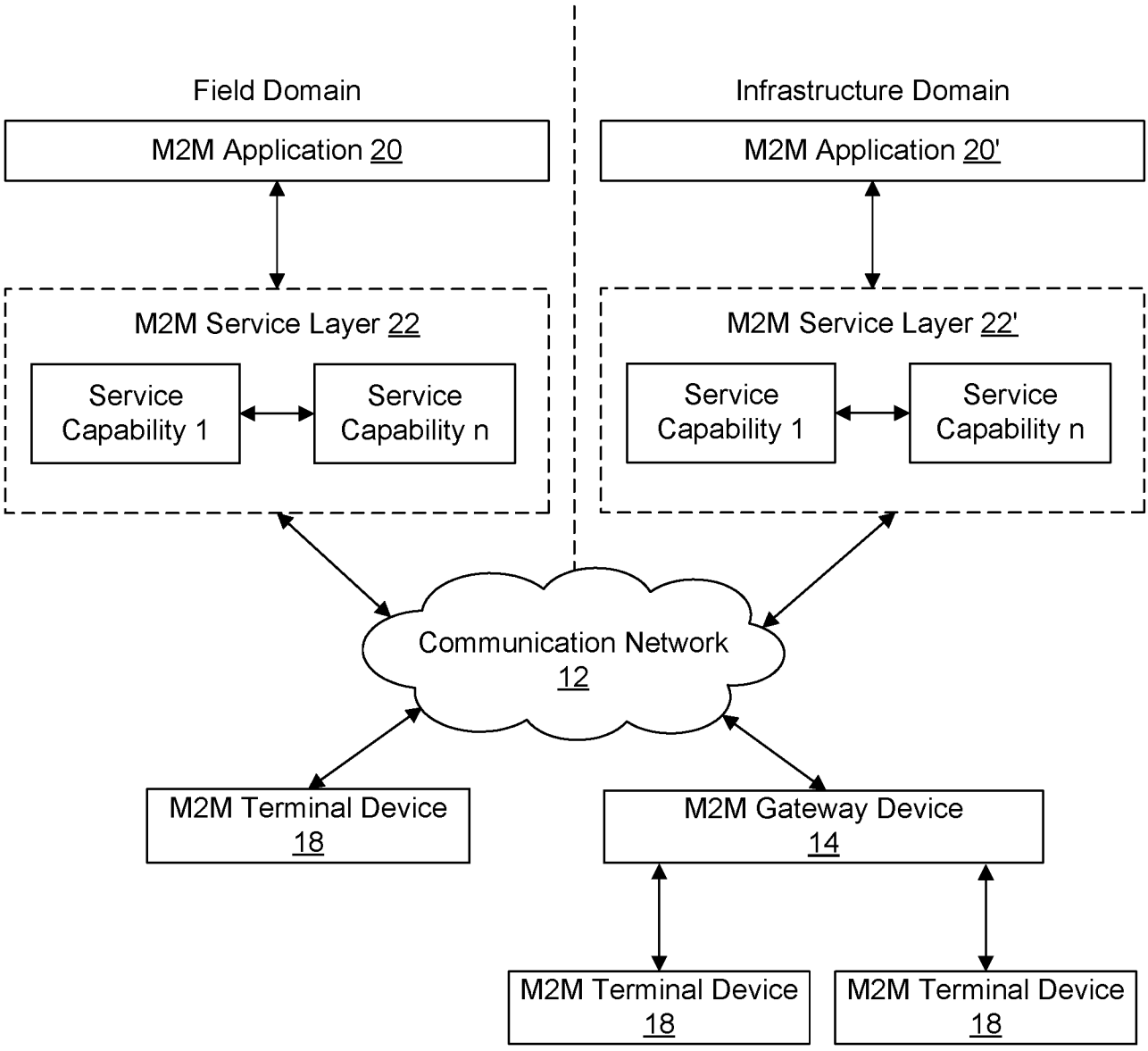
Field Domain
Infrastructure Domain
M2M Application 20
M2M Application 20'
M2M Service Layer 22
M2M Service Layer 22'
Service Capability 1
Service Capability n
Service Capability 1
Service Capability n
Communication Network 12
M2M Terminal Device 18
M2M Gateway Device 14
M2M Terminal Device 18
M2M Terminal Device 18

FIG. 16B

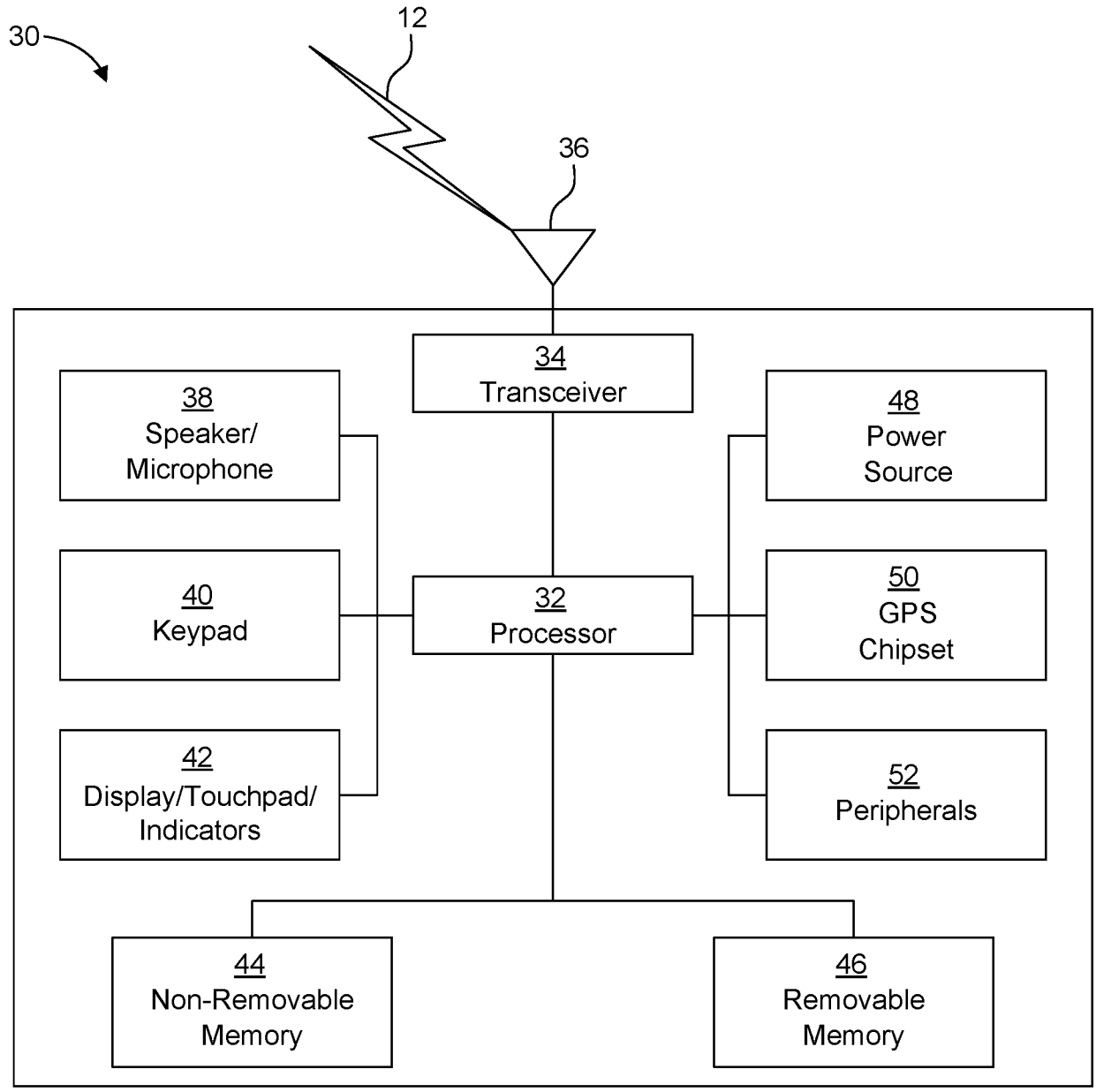
30
12
36
34
Transceiver
38
Speaker/
Microphone
40
Keypad
42
Display/Touchpad/
Indicators
32
Processor
48
Power
Source
50
GPS
Chipset
52
Peripherals
44
Non-Removable
Memory
46
Removable
Memory

FIG. 16C

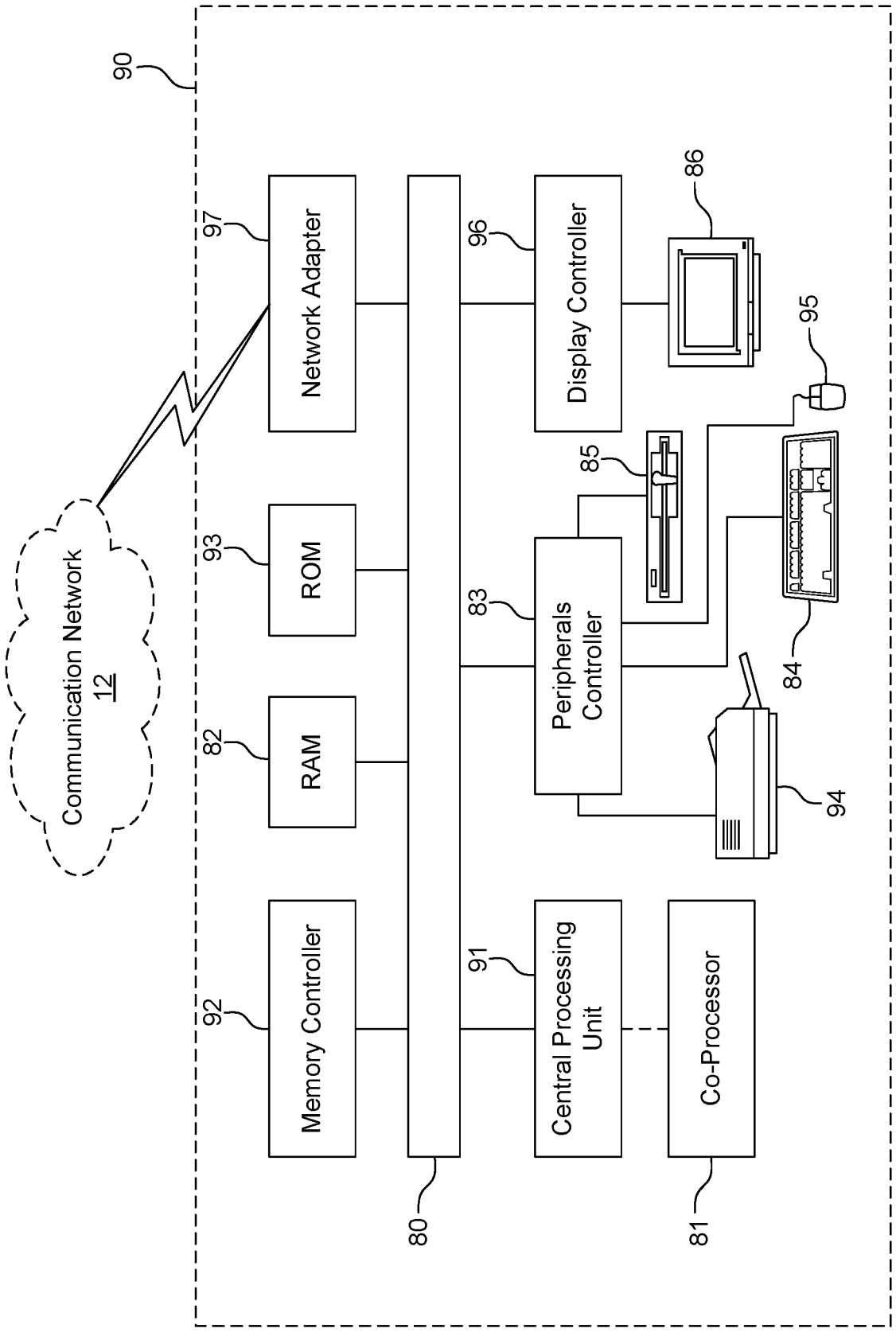
90
Network Adapter
97
ROM
93
RAM
82
Memory Controller
92
Communication Network
12
80
Display Controller
96
86
Peripherals Controller
83
85
95
84
94
Central Processing Unit
91
Co-Processor
81

FIG. 16D

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2018/067624

A. CLASSIFICATION OF SUBJECT MATTER
INV. 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

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.
X	US 2017/163957 A1 (ANDERSON GLEN J [US]) 8 June 2017 (2017-06-08) paragraph [0022] - paragraph [0038] paragraph [0042] - paragraph [0043] paragraph [0048] - paragraph [0049] paragraph [0062] - paragraph [0065] paragraph [0104] - paragraph [0122]	1-24,39, 40
A	EP 2 613 502 A1 (ERICSSON TELEFON AB L M [SE]) 10 July 2013 (2013-07-10) paragraph [0015] - paragraph [0016] paragraph [0053] - paragraph [0066]	1-24,39, 40

☐ Further documents are listed in the continuation of Box C.

☒ 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
2 April 2019	04/06/2019

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	Lázaro, Marisa

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

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2018/067624

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

see additional sheet

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

1-24, 39, 40

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 210

This International Searching Authority found multiple (groups of) inventions in this international application, as follows:

1. claims: 1-24, 39, 40

   Method and apparatus for registration of a passive object

   ---

2. claims: 25-31, 41

   Apparatus and method for determining permissions and parameters associated with a passive object

   ---

3. claims: 32-38, 42

   Apparatus for executing a service request for a passive object

   ---

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2018/067624

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2017163957 A1	08-06-2017	US 2017163957 A1	08-06-2017
		WO 2017095524 A1	08-06-2017
EP 2613502 A1	10-07-2013	CN 101513018 A	19-08-2009
		CN 101513019 A	19-08-2009
		EP 2060092 A1	20-05-2009
		EP 2060094 A1	20-05-2009
		EP 2613502 A1	10-07-2013
		JP 5279711 B2	04-09-2013
		JP 2010503248 A	28-01-2010
		US 2008056207 A1	06-03-2008
		US 2010166003 A1	01-07-2010
		US 2012224587 A1	06-09-2012
		US 2014233426 A1	21-08-2014
		WO 2008028890 A1	13-03-2008
		WO 2008030170 A1	13-03-2008