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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 April 2015 (02.04.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/047438 A1

(51) International Patent Classification:
G06F 21/62 (2013.01) *G06F 17/30* (2006.01)

(21) International Application Number:
PCT/US2013/076712

(22) International Filing Date:
19 December 2013 (19.12.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/884,022 28 September 2013 (28.09.2013) US

(71) Applicant *(for all designated States except US)*: **MCAFEE, INC.** [US/US]; 2821 Mission College Blvd., Santa Clara, California 95054 (US).

(72) Inventor; and
(71) Applicant *(for US only)*: **KRISHNAPRASAD, Vijayasree** [US/US]; 16662 NW Paddington Drive, Beaverton, Oregon 97006 (US).

(74) Agent: **CRANDALL, Sean C.**; Patent Capital Group, c/o CPA Global, P.O. Box 52050, Minneaplis, Minnesota 55402 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, 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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

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

(54) **Title**: REAL-TIME POLICY DISTRIBUTION

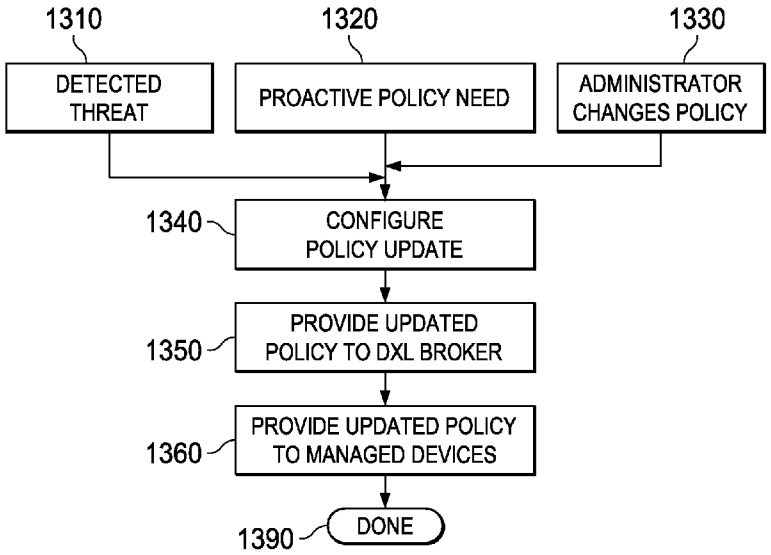


FIG. 13

(57) **Abstract**: In an example, there is disclosed a method and system for real-time policy and task distribution to endpoints over a data exchange layer. According to one embodiment, a persistent point-to-point messaging framework is used to distributed configuration policy and tasks to a distributed, disparate set of devices immediately upon policy definition. Advantageously, the data exchange layer may facilitate delivery of messages even to endpoints that sit, for example, behind a firewall or NAT.

WO 2015/047438 A1

REAL-TIME POLICY DISTRIBUTION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Provisional Application 61/884,022, filed 28 September 2013, entitled "Real-Time Policy Distribution to Unreachable Systems," which is incorporated herein by reference.

FIELD OF THE DISCLOSURE

[0002] This application relates to the field of enterprise security, and more particularly to real-time policy distribution over a data exchange layer.

BACKGROUND

[0003] An enterprise service bus (ESB) is a software-based network architecture that provides a medium of data exchange over a service-oriented architecture. In some embodiments, ESB is a special case of a client-server software architecture in which clients may route messages through the server.

[0004] Software, binaries, executables, advertising, web pages, documents, macros, executable objects, and other data provided to users (collectively "executable objects") may include security flaws and privacy leaks that are subject to exploitation by malware. As used throughout this specification, malicious software ("malware") may include a virus, Trojan, zombie, rootkit, backdoor, worm, spyware, adware, ransomware, dialer, payload, malicious browser helper object, cookie, logger, or similar application or part of an application designed to take a potentially-unwanted action, including by way of non-limiting example data destruction, covert data collection, covert communication, browser hijacking, network proxy hijacking or redirection, covert tracking, data logging, keylogging, excessive or deliberate barriers to removal, contact harvesting, unwanted use of premium services, and unauthorized self-propagation. In some cases, malware may also include legitimate software that includes inadvertent security flaws that cause or enable malware behavior. "Malware behavior" is defined as any behavior that qualifies an application as malware or grayware. Some prior art systems are configured to identify and block malware, for example by maintaining databases of known malware.

[0005] In addition to executable objects, computing devices may encounter static objects, which are not intended to change the operating state of a computer. As a class, executable objects and static objects may be referred to simply as "objects." An enterprise security concern is the classification of objects' malware status.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The present disclosure is best understood from the following detailed description when read with the accompanying FIGUREs. It is emphasized that, in accordance with the standard practice in the industry, various features are not drawn to scale and are used for illustration purposes only. In fact, the dimensions of the various features may be arbitrarily increased or reduced for clarity of discussion.

[0007] FIGURE 1 is a network diagram of a context-aware network with DXL capability according to one or more examples of the present specification.

[0008] FIGURE 1A is an example where a domain master is a joint threat intelligence (JTI) server according to one or more examples of the present specification.

[0009] FIGURE 1B is a network diagram of select elements and a context-aware network according to one or more examples of the present specification.

[0010] FIGURE 2 is a network diagram disclosing a distributed architecture according to one or more examples of the present specification.

[0011] FIGURE 3 is a network diagram of an example DXL network operating across a traditional enterprise boundary according to one or more examples of the present specification.

[0012] FIGURE 4 is a network diagram of a context-aware network 400 according to one or more examples of the present specification.

[0013] FIGURE 5 is a block diagram of a DXL broker according to one or more examples of the present specification.

[0014] FIGURE 6 is a block diagram of a CIM server according to one or more examples of the present specification.

[0015] FIGURE 7 is a block diagram of a domain master 160 according to one or more examples of the present specification.

[0016] FIGURE 8 is a block diagram of a client according to one or more examples of the present specification.

[0017] FIGURE 9 is a flow diagram illustrating evaluation of an object in a hierarchical manner according to one or more examples of the present specification.

[0018] FIGURE 10 is a flow diagram of a method of performed by an client 120 according to one or more examples of the present specification.

[0019] FIGURE 10A is a flow diagram of a method performed by a JTI server in connection with the method of FIGURE 10 according to one or more example embodiments of the present specification.

[0020] FIGURE 11 is a flow diagram of a method of a JTI server servicing a reputation request according to one or more examples of the present specification.

[0021] FIGURE 12 is a flow diagram of an enhancement to the method of FIGURE 10A according to one or more examples of the present specification.

[0022] FIGURE 13 is a flow diagram of a method performed by a DXL master according to one or more examples of the present specification.

[0023] FIGURE 14 is a flow diagram of a method performed by a DXL broker in response to a command-level message containing a security policy configuration update, according to one or more examples of the present specification.

[0024] FIGURE 15 is a flow diagram of a method performed by a client responsive to receiving a security policy update, either from domain master or DXL broker according to one or more examples of the present specification.

DETAILED DESCRIPTION OF THE EMBODIMENTS

OVERVIEW

[0025] In an example, there is disclosed a method and system for real-time policy and task distribution to endpoints over a data exchange layer. According to one embodiment, a persistent point-to-point messaging framework is used to distributed configuration policy and tasks to a distributed, disparate set of devices immediately upon policy definition. Advantageously, the data exchange layer may facilitate delivery of messages even to endpoints that sit, for example, behind a firewall or NAT.

EXAMPLES OF THE DISCLOSURE

[0026] The following disclosure provides many different embodiments, or examples, for implementing different features of the present disclosure. Specific examples of components and arrangements are described below to simplify the present disclosure. These are, of course, merely examples and are not intended to be limiting. Further, the present disclosure may repeat reference numerals and/or letters in the various examples. This repetition is for the purpose of simplicity and clarity and does not in itself dictate a relationship between the various embodiments and/or configurations discussed.

[0027] Different embodiment many have different advantages, and no particular advantage is necessarily required of any embodiment.

[0028] In an increasingly heterogeneous software ecosystem, enterprises may face new and enhanced security challenges and malware threats. This creates a situation in which real-time exchange of threat intelligence between otherwise autonomous network elements is desirable. Increased sharing may improve security between devices that otherwise operate in their own security "silos."

[0029] The system and method of the present specification addresses such challenges by providing standardized data representations across data sources, and safeguarding the quality of data shared by disparate sources.

[0030] Context-Aware Computing (CAC) is a style of computing in which situational and environmental information about people, places and things is used to anticipate immediate needs and proactively offer enriched, situation-aware, and usable functions and experiences. Context-aware computing relies on capturing data about the world as it is at the moment the system is running.

[0031] According to one or more examples of the present specification, a "context-aware network" is an adaptive system, including for example a security system, of interconnected services that communicate and share information to make real-time, accurate decisions by individual products and/or as a collective. According to an example, network, endpoint, database, application and other security solutions are no longer to operate as separate "silos" but as one synchronized, real-time, context-aware and adaptive, security system.

[0032] In an example, multiple network elements are connected to one another via a data exchange layer (DXL), which is a type of ESB that is suitable for exchange of security-related messages among other things. As used herein, "network elements" include any type of client or server (e.g., a video server, a web server, etc.), routers, switches, gateways, bridges, load-balancers, firewalls, inline service nodes, proxies, network appliances, processors, modules, or any other suitable device, component, element, or object operable to exchange information in a network environment. More specifically, DXL endpoints are network elements that interact over a DXL ESB. DXL endpoints may be distributed across a customer network and communicate in "real-time" in a trusted, secure, and reliable fashion. This may provide increased automation and improved security services.

[0033] In an example, DXL endpoints are deployed at strategic locations within a network to intercept ongoing business activity, inspect and interpret it, and ultimately determine whether it is authorized, meaning for example that it is consistent with enterprise security policies. In some cases, network elements must such decisions "in-band," momentarily suspending the business activity, and in "machine-real-time," at latencies low enough to avoid a significant user-perceptible delay in the business activity.

[0034] In some cases, network elements may have independent access to security data only by way of their own independent analysis and observation, and via scheduled definition updates, which may come, for example, on a weekly basis as updated malware definition.

[0035] Because network elements are often heterogeneous and may be deployed, particularly in a modern network, in a temporary or *ad hoc* fashion, real-time intelligence becomes a challenge, particularly when "in-band" decisions are necessary. Furthermore, an enterprise may procure security solutions in a piecemeal fashion, so that one product cannot always assume the presence of another product. For example, there may not be a single, pre-defined repository of threat intelligence for network elements to consult, and regular malware definition updates may not include lately discovered threats. Representation and interpretation of data offer yet another challenge. Network elements may use disparate, proprietary data representations. Thus, for example, even an antivirus scanner may not be configured to share newly-discovered malware information with a network-based security device. Trustworthiness of information may be yet another challenge in other contexts. In

other words, even if an antivirus scanner and network-based security device are configured to share security intelligence, each may not have a means of validating intelligence received from the other.

[0036] In an example, the present specification provides a data exchange layer (DXL), which may operate on a lightweight messaging-based communications infrastructure such as ESB and be configured to allow endpoints to share contextual data. DXL may be one element of a larger security-connected framework, which is an adaptive system, such as a security system, of interconnected services that communicate and share information to make real-time, accurate security decisions by individual security products and/or as a collective. According to an example, network, endpoint, database, application and other security solutions need not operate as separate 'silos' but as one synchronized, real-time, context-aware and adaptive security system.

[0037] FIGURE 1 is a network diagram of a context-aware network 100 with DXL capability. According to this example, a plurality of clients 120 connected to a DXL enterprise service bus (ESB) 130. DXL ESB 130 is an example of a DXL fabric, and may be provided on top of an existing network, such as a local area network (LAN). Client 120 may be any suitable computing device, including by way of non-limiting example a computer, a personal digital assistant (PDA), a laptop or electronic notebook, a cellular telephone, an IP telephone, an iPhone™, an iPad™, a Microsoft Surface™, an Android™ phone, a Google Nexus™, or any other device, component, element, or object capable of initiating voice, audio, or data exchanges within a communication system, including a suitable interface to an end user, such as a microphone, a display, or a keyboard or other terminal equipment. In the example of FIGURE 1, client 120-1 is an embedded device, such as a network security sensor. Client 120-2 is a virtual machine. Client 120-3 is a laptop or notebook computer. Client 120-4 is a desktop computer.

[0038] DXL ESB 130 maybe any type of physical or virtual network connection over which appropriate data may pass. At present, there is no fixed or global standard for an ESB, and the term, as used herein, is intended broadly to encompass any network technology or topology suitable for message exchange. In one embodiment, message queuing telemetry transport (MQTT) messages are exchanged on port 8883. In some cases, clients 120, DXL broker 110, domain master 160, database 162, JTI server (FIGURE 1A), proxy 170 (FIGURE 1A),

and threat intelligence service 180 (FIGURE 1A), all by way of non-limiting example, may be referred to as "network elements."

[0039] Network elements configured to operate on or with DXL ESB 130 may be referred to as "DXL endpoints." These may include, in an example, clients 120, DXL broker 110, and domain master 160.

[0040] DXL broker 110 may be configured to provide DXL messaging services over DXL ESB 130, such as maintaining DXL routing tables and delivering messages.

[0041] DXL broker 110 provides DXL services 190, which in an example are network services operable to provide DXL ESB 130 to DXL endpoints.

[0042] Domain master 160 may be configured to communicatively couple to DXL broker 130. Domain master 160 may maintain domain data in a database such as database 162. In this example, domain master 160 and database 162 are shown as two distinct entities, but it should be noted that many configurations are possible. For example, database 162 may reside on a disk drive local to the domain master 160, or may be separately or remotely hosted. Database 162 is disclosed by way of example, and may be any suitable data store, including a structured or relational database, distributed database, or flat file by way of non-limiting example.

[0043] By way of operative example, a client such as laptop 120-3 connects to the LAN and receives a new IP address. At this point, several properties of laptop 120-3 become knowable to other network elements, including by way of non-limiting example, its , IP address, information about its operating system, and the logged-on user's username. For ease of reference, these are referred to as "client properties" throughout this example. Client properties are embodiments of security-intelligence data, and are of interest to virtual machine 120-2, which has previously subscribed the security-intelligence data topic with domain master 160.

[0044] Client properties may be reported to the DXL simultaneously by two distinct sources, namely by laptop 120-3, and by network security sensor 120-1. However, network security sensor 120-1 may fail to report a username value. It may also report an OS value different from that reported by the laptop 120-3. This may be, for example, because network security sensor 120-1 is sensing data remotely, and may not be able to determine these values as reliably as laptop 120-3 itself.

[0045] Domain master 160 is responsible for the "client system" data domain, which includes client properties. When laptop 120-3 and client 120-1 publish a message containing the client properties, both messages are first routed to DXL broker 110. DXL broker 110 may then forward the client properties to domain master 160.

[0046] Domain master 160 may combine and reconcile the client properties received from the two sources into a single record of truth, containing a single value for IP address, operating system, and username respectively. Specifically, it may determine via its own logic and perhaps prior configuration that laptop 120-3 is trusted more for OS value than network security sensor 120-1. Therefore, domain master 160 may ignore the "operating system" value received from network security sensor 120-1 when it conflicts with laptop 120-3.

[0047] The reconciled client properties are persistently stored in domain database 162. Domain master 160 may then publish the client properties on DXL ESB 130. DXL broker 110 may then forward the published message to virtual machine 120-2, which receives a singular and most accurate value for client properties.

[0048] Subsequently, client 120-4 may send a DXL request over DXL ESB 130, inquiring about the client properties for laptop 120-3. DXL broker 110 receives this request and automatically routes it to domain master 160. Domain master 160 retrieves the client properties from domain database 162 and sends a DXL response message, which DXL broker 110 receives and forwards to client 120-4. Note that while the "publish-subscribe" transactions in this example are one-to-many, the "request-response" transactions are one-to-one.

[0049] In some embodiments, DXL may be characterized by messaging that allows loose integration or coupling of multiple network elements. Loose coupling may reduce the assumptions each DXL endpoint must make about other DXL endpoints, such as the presence of certain capabilities, hardware, or software. According to one or more examples of the present specification, DXL is a 'Plug-and-Play' API, and may facilitate context-aware and adaptive security by enabling context to be shared between products.

[0050] Further according to one or more examples of the present specification, DXL is an elastic architecture with multiple deployment options and is highly scalable. DXL may also be designed with openness in mind and enable third-party integration.

[0051] DXL ESB 130 may be based on a two-layer protocol. The "bottom" layer is a secure, reliable, low-latency data transport fabric that connects diverse security elements as a mesh, or in a hub-and-spoke configuration. The "top" layer is an extensible data exchange framework that is configured to facilitate trustworthy data representation.

[0052] In an example, DXL endpoints connect to DXL ESB 130. Each DXL endpoint is assigned a distinct identity, and authenticates itself to DXL ESB 130 upon startup, for example via a certificate or other secure token. DXL endpoints may establish one-to-one communications via DXL ESB 130, for example by sending a DXL message addressed to a DXL endpoint with a specific identity. This enables DXL endpoints to communicate with each other without having to establish a point-to-point network connection. In an example, this is analogous to a person-to-person phone call.

[0053] In another example, DXL may provide a publish-subscribe framework in which certain DXL endpoints "subscribe" to messages of a certain type. When a DXL endpoint "publishes" a message of that type on DXL ESB 130, all subscribers may process the message, while non-subscribers may safely ignore it. In an example, this is analogous to a podcast subscription service. In yet another example, DXL may provide a request-response framework. In this case, one DXL endpoint may publish a request over DXL ESB 130. An appropriate DXL endpoint receiving the request may provide a response. Advantageously, the response may be used by more than just the DXL endpoint that originally published the request. For example, if a client 120 publishes a request for an object's reputation, JTI server 150 may respond by publishing the reputation. Thus, other clients 120 that find instances of the object may benefit from the response. For example, clients 120 may maintain a comprehensive cache of reputations published on the network. If a client 120 then newly encounters an object that is known on the network, client 120 already has an up-to-date reputation for the object.

[0054] DXL ESB 130 may be implemented using diverse software elements, patterns, and constructs suited to the specific infrastructure connecting the security elements. For instance, in a physical enterprise network, messaging middleware consisting of multiple interconnected message brokers may be deployed, where endpoints connect to the closest broker. In a virtual network infrastructure, the fabric may leverage hypervisor provided channels.

[0055] As noted above, DXL ESB 130 may be configured to provide real-time, trusted exchange of data among otherwise-autonomous, dynamically-assembled DXL endpoints. Thus, in an example DXL ESB 130's conceptual framework may comprise two virtual components:

a. Broad collections of security-relevant data are categorized into "data domains." Each data domain is a closely related sub-collection of entities, attributes, and inter-relationships.

b. Domain master 160 is a data provider assigned ownership of data for each domain. Domain master 160 acts as an intermediate trusted data broker between first-hand sources of raw "intelligence" data, and data consumer endpoints such as clients 120. Intelligence data may flow from data producer endpoints to the appropriate domain master 160, and then be relayed to data consumer endpoints such as clients 120. Note that in this example, the concepts of "data producer" and "data consumer" are contextual roles, and not necessarily physical devices. A client 120 may be a data producer in one context and a data consumer in another context.

[0056] In an example, domain master 160 may establish first-hand trust relationships with data-provider endpoints. This enables it to gauge the quality (including accuracy and reliability) of data (such as reputation data) it receives from any particular source. When duplicate, piecemeal data is received from multiple (independent) sources, such as different clients 120, domain master 160 may reconcile the data and resolve conflicts to derive a single best-known record of truth (such as, for example, a reputation) for each object. This ensures that clients 120 receive consistent data.

[0057] Domain master 160 may also transform data into a well-understood, standardized representation. This representation may be published on DXL ESB 130, so that all clients 120 receive usable data.

[0058] Advantageously, DXL endpoints do not need to know what device originated data, or make point-to-point connections to other DXL endpoints, even when one-to-one communication is necessary. Rather, DXL client software or DXL extensions enable a DXL endpoint to use its own local APIs for querying and receiving data. To increase network efficiently, DXL endpoints may cache received data locally, which data may be trusted until it

is superseded by an authorized DXL message. For example, clients 120 may subscribe to published reputations for objects. When an object reputation is received, either in response to a request-response transaction, or in a publish-subscribe model, client 120 may store the reputation in a local database. The reputation may be trusted until superseded, because DXL master 160 is configured to publish a reputation update whenever it receives an updated reputation. Thus, frequent individual data requests from clients 120 become bulk data subscriptions, as published reputations are available to all clients 120 that subscribe to reputations. Advantageously, this may reduce the latency of data exchanges.

[0059] In yet another example, DXL broker 110 provides a discovery and location service that informs DXL endpoints of the specific domain master 160 to which data query and subscription requests should be routed.

[0060] Advantageously, the example DXL architecture described herein is flexible. For example, individual data sources may connect and disconnect from the network without affecting data consumers. Domain master 160 may simply rely on whatever data sources are available. Furthermore, the framework makes no assumptions about the physical location or specifically how domain master 160 or domain endpoints are deployed or configured. So long as each network element provides valid DXL messaging, traffic is routed correctly.

[0061] In the example above, DXL master 160 is a logical singleton, but it should be noted that DXL master 160 may be implemented, for example, as a set of distributed service components, where each component services either a subset of the domain, or provides a local data replica of a service running elsewhere. Such configurations may enhance scale, performance, and reliability. This may also allow services to be transparently relocated.

[0062] Further advantageously, the DXL framework provided herein is extensible. For example, data about new entities and relationships simply by creating new data domains. New attributes and relationships for existing data domains may be provided simply by defining new message types for that domain.

[0063] In an example, domain master 160 has responsibility over the domain of malware data. To distinguish messages from the “malware” domain, such as network status and device maintenance, a namespace may be defined for each. For example, the reputation domain may use the “MALWARE” namespace, the network status domain may use the “STATUS” namespace, and the device maintenance domain may use the “MAINT”

namespace. Thus, but if domain master 160 is the master for the reputation domain, it knows to process messages within the malware namespace, and to ignore all others. This allows a designer for each domain to assign a set of messages without having to consult existing domains to avoid name collisions for messages.

[0064] For example, both the reputation domain and a device maintenance domain may have use for a message such as DOWNLOAD_UPDATES. In the case of the reputation domain, this message maybe an instruction to retrieve updated now are definitions from JTI server 150. In the device maintenance domain, this may be an instruction to download operating system updates from a vendor.

[0065] Clients 120 may be configured to exchange data from several DXL domains and may subscribe to messages on both the reputation domain and the device maintenance domain. Thus, client 120-1 for example, to parse and respond to a DXL message DOWNLOAD_UPDATES by requesting bulk reputation updates. In one embodiment, the message requesting malware updates may itself be a DXL message. In some cases, for example where the updates are large and are not needed in real-time, delivery of the updates may be completed outside of the DXL architecture to reserve a DXL for lightweight, high-speed messaging.

[0066] Client 120 may also know that it should parse and respond to MAINT: DOWNLOAD_UPDATES by contacting a vendor's servers and requesting updates.

[0067] In the case where domain master 160 is configured as the master for the reputation domain, it may know to ignore all DXL messages that are not in the MALWARE namespace. Note, however that a single physical device may be configured to act as a domain master for multiple domains, in which case traffic in different namespaces may be passed two different subroutines. In some embodiments, DXL broker 110 may be configured to synthesize reports from a plurality of DXL network devices such as clients 120 that are given lesser privileges, such as "suggest" privileges on DXL ESB 130.

[0068] Further adding to extensibility, new or better data sources may be incorporated, by integrating them with domain master 160. This may be completely transparent to clients 120 and other DXL endpoints.

[0069] Additional features of a DXL broker 110 may include, by way of non-limiting example: service and location registries to lookup registered endpoints, available services,

and their locations; publish/subscribe (1:N), request/response (1:1), device-to-device (1:1), and push notification messaging interfaces; optimized message passing between brokers; destination-aware message routing; and broker-to-broker failover.

[0070] Advantageously, domain master 160 need not be concerned with how each DXL endpoint treats a published message. Rather, that can be a matter of enterprise security policy.

[0071] Additional DXL features for clients 120 may include, by way of non-limiting example: local message bus integration an API to discover brokers, authenticate to DXL, and send and receive catalogued messages.

[0072] Additional general features of context-aware network 100 may include, by way of non-limiting example: DXL broker and client provisioning and management of domain master 160; policy-based authorization of endpoints onto DXL ESB 130; secure SSL-based communications; proxy support for off-premises communications; and domain master appliances pre-configured with DXL broker functionality (thus joining domain master 160 and DXL broker 110 into one device).

[0073] FIGURE 1A is an example where domain master 160 is a joint threat intelligence (JTI) server 150, providing for example, object reputation services on the "reputation" domain according to one or more examples of the present specification. JTI server 150 may communicatively couple to DXL broker 110. JTI server 150 may be a middleware appliance configured to provide reputation services, maintain metadata about network objects (such as reputation, prevalence, and intelligence, by way of non-limiting example), call out to external scanners for an object's reputation classification, and provide telemetry data to threat intelligence service 180. JTI server 150 may communicate with a global threat intelligence service 180 via a proxy 170, which may include either communication over DXL ESB 130, or over a more traditional IP network.

[0074] Advantageously, JTI server 150, in connection with threat intelligence service 180, may provide crowd-sourced object reputations. JTI server 150 may also provide administrator overrides. These may include aggregation of administrator override policies, from several enterprises, for whether an object should be run on a network and whether a certificate is considered "clean." These may also include client-side results, such as an aggregate of end-user decisions of whether to allow or block an object.

[0075] In another example, JTI server 150 may track prevalence data, including the prevalence of an object on context-aware network 100.

[0076] In yet another example, JTI server 150 may serve as a telemetry aggregator/proxy to gather and send to threat intelligence service 180 telemetry from endpoints that do not directly interact with threat intelligence service 180. JTI server 150 may also contribute file metadata to threat intelligence service 180, such as hashes, digital signature data, and file attributes.

[0077] JTI server 150 may receive several such messages from different clients 120, and in some cases, the messages may conflict with one another. When the upon receiving an initial report from an client 120 identifying an object as malware, domain master 160 may publish a warning that the other clients should give additional scrutiny to the object, such as deep scanning or requesting user verification before executing the object. In other embodiments, domain master 160 may provide such information in response to a request for an object reputation from a client 120.

[0078] If additional clients 120 identify the object as malware, then a confidence indicator may pass a threshold. JTI server 150 may then publish a higher-level, such as command-level, message identifying the object as malware. If JTI server 150 receives multiple conflicting reports from different clients, and a synthesis algorithm may provide an appropriate action. For example, if one or a few clients are found to be outliers, reporting a status different from a large majority of other clients, the outliers may be discarded. This synthesis algorithm may also account for specific clients' past reputations for accurate reporting, or for a reputation assigned based on installed hardware or software.

[0079] In an example where domain master 160 is a JTI server 150, this example illustrates the value of designating a hierarchy or other schema for assigning DXL endpoint privileges. For example, with respect to designating an object as malware, clients 120 may have “suggest” privileges. This may mean, for example, that clients 120 may identify an object that they believe to be malware, and may either direct a specific message to JTI server 150, or may publish a general message identifying the object as malware. Because clients have only “suggest” privileges on DXL ESB 130, other DXL endpoints that subscribe to reputation updates may treat the identification as less authoritative than an identification from a DXL network element with elevated privileges such as “assign” privileges. Privileges, and

especially elevated privileges such as "assign" privileges, may be authenticated by a secure certificate provided as part of a DXL message.

[0080] FIGURE 1B is a network diagram of select elements and a context-aware network 100 according to one or more examples of the present specification. As seen in FIGURE 1B, an additional DXL broker 110-2 may be added to service endpoints 120-3 and 120-4. DXL broker 110-2 may communicate with a second domain master 160-2, which may share domain database 162 with domain master 160-1.

[0081] FIGURE 2 is a network diagram disclosing a distributed architecture according to one or more examples of the present specification. In this example, DXL broker 110-1 may be designated as the "hub," while DXL broker is 110-2, 110-3, 110-4, and 110-5 may be designated as "spokes." in an example, all DXL traffic that passes through spoke will be forwarded to the hub, which will distribute the traffic to other spokes. Designation of a DXL broker 110 as the hub may be accomplished via any suitable means, such as selecting the hub based on MAC ID, IP address, or network proximity to domain master 160.

[0082] If DXL broker 110-1 goes offline, another hub may be at least temporarily needed. In that case, another have may be elected. When DXL broker 110-1 comes back online, it may resume its duties as a hub, or may act as a spoke, depending on network topology and design considerations.

[0083] In another example, spokes may form a temporary mesh network upon using effectively to DXL broker 110-1. In yet other embodiments, DXL broker is 110 may be configured to operate full time in a mesh configuration.

[0084] Additional extensibility may be provided by bridging DXL ESB 130 across disparate networks, enabling data to be exchanged over larger networks, including the internet. FIGURE 3 is a network diagram of an example DXL network operating across a traditional enterprise boundary according to one or more examples of the present specification. In this example, first enterprise 302 includes a DXL broker 110-2 which communicatively couples to a domain master 160 and enterprise switch master (ESM) 330. In one embodiment, domain master 160 and ESM 330 may be coupled over a traditional (non-DXL) network interface 312. Domain master 160 may connect to an IP Network 310. IP Network 310 may be any communicative platform operable to exchange data or information emanating from endpoints, including by way of non-limiting example, an internet architecture

providing end users with the ability to electronically interact, a plain old telephone system (POTS), which end users could use to perform transactions in which they may be assisted by human operators or in which they may manually key data into a telephone or other suitable electronic equipment, any packet data network (PDN) offering a communications interface or exchange between any two nodes in a system, or any local area network (LAN), metropolitan area network (MAN), wide area network (WAN), wireless local area network (WLAN), virtual private network (VPN), intranet, or any other appropriate architecture or system that facilitates communications in a network or telephonic environment..

[0085] Second enterprise 304 also includes a DXL broker 110-1. DXL brokers 110 may communicate with each other over DXL ESB 130. In this example, DXL ESB 130 is physically provided by IP Network 310, but DXL traffic may be distinguished from other types of Internet traffic such as http and other user-centric network traffic, for example because it is provided on a different port or protocol.

[0086] DXL broker 110-1 may also be coupled, for example, to a network service provider (NSP) 340, antivirus agent 350, enterprise firewall 360, and advanced threat detection appliance (ATD) 370.

[0087] ATD 370 may be a dedicated appliance, or a general-purpose computing machine running advanced that detection software. In one example, ATD 370 is configured to analyze objects without existing reputations and to assign those objects threat levels based on the analysis.

[0088] As this figure demonstrates, DXL ESB 130 may be used to integrate heterogeneous or otherwise unrelated network architectures, even across different enterprises. In an example, second enterprise 304 may be a third-party JTI service provider delivering ATD services to first enterprise 302.

[0089] FIGURE 4 is a network diagram of a context-aware network 400 according to one or more examples of the present specification. In an example, context-aware network 400 is substantially similar to context-aware 100. However, in context-aware network 400 domain master 160 is a common information model (CIM) server 410.

[0090] According to one or more examples of the present specification, CIM is open and extensible logical database schema designed to host different types of situational and environmental information. CIM may provide robust representation of new objects, building

blocks, and data types. Core entities of CIM include, by way of non-limiting example assets, identities, applications, and locations.

[0091] CIM may provide many-to-many relationships and building blocks that drive multiple use cases. CIM may also support advanced data visualization that. Advantageously, according to one or more examples of the present specification, CIM scales massively and is open and extensible, allowing different product groups and third parties to develop new data extensions.

[0092] In an example of a CIM use case, a "location" represents a collection of network elements forming a logical location mapped into a physical location or a place. The location context can be used to, by way of non-limiting example, gather complete understanding with regards to the true scope and size of an organization (i.e. in the form of a network topology map).

[0093] CIM may also maintain the global authoritative state of situational and environmental information by synthesizing contextual information shared by multiple data sources over DXL ESB 130.

[0094] FIGURE 5 is a block diagram of a DXL broker according to one or more examples of the present specification. In some embodiments, DXL broker 110, DXL services 190, and JTI server 150 may be provided in a single physical computing device.

[0095] In an example, DXL broker 110 is controlled by a processor 510. Processor 510 may connect to other system elements via system bus 570. Those other elements may include, by way of non-limiting example, a memory 520, network interface 540, and storage 550.

[0096] Processor 510 is configured to control DXL broker 110, for example via executable software or firmware instructions. A "processor" as used herein includes any combination of hardware, software, or firmware providing programmable logic, including by way of non-limiting example a microprocessor, digital signal processor, field-programmable gate array, programmable logic array, application-specific integrated circuit, or virtual machine processor.

[0097] Memory 520 may be a relatively low-latency volatile memory in some embodiments, and may include main memory, cache, on-chip memory, L1 memory, L2 memory, or similar. Note that in this embodiment, processor 510 is depicted in a direct

memory access arrangement with memory 520, but in other embodiment, memory 520 may communicate with processor 510 via system bus 570, via some other bus, or via some other means. Furthermore, although memory 520 and storage 550 are depicted in this example as physically or conceptually separate devices, it should be appreciated that in some embodiments, memory 520 and storage 550 may share a physical device, which may or may not be divided into separate memory areas. Thus, it should be appreciated that the arrangement disclosed here is an example only, and not limiting. Rather, it is expressly intended that even where a memory and storage are spoken of separately, they may be embodied in a single physical or logical device unless expressly stated otherwise.

[0098] In this example, network interface 540 provides a physical and logical interface to DXL ESB 130, and includes any communication medium, whether analog, digital, or mixed-signal, that is configured to communicatively couple client 120 to other computing devices. Network interface 540 may include, by way of non-limiting example, a WiFi, ethernet, Firewire, fiberoptic, USB, serial interface, infrared, cellular network, digital PCS networks, 2G data network, 3G data network, 4G WiMAX, or 4G LTE data networks. In some embodiments, network interface 540 may also provide a physical and logical interface to IP Network 310.

[0099] Storage 550 is disclosed as an example of a non-volatile memory medium, which may be a species of memory 520. In some embodiments, memory 520 and storage 550 may be separate devices, with memory 520 being a relatively low-latency volatile memory device, and storage 550 being a relatively high-latency non-volatile memory device. Storage 550 may also be another device, such as a hard drive, solid-state drive, external storage, redundant array of independent disks (RAID), network-attached storage, optical storage, tape drive, backup system, or any combination of the foregoing. Many other configurations are also possible, and are intended to be encompassed within the broad scope of this specification.

[0100] In an example, memory 520 includes DXL broker 522 and DXL services software 526. DXL broker softer 522 provides DXL broker services as described herein. DXL services software 526 may provide DXL client services or DXL broker 110. For example, DXL broker 110 may subscribe to certain types of messages in a client capacity.

[0101] FIGURE 6 is a block diagram of a CIM server 410 according to one or more examples of the present specification. In an example, CIM server 410 is controlled by a processor 610. Processor 610 may connect to other system elements via system bus 670. Those other elements may include, by way of non-limiting example, a memory 620, network interface 640, and storage 650. Reference is made to corresponding elements in FIGURE 5, which contains additional details and definitions.

[0102] Memory 620 may include CIM server software 622. CIM server software 622 may be configured to provide CIM services as described herein.

[0103] Storage 650 may include a local object database 652. Object database 652 may contain stored information about objects on the network, including, for example, reputations and metadata.

[0104] FIGURE 7 is a block diagram of a domain master 160 according to one or more examples of the present specification. In an example, domain master 160 is controlled by a processor 710. Processor 710 may connect to other system elements via system bus 770. Those other elements may include, by way of non-limiting example, a memory 720, network interface 740, and storage 750. Reference is made to corresponding elements in FIGURE 5, which contains additional details and definitions.

[0105] If in an example, memory 720 includes domain master software 722 and DXL extension 724. Domain master software 722 may be configured to provide domain master services as described herein. DXL extensions 724 may provide server extensions that allow domain master 162 operate on DXL ESB 130. In some embodiments, DXL extension 724 may include instructions that permit domain master 160 to act as a DXL client in some situations.

[0106] FIGURE 8 is a block diagram of a client 120 according to one or more examples of the present specification. Client 120 is controlled by a processor 810, which is communicatively coupled to a memory element 820. In an example, processor 810 is communicatively coupled to other system elements via bus 870. Those elements may include, by way of non-limiting example, a network interface 840, storage 850, which in some cases may be a species of memory element 820, and a user interface 860. It is expressly intended that any of the above elements can be realized in hardware, software, firmware, or any combination thereof.

[0107] In some embodiments, a user interface 860 may be provided to aid a user in interacting with client 120. A "user interface" includes any combination of hardware, software, and firmware configured to enable a user to interact with client 120, whether or not in real-time. In the example, user interface 360 may include, by way of non-limiting example, a keyboard (not shown), mouse (not shown), display monitor 842, speaker 846, microphone 844, touch-sensitive display, which may act as a combined input/output device, and which may be a species of display 842, and a camera 848. User interface 860 may include software services such as a graphical user interface, including real-time dialog boxes that solicit input or confirmation from a user.

[0108] In an example, memory 820 has stored therein executable instructions operable to provide, which may be contained in several distinct modules. A DXL client 826 may provide software services for interacting with DXL ESB 130, and may include, for example, certificates authenticating client 120 on DXL ESB 130, subroutines for publishing messages, and subroutines for parsing subscribed incoming messages. CIM client 822 may provide CIM services as described with more particularity in relation to FIGURE 4. CIM client 822 may also maintain a comprehensive catalog of objects stored objects 852, including for example a comprehensive catalog of installed applications. JTI client 824 may provide JTI client services, such as local reputation management and interactions with JTI server 150. Client 120 may also have a separate antimalware agent 828, which may provide antivirus and other antimalware services. In some cases, antimalware agent 828 is integrated with JTI client 824.

[0109] FIGURE 9 is a flow diagram illustrating evaluation of an object in a hierarchical manner according to one or more examples of the present specification.

[0110] In this example, policies on the left are designated as "black," and deterministically block an object, while policies on the right are designated as "white," and deterministically allow an object. By way of non-limiting example, the policies of FIGURE 9 are disclosed as a fixed hierarchy. However, this is not intended to be limiting. For example, in other embodiments, policies may be weighted, or serviced in a round-robin fashion. Furthermore, in embodiments where a hierarchy is used, this particular hierarchy is not required.

[0111] In block 910, an administrator, which may be either or both of a human operator or a device with administrative credentials, may assign a "black" override to the object, which may be identified by a hash. In block 912, the administrator may provide a "white" override to an object identified by a hash.

[0112] In block 920, an administrator may assign a black to override to an object identified by a certificate. In block 922, an administrator may assign a white overwrite to an object the identified by a certificate.

[0113] In block 930, threat intelligence service 180 may assign a black status based on a certificate. In block 932, threat intelligence service 180 may assign an object in white status based on a certificate.

[0114] In block 940, threat intelligence service 180 may assign an object black status base on a hash. In block 942, threat intelligence service 180 may assign an object white status based on a hash.

[0115] In block 950, malware definitions may identify an object as black, while in block 952, malware definitions may identify the object as white.

[0116] In block 960, JTI rules may assume that the object is dirty. In block 962, JTI rules may assume that the object is clean.

[0117] JTI client 824 may prompt an end user, via user interface 860, to confirm executing or opening an object. If the user declines, then in block 970 the object is blocked. If the user confirms, then in block 972 the object is allowed.

[0118] In block 980, if the object has passed all the preceding filters, a decision to either block or allow the object depends on JTI policies and algorithms. This may include requesting a reputation for the object.

[0119] FIGURE 10 is a flow diagram of a method of performed by an client 120 according to one or more examples of the present specification. In block 1010, client 120 opens a new object. This may be, for example, because a user interacts with the object, or because an automated process and racks of the object. In block 1020, client 120 checks to see if the object as a cache reputation and its local reputation database. If the object has an existing reputation, then in block 1030, client 120 may make a decision to either block or allow the object. In some examples, client 120 may publish the block or allowed decision over DXL ESB 130. This allows other DXL Network objects that have subscribed to such updates to

receive and integrate the decision. In block 1022, if there is no cash reputation, then client 120 may request a reputation from JTI server 150. In block 1024, client 120 may receive a reputation data from JTI server 150, whereupon control passes to block 130. As shown by the looping arrow back to 1010, this process may be repeated for each new object.

[0120] FIGURE 10A is a flow diagram of a method performed by JTI server 150 in connection with the method of FIGURE 10 according to one or more example embodiments of the present specification. In block 1040, JTI server 150 receives a reputation request from client 120. In block 1060, JTI server 150 may query its local database to find whether the object is already known. If the object is known, then in block 1070, JTI server 150 may update its local threat intelligence database. This may include, for example, noting the circumstances of the request from client 120. In block 1080, JTI server 150 may return the reputation data to client 120. In block 1090, the method is complete.

[0121] Returning to block 1060, if the object is not known, then in block 1050 JTI server 150 may request information from the threat intelligence service 180. In block 1052, J to a server 150 receives threat intelligence data from threat intelligence service 180, and in block 1070 updates its local JTI database with the received intelligence. Again in block 1080, the intelligence is returned to client 120, and in block 1090 the process is done.

[0122] Block 1034 represents an alternate entry point into the method. A lot 1034, a JTI server 150 receives a published block more about decision, for example a decision to published by client 120 according to a method of FIGURE 10. Depending on the network configuration, JTI server 150 may subscribe two published block/allow decisions, and in block 1070 may use a publish decision to update its local take a database. In this case, control passes directly from block 1080 to block 1090.

[0123] FIGURE 11 is a flow diagram of a method of a JTI server 150 servicing a reputation request according to one or more examples of the present specification. In some embodiments, the method of FIGURE 11 may be performed in connection with block 1080 of FIGURE 10A. In block 1110, an administrator may enter white or black overrides, for example as disclosed in connection with FIGURE 9. Block 1040 corresponds to block 1040 of FIGURE 10A, in which the JTI server 150 receives a reputation request from client 120. In block 1130, JTI server 150 checks to see whether the administrator has entered an override, such as an enterprise-wide override. In particular, an override may remove the decision of whether to

allow or block the object. Instead, the override may deterministically allow or block the object. Thus, in block 1140, JTI server 150 returns to client 120 the override. In block 1130, if there is no enterprise override, then in block 1150, JTI server 150 returns the reputation for client 120 to act on.

[0124] FIGURE 12 is a flow diagram of an enhancement to the method of FIGURE 10A according to one or more examples of the present specification. Blocks in FIGURE 12 with the numbering identical to the blocks in FIGURE 10A may be identical and functions of blocks disclosed in FIGURE 10A.

[0125] New block 1210 queries whether threat intelligence is available from threat intelligence service 180. If threat intelligence is available, then in block 1052, JTI server 150 receives the threat intelligence as with FIGURE 10A. In block 1220, if no threat intelligence is available, then JTI server 150 may send the object to ATD 370 for analysis. Rather than respond directly to the request, ATD 370 may publish reputation data for the object once it has completed its analysis. In block 1230, if JTI server 150 may subscribe two published a reputation data messages, and thus receives the DXL message published by ATD 370. The remainder of the method may be unchanged.

[0126] Certain prior art security architectures rely on a "pull" model of security updates, such as weekly downloads of updated virus definitions. In some cases, this may introduce unacceptable latency in providing security updates to clients 120 and other DXL endpoints. Advantageously, the context-aware network 100 of the present specification may provide machine-real-time security updates when new threats are identified.

[0127] FIGURE 13 is a flow diagram of a method performed by a DXL master 160 according to one or more examples of the present specification. Blocks 1310, 1320, and 1330 represent three example “policy update driver” scenarios in which a security policy update is desirable. In block 1310, domain master 160 detects a new threat. In an example, domain master 160 detects the threat by receiving updated reputation information, for example from threat intelligence service 180. In another example, domain master 160 detects the threat by receiving a published notification on DXL ESB 130 identifying an object as malware. In yet another example, domain master 160 identifies the object as malware by analyzing the object and determining that it exhibits malware behavior.

[0128] In block 1320, domain master 160 proactively identifies a policy change requirement, such as an update to an existing security policy, or the need for a new policy. This may be the result, for example, of network heuristics, or of proactive analysis of network traffic.

[0129] In block 1330, an administrator may provide a change in policy, or a new policy.

[0130] In block 1340, domain master 160 configures a policy update responsive to inputs. Configuring the policy update may include designating an action to take in response to the object. The action may include, by way of nonlimiting example, quarantine, block, delete, sandbox, deny permissions (e.g., if the object is identified as grayware), remedy, prompt, or custom actions.

[0131] In block 1350, domain master 160 may provide the updated policy to DXL brokers 110. Providing the updated policy may include publishing a command-level message on DXL ESB 130. In some examples, in block 1360, domain master 160 may also publish a message suitable for clients 120 to receive the updated security policy. In some cases, clients 120 may be managed devices, and the published message may be a command-level message. Once domain master 160 has published the message to appropriate endpoints, in block 1390 the process may be complete.

[0132] FIGURE 14 is a flow diagram of a method performed by DXL broker 110, in response to a command-level message containing a security policy configuration update, according to one or more examples of the present specification. In block 1410, DXL broker 110 receives the policy update from domain master 160. In block 1420, DXL broker 110 verifies credentials received from domain master 160. This may include, for example, verifying a security certificate provided with the command-level message. If the certificate is not verified, then no action is taken, and in block 1490, the process is done. If the certificate is verified, then in block 1440, DXL broker 110 may update its local security database. In block 1450, DXL broker 110 may also publish a message on DXL ESB 130 notifying endpoints of the change. Note that in some embodiments, this publication may be unnecessary, as domain master 160 may take responsibility for publishing the security update. Thus, in some embodiments, the security policy update itself may include instructions for DXL broker 110,

including whether or not to additionally publish the updated security policy. In block 1490, and the process is done.

[0133] FIGURE 15 is a flow diagram of a method performed by a client 120 responsive to receiving a security policy update, either from domain master 160, or DXL broker 110 according to one or more examples embodiments of the present specification. In block 1510, client 120 receives the updated security policy. In block 1520, client 120 verifies the credentials of the device that sent the updated security policy. In block 1530, if the credentials are not valid, then no action is taken, and in block 1590 the method is done.

[0134] Advantageously, according to the methods disclosed herein, DXL endpoints may maintain a state of consistent readiness in dealing with threats. This may be particularly useful in dealing with so-called "zero day" exploits, where rapid response is critical.

[0135] The foregoing outlines features of several embodiments so that those skilled in the art may better understand the aspects of the present disclosure. Those skilled in the art should appreciate that they may readily use the present disclosure as a basis for designing or modifying other processes and structures for carrying out the same purposes and/or achieving the same advantages of the embodiments introduced herein. Those skilled in the art should also realize that such equivalent constructions do not depart from the spirit and scope of the present disclosure, and that they may make various changes, substitutions, and alterations herein without departing from the spirit and scope of the present disclosure.

[0136] The particular embodiments of the present disclosure may readily include a system on chip (SOC) central processing unit (CPU) package. An SOC represents an integrated circuit (IC) that integrates components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed-signal, and radio frequency functions: all of which may be provided on a single chip substrate. Other embodiments may include a multi-chip-module (MCM), with a plurality of chips located within a single electronic package and configured to interact closely with each other through the electronic package. In various other embodiments, the digital signal processing functionalities may be implemented in one or more silicon cores in Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs), and other semiconductor chips.

[0137] In example implementations, at least some portions of the processing activities outlined herein may also be implemented in software. In some embodiments, one

or more of these features may be implemented in hardware provided external to the elements of the disclosed FIGUREs, or consolidated in any appropriate manner to achieve the intended functionality. The various components may include software (or reciprocating software) that can coordinate in order to achieve the operations as outlined herein. In still other embodiments, these elements may include any suitable algorithms, hardware, software, components, modules, interfaces, or objects that facilitate the operations thereof.

[0138] Additionally, some of the components associated with described microprocessors may be removed, or otherwise consolidated. In a general sense, the arrangements depicted in the FIGURES may be more logical in their representations, whereas a physical architecture may include various permutations, combinations, and/or hybrids of these elements. It is imperative to note that countless possible design configurations can be used to achieve the operational objectives outlined herein. Accordingly, the associated infrastructure has a myriad of substitute arrangements, design choices, device possibilities, hardware configurations, software implementations, equipment options, etc.

[0139] Any suitably-configured processor component can execute any type of instructions associated with the data to achieve the operations detailed herein. Any processor disclosed herein could transform an element or an article (for example, data) from one state or thing to another state or thing. In another example, some activities outlined herein may be implemented with fixed logic or programmable logic (for example, software and/or computer instructions executed by a processor) and the elements identified herein could be some type of a programmable processor, programmable digital logic (for example, a field programmable gate array (FPGA), an erasable programmable read only memory (EPROM), an electrically erasable programmable read only memory (EEPROM)), an ASIC that includes digital logic, software, code, electronic instructions, flash memory, optical disks, CD-ROMs, DVD ROMs, magnetic or optical cards, other types of machine-readable mediums suitable for storing electronic instructions, or any suitable combination thereof. In operation, processors may store information in any suitable type of non-transitory storage medium (for example, random access memory (RAM), read only memory (ROM), field programmable gate array (FPGA), erasable programmable read only memory (EPROM), electrically erasable programmable ROM (EEPROM), etc.), software, hardware, or in any other suitable component, device, element, or object where appropriate and based on particular needs.

Further, the information being tracked, sent, received, or stored in a processor could be provided in any database, register, table, cache, queue, control list, or storage structure, based on particular needs and implementations, all of which could be referenced in any suitable timeframe. Any of the memory items discussed herein should be construed as being encompassed within the broad term 'memory.' Similarly, any of the potential processing elements, modules, and machines described herein should be construed as being encompassed within the broad term 'microprocessor' or 'processor.'

[0140] Computer program logic implementing all or part of the functionality described herein is embodied in various forms, including, but in no way limited to, a source code form, a computer executable form, and various intermediate forms (for example, forms generated by an assembler, compiler, linker, or locator). In an example, source code includes a series of computer program instructions implemented in various programming languages, such as an object code, an assembly language, or a high-level language such as OpenCL, Fortran, C, C++, JAVA, or HTML for use with various operating systems or operating environments. The source code may define and use various data structures and communication messages. The source code may be in a computer executable form (e.g., via an interpreter), or the source code may be converted (e.g., via a translator, assembler, or compiler) into a computer executable form.

[0141] In the discussions of the embodiments above, the capacitors, buffers, graphics elements, interconnect boards, clocks, DDRs, camera sensors, dividers, inductors, resistors, amplifiers, switches, digital core, transistors, and/or other components can readily be replaced, substituted, or otherwise modified in order to accommodate particular circuitry needs. Moreover, it should be noted that the use of complementary electronic devices, hardware, non-transitory software, etc. offer an equally viable option for implementing the teachings of the present disclosure.

[0142] In one example, any number of electrical circuits of the FIGURES may be implemented on a board of an associated electronic device. The board can be a general circuit board that can hold various components of the internal electronic system of the electronic device and, further, provide connectors for other peripherals. More specifically, the board can provide the electrical connections by which the other components of the system can communicate electrically. Any suitable processors (inclusive of digital signal processors,

microprocessors, supporting chipsets, etc.), memory elements, etc. can be suitably coupled to the board based on particular configuration needs, processing demands, computer designs, etc. Other components such as external storage, additional sensors, controllers for audio/video display, and peripheral devices may be attached to the board as plug-in cards, via cables, or integrated into the board itself. In another example, the electrical circuits of the FIGURES may be implemented as stand-alone modules (e.g., a device with associated components and circuitry configured to perform a specific application or function) or implemented as plug-in modules into application specific hardware of electronic devices.

[0143] Note that with the numerous examples provided herein, interaction may be described in terms of two, three, four, or more electrical components. However, this has been done for purposes of clarity and example only. It should be appreciated that the system can be consolidated in any suitable manner. Along similar design alternatives, any of the illustrated components, modules, and elements of the FIGURES may be combined in various possible configurations, all of which are clearly within the broad scope of this Specification. In certain cases, it may be easier to describe one or more of the functionalities of a given set of flows by only referencing a limited number of electrical elements. It should be appreciated that the electrical circuits of the FIGURES and its teachings are readily scalable and can accommodate a large number of components, as well as more complicated/sophisticated arrangements and configurations. Accordingly, the examples provided should not limit the scope or inhibit the broad teachings of the electrical circuits as potentially applied to a myriad of other architectures.

[0144] Numerous other changes, substitutions, variations, alterations, and modifications may be ascertained to one skilled in the art and it is intended that the present disclosure encompass all such changes, substitutions, variations, alterations, and modifications as falling within the scope of the appended claims. In order to assist the United States Patent and Trademark Office (USPTO) and, additionally, any readers of any patent issued on this application in interpreting the claims appended hereto, Applicant wishes to note that the Applicant: (a) does not intend any of the appended claims to invoke paragraph six (6) of 35 U.S.C. section 112 as it exists on the date of the filing hereof unless the words "means for" or "steps for" are specifically used in the particular claims; and (b) does not

intend, by any statement in the specification, to limit this disclosure in any way that is not otherwise reflected in the appended claims.

EXAMPLE EMBODIMENT IMPLEMENTATIONS

[0145] There is disclosed in an example 1, one or more non-transitory computer-readable mediums having stored thereon executable instructions for providing a domain master, the executable instructions operable to instruct a processor to:

identify a policy update driver;

configure a policy update; and

publish the policy update as a data exchange layer (DXL) message.

[0146] There is disclosed in an example 2, the computer-readable medium of example 1, wherein the DXL message is a command-level message.

[0147] There is disclosed in an example 3, the computer-readable medium of example 1 or 2, wherein the DXL message is targeted to DXL endpoints.

[0148] There is disclosed in an example 4, the computer-readable medium of any of examples 1 – 3, wherein the instructions operable to identify a policy update driver are operable to receive a malware threat message from a threat intelligence service.

[0149] There is disclosed in an example 5, the computer-readable medium of any of examples 1 – 4, wherein the instructions operable to identify a policy update driver are operable to proactively identify a policy update need.

[0150] There is disclosed in an example 6, the computer-readable medium of any of examples 1 – 5, wherein the instructions operable to identify a policy update driver are operable to receive a policy update from an administrator.

[0151] There is disclosed in an example 7, the computer-readable medium of any of examples 1 – 6, wherein the instructions operable to configure a policy update are operable to direct DXL endpoints to take an action selected from the group consisting of quarantine, block, delete, sandbox, deny permissions, remedy, prompt, or custom action.

[0152] There is disclosed in an example 8, a data exchange layer (DXL) domain master comprising:

a processor;

a network interface configured to communicatively coupled the processor to a DXL enterprise service bus; and

a memory having stored therein executable instructions operable to instruct the processor to:

identify a policy update driver;

configure a policy update; and

publish the policy update as a data exchange layer (DXL) message.

[0153] There is disclosed in an example 9, the DXL domain master of example 8, wherein the DXL message is a command-level message.

[0154] There is disclosed in an example 10, the DXL domain master of example 8 or 9, wherein the DXL message is targeted to DXL endpoints.

[0155] There is disclosed in an example 11, the DXL domain master of any of examples 8 – 10, wherein the instructions operable to identify a policy update driver are operable to receive a malware threat message from a threat intelligence service.

[0156] There is disclosed in an example 12, the DXL domain master of any of examples 8 – 11, wherein the instructions operable to identify a policy update driver are operable to proactively identify a policy update need.

[0157] There is disclosed in an example 13, the DXL domain master of any of examples 8 – 12, wherein the instructions operable to identify a policy update driver are operable to receive a policy update from an administrator.

[0158] There is disclosed in an example 14, the DXL domain master of any of examples 8 – 13, wherein the instructions operable to configure a policy update are operable to direct DXL endpoints to take an action selected from the group consisting of quarantine, block, delete, sandbox, deny permissions, remedy, prompt, or custom action.

[0159] There is disclosed in an example 15, a method of providing DXL domain master services, comprising:

identifying a policy update driver;

configuring a policy update; and

publishing the policy update as a data exchange layer (DXL) message.

[0160] There is disclosed in an example 16, the method of example 15, wherein the DXL message is a command-level message.

[0161] There is disclosed in an example 17, the method of example 15 or 16, wherein the DXL message is targeted to DXL endpoints.

[0162] There is disclosed in an example 18, the method of any of examples 15 – 17, wherein identifying a policy update driver comprises receiving a malware threat message from a threat intelligence service.

[0163] There is disclosed in an example 19, the method of any of examples 15 – 18, wherein identifying a policy update driver comprises proactively identifying a policy update need.

[0164] There is disclosed in an example 20, the method of any of examples 15 – 19, wherein identifying a policy update driver comprises receiving a policy update from an administrator.

[0165] There is disclosed in an example 21, the method of any of examples 15 – 20, wherein configuring a policy update comprises directing DXL endpoints to take an action selected from the group consisting of quarantine, block, delete, sandbox, deny permissions, remedy, prompt, or custom action.

CLAIMS

WHAT IS CLAIMED IS:

1. One or more non-transitory computer-readable mediums having stored thereon executable instructions for providing a domain master, the executable instructions operable to instruct a processor to:

identify a policy update driver;

configure a policy update; and

publish the policy update as a data exchange layer (DXL) message.

2. The computer-readable medium of claim 1, wherein the DXL message is a command-level message.

3. The computer-readable medium of claim 1 or 2, wherein the DXL message is targeted to DXL endpoints.

4. The computer-readable medium of any of claims 1 – 3, wherein the instructions operable to identify a policy update driver are operable to receive a malware threat message from a threat intelligence service.

5. The computer-readable medium of any of claims 1 – 4, wherein the instructions operable to identify a policy update driver are operable to proactively identify a policy update need.

6. The computer-readable medium of any of claims 1 – 5, wherein the instructions operable to identify a policy update driver are operable to receive a policy update from an administrator.

7. The computer-readable medium of any of claims 1 – 6, wherein the instructions operable to configure a policy update are operable to direct DXL endpoints to take an action selected from the group consisting of quarantine, block, delete, sandbox, deny permissions, remedy, prompt, or custom action.

8. A data exchange layer (DXL) domain master comprising:

a processor;

a network interface configured to communicatively coupled the processor to a DXL enterprise service bus; and

a memory having stored therein executable instructions operable to instruct the processor to:

identify a policy update driver;

configure a policy update; and

publish the policy update as a data exchange layer (DXL) message.

9. The DXL domain master of claim 8, wherein the DXL message is a command-level message.

10. The DXL domain master of claim 8 or 9, wherein the DXL message is targeted to DXL endpoints.

11. The DXL domain master of any of claims 8 – 10, wherein the instructions operable to identify a policy update driver are operable to receive a malware threat message from a threat intelligence service.

12. The DXL domain master of any of claims 8 – 11, wherein the instructions operable to identify a policy update driver are operable to proactively identify a policy update need.

13. The DXL domain master of any of claims 8 – 12, wherein the instructions operable to identify a policy update driver are operable to receive a policy update from an administrator.

14. The DXL domain master of any of claims 8 – 13, wherein the instructions operable to configure a policy update are operable to direct DXL endpoints to take an action selected from the group consisting of quarantine, block, delete, sandbox, deny permissions, remedy, prompt, or custom action.

15. A method of providing DXL domain master services, comprising:

identifying a policy update driver;

configuring a policy update; and

publishing the policy update as a data exchange layer (DXL) message.

16. The method of claim 15, wherein the DXL message is a command-level message.

17. The method of claim 15 or 16, wherein the DXL message is targeted to DXL endpoints.

18. The method of any of claims 15 – 17, wherein identifying a policy update driver comprises receiving a malware threat message from a threat intelligence service.

19. The method of any of claims 15 – 18, wherein identifying a policy update driver comprises proactively identifying a policy update need.

20. The method of any of claims 15 – 19, wherein identifying a policy update driver comprises receiving a policy update from an administrator.

21. The method of any of claims 15 – 20, wherein configuring a policy update comprises directing DXL endpoints to take an action selected from the group consisting of quarantine, block, delete, sandbox, deny permissions, remedy, prompt, or custom action.

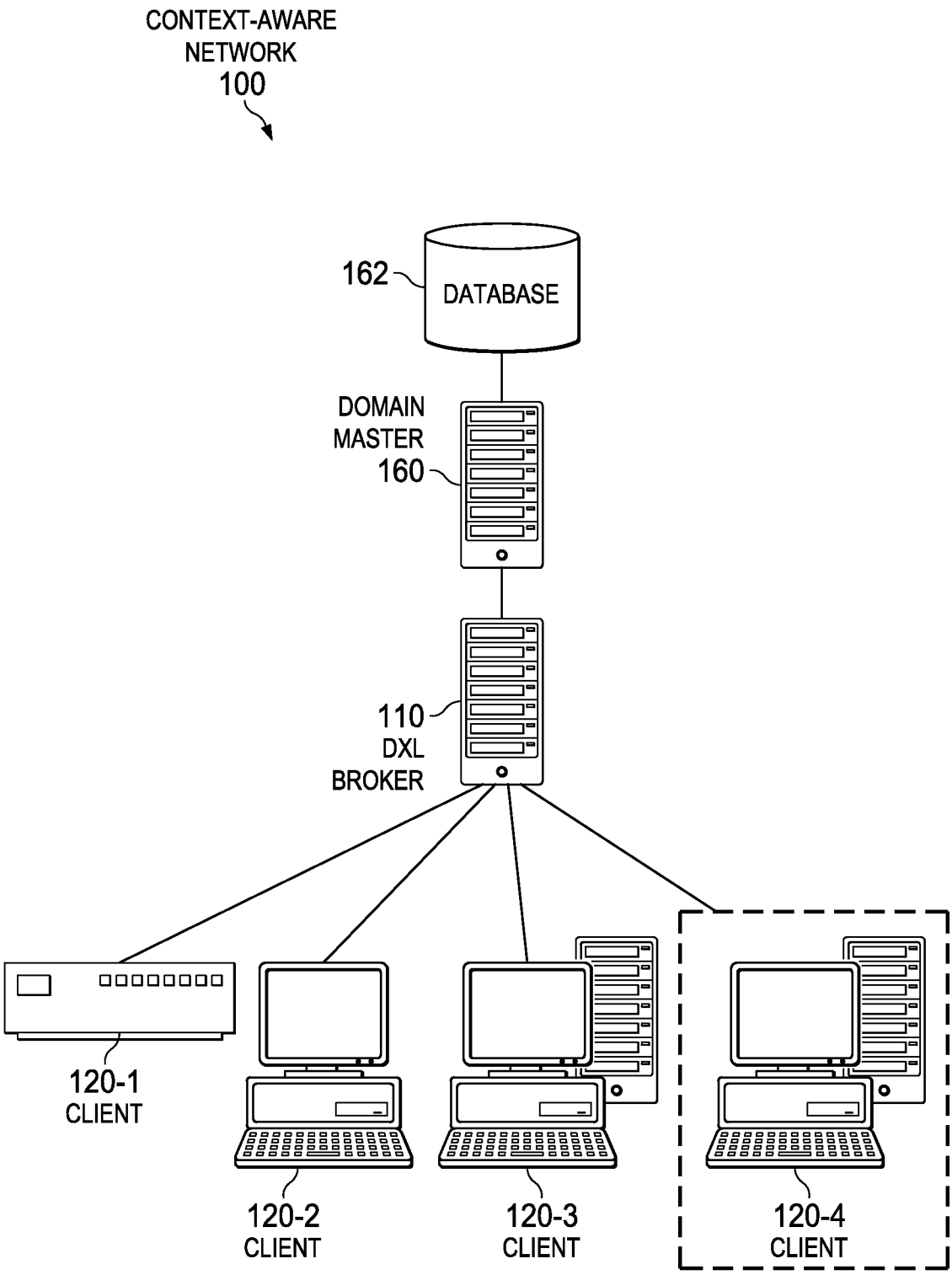

CONTEXT-AWARE
NETWORK
100
162
DATABASE
DOMAIN
MASTER
160
110
DXL
BROKER
120-1
CLIENT
120-2
CLIENT
120-3
CLIENT
120-4
CLIENT

FIG. 1

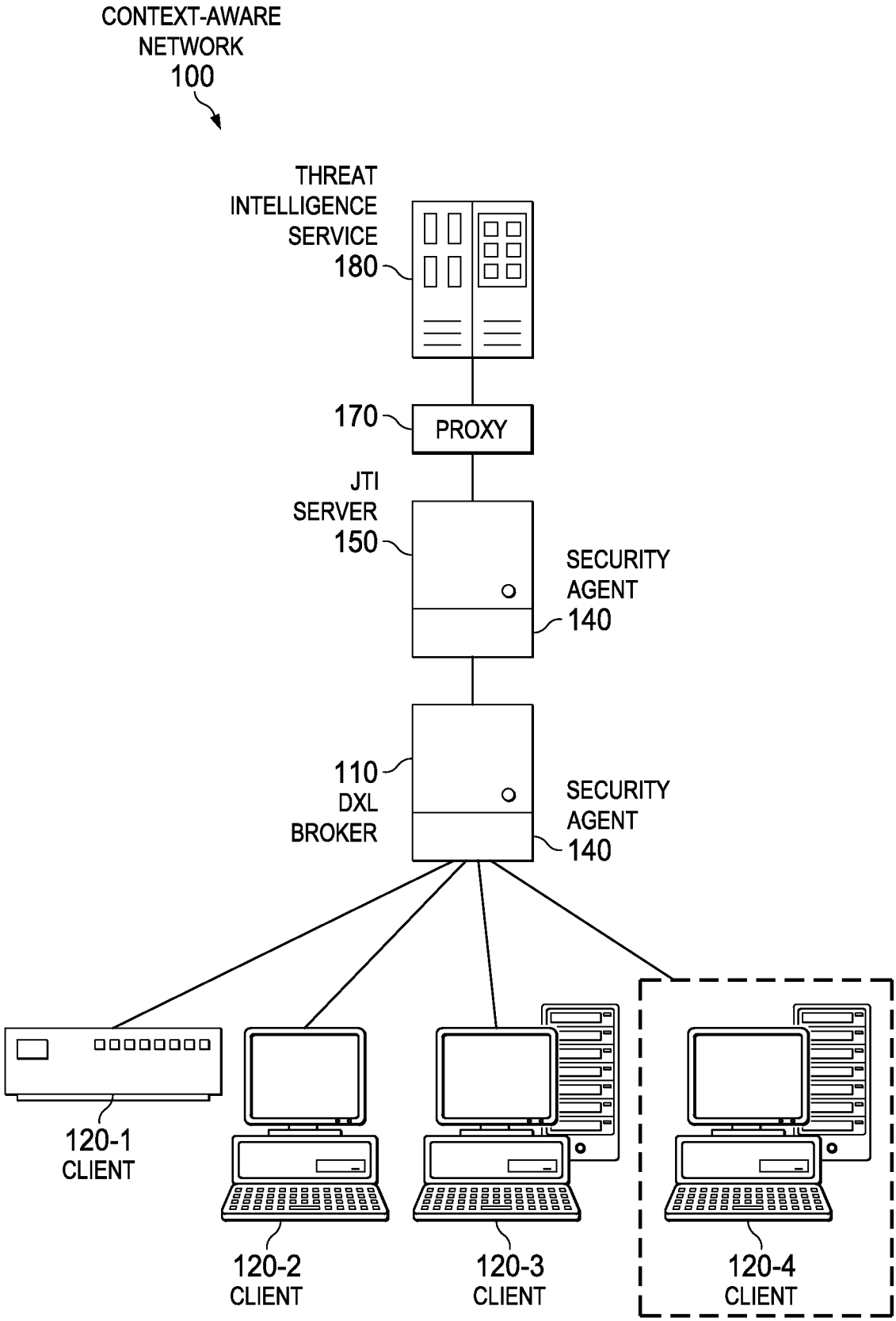

CONTEXT-AWARE
NETWORK
100
THREAT
INTELLIGENCE
SERVICE
180
170
PROXY
JTI
SERVER
150
SECURITY
AGENT
140
110
DXL
BROKER
SECURITY
AGENT
140
120-1
CLIENT
120-2
CLIENT
120-3
CLIENT
120-4
CLIENT

FIG. 1A

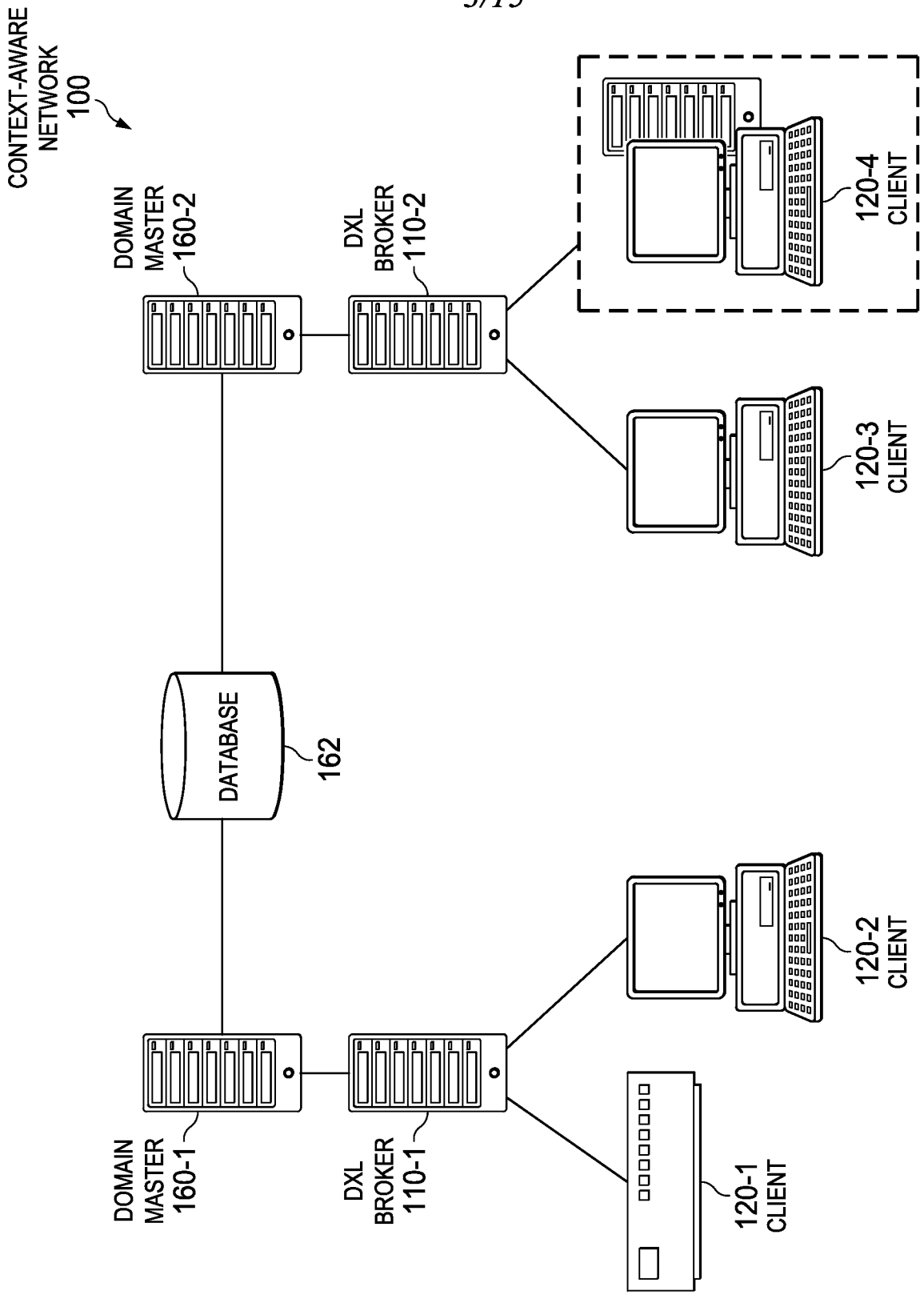

CONTEXT-AWARE NETWORK 100
DOMAIN MASTER 160-2
DXL BROKER 110-2
120-4 CLIENT
120-3 CLIENT
DATABASE
162
120-2 CLIENT
DOMAIN MASTER 160-1
DXL BROKER 110-1
120-1 CLIENT

FIG. 1B

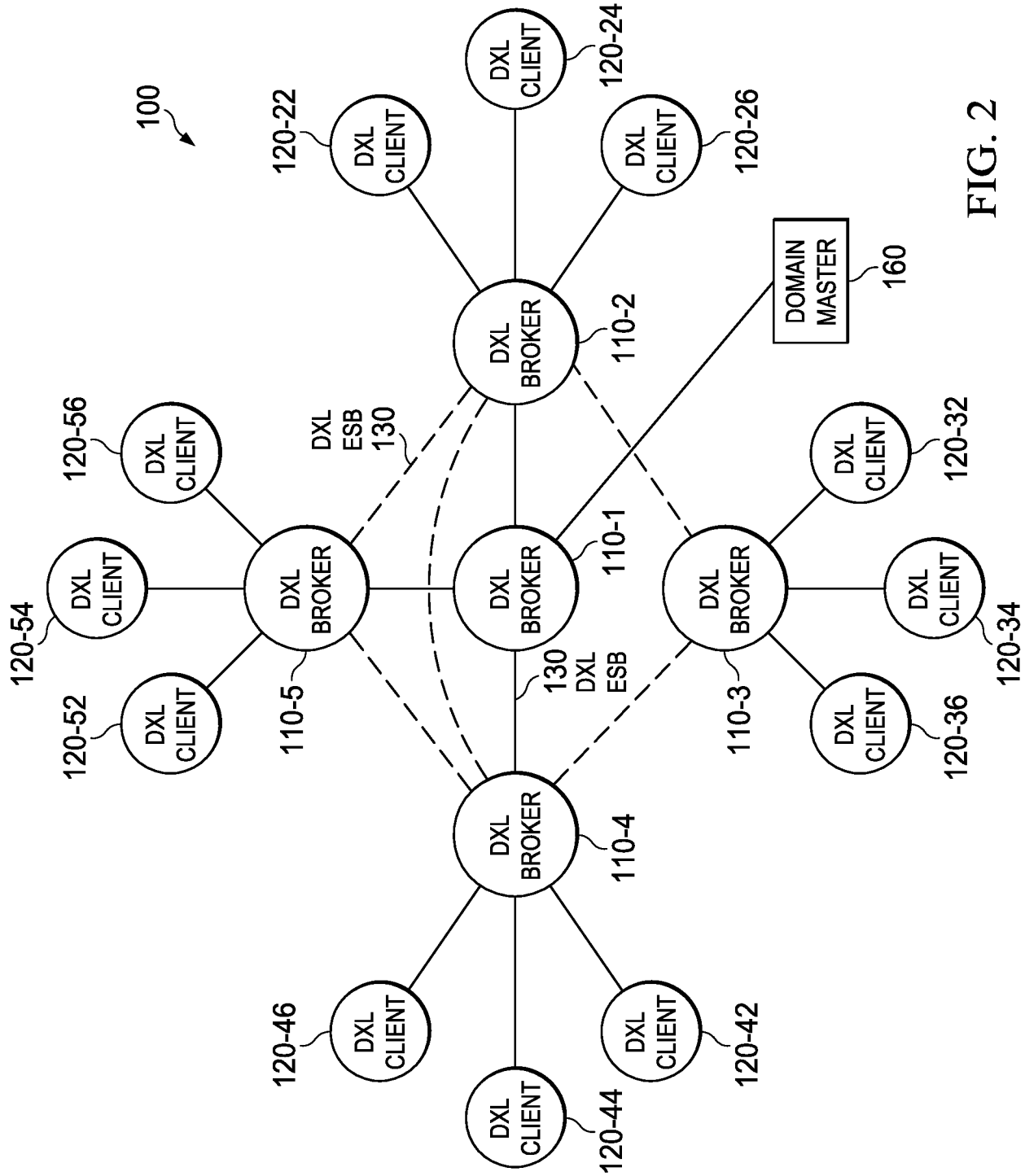

100
DXL BROKER
110-1
DXL BROKER
110-2
DXL BROKER
110-3
DXL BROKER
110-4
DXL BROKER
110-5
DOMAIN MASTER
160
DXL ESB
130
DXL ESB
130
DXL CLIENT
120-22
DXL CLIENT
120-24
DXL CLIENT
120-26
DXL CLIENT
120-32
DXL CLIENT
120-34
DXL CLIENT
120-36
DXL CLIENT
120-42
DXL CLIENT
120-44
DXL CLIENT
120-46
DXL CLIENT
120-52
DXL CLIENT
120-54
DXL CLIENT
120-56

FIG. 2

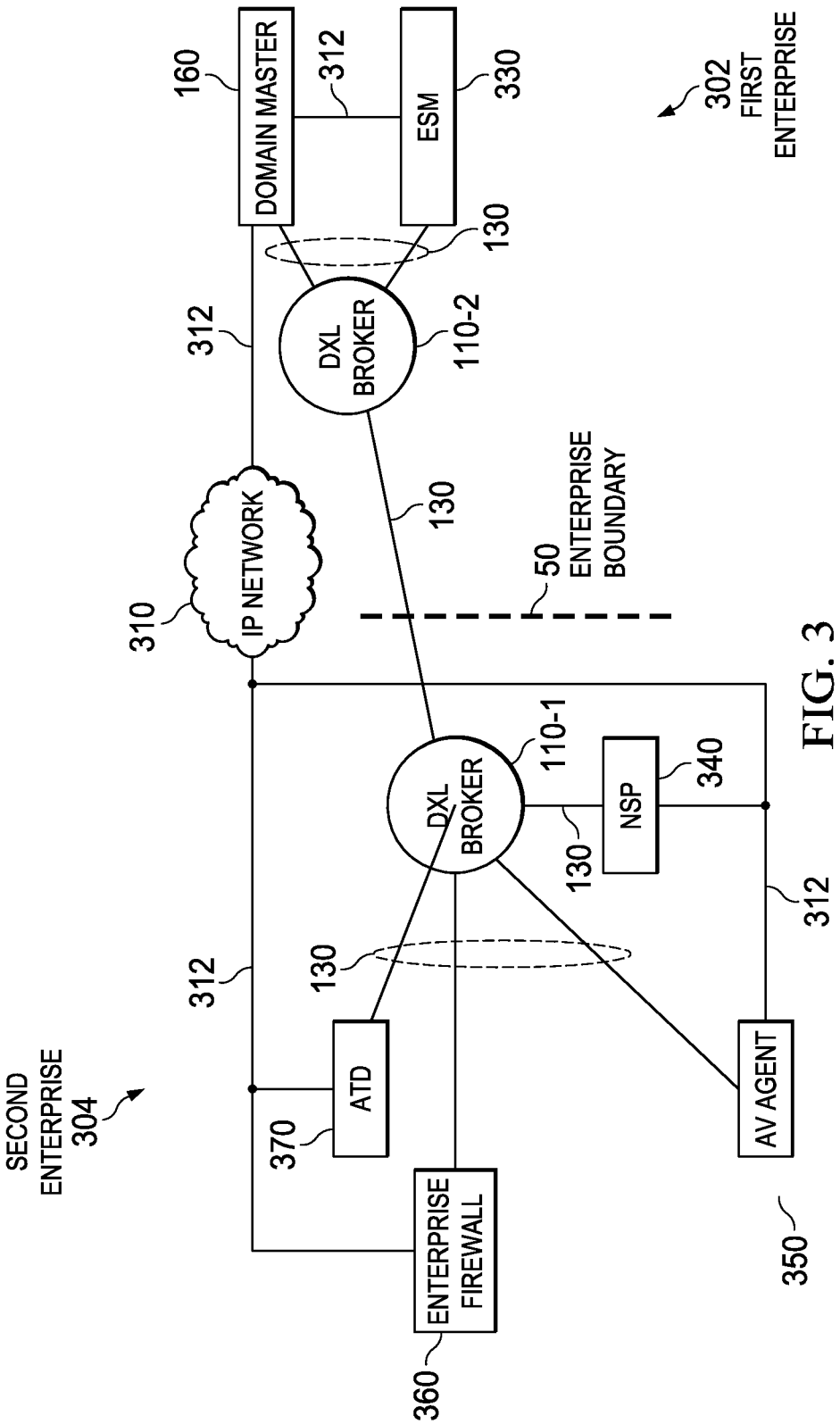

160
DOMAIN MASTER
312
ESM
330
302
FIRST
ENTERPRISE
130
312
DXL
BROKER
110-2
130
310
IP NETWORK
50
ENTERPRISE
BOUNDARY
DXL
BROKER
110-1
NSP
340
130
312
312
130
ATD
370
AV AGENT
SECOND
ENTERPRISE
304
ENTERPRISE
FIREWALL
360
350

FIG. 3

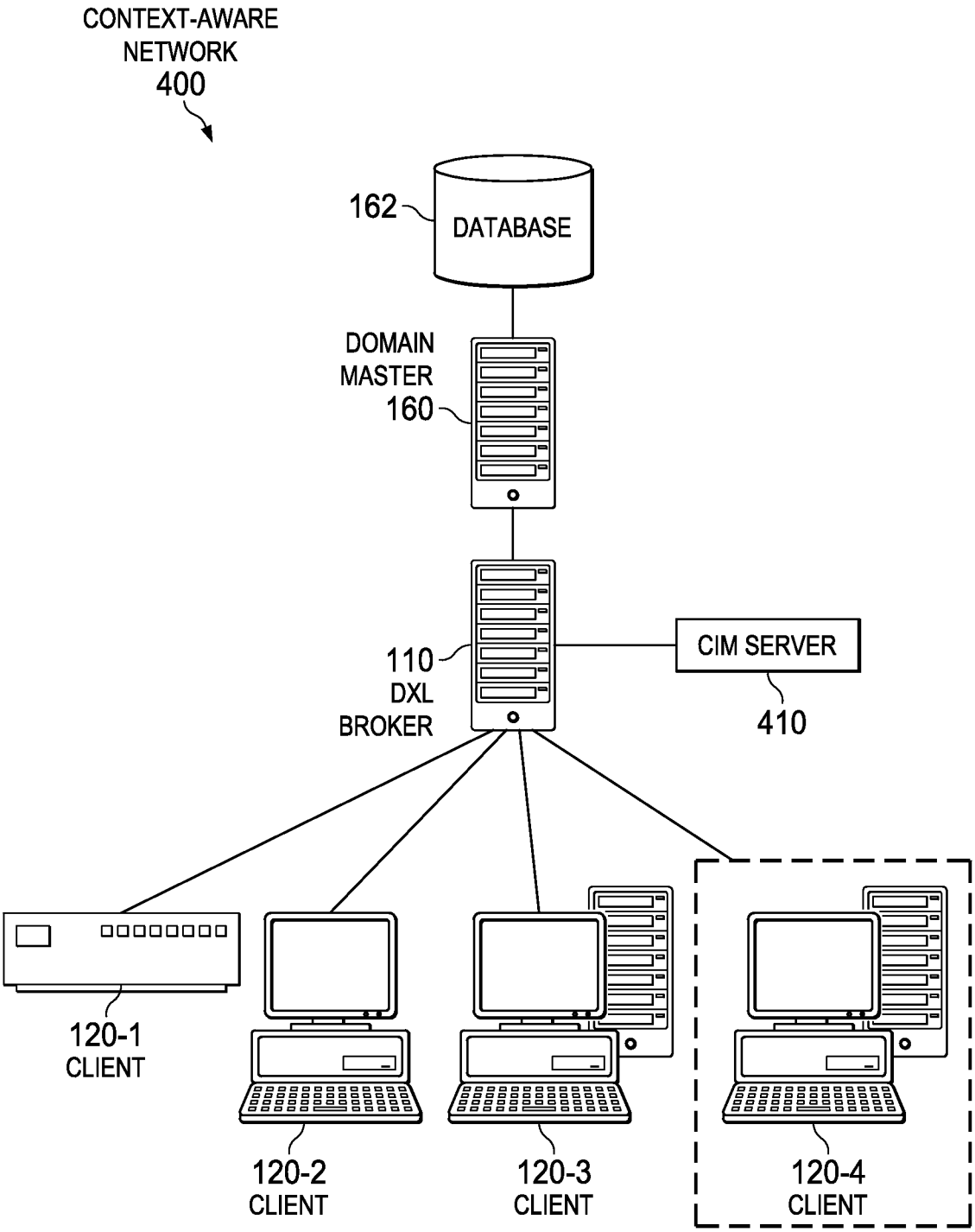

CONTEXT-AWARE
NETWORK
400
162
DATABASE
DOMAIN
MASTER
160
110
DXL
BROKER
CIM SERVER
410
120-1
CLIENT
120-2
CLIENT
120-3
CLIENT
120-4
CLIENT

FIG. 4

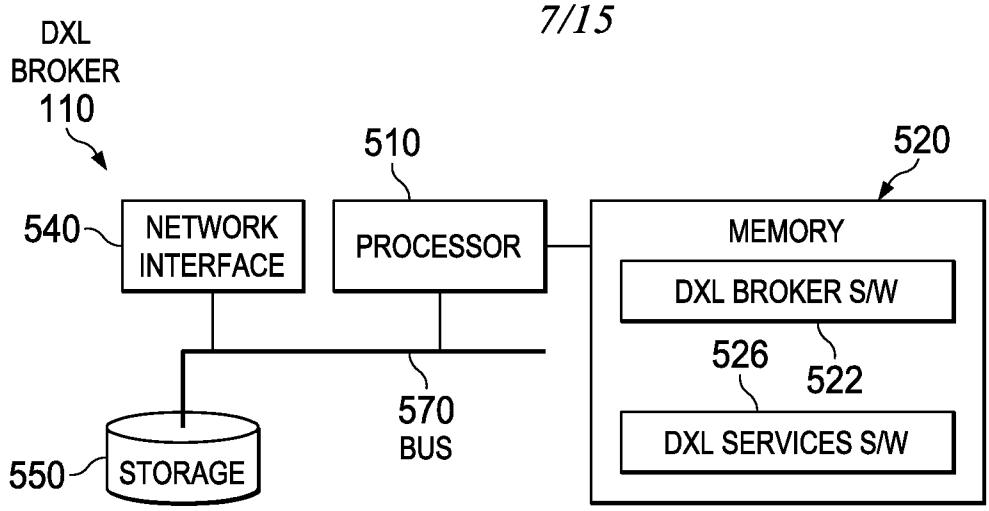


FIG. 5

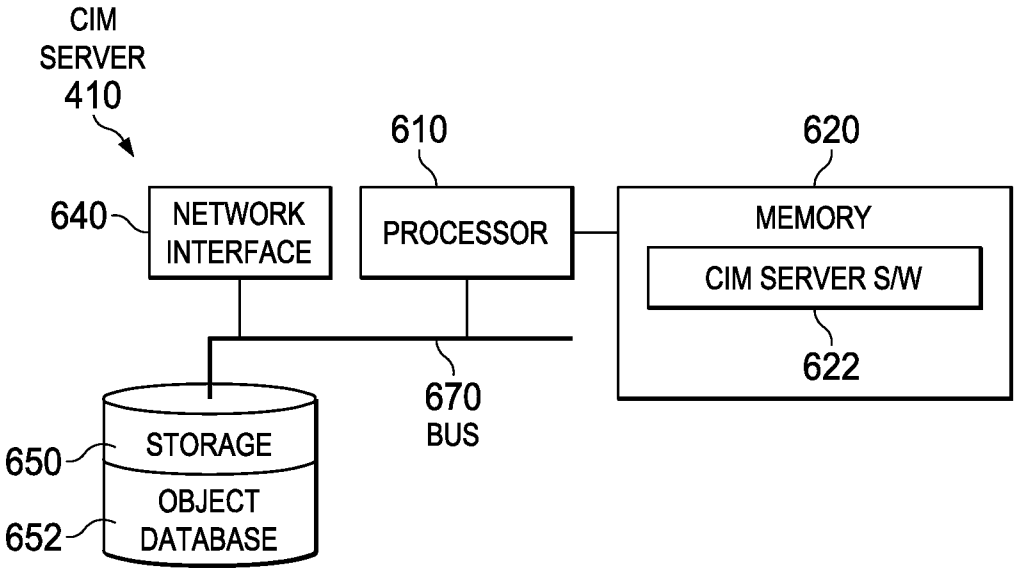


FIG. 6

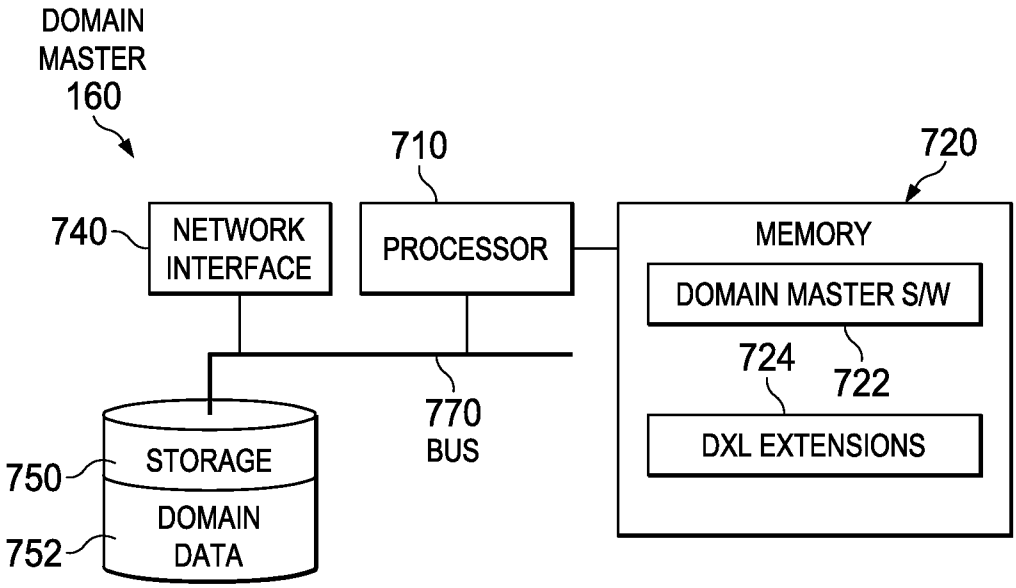


FIG. 7

FIG. 8

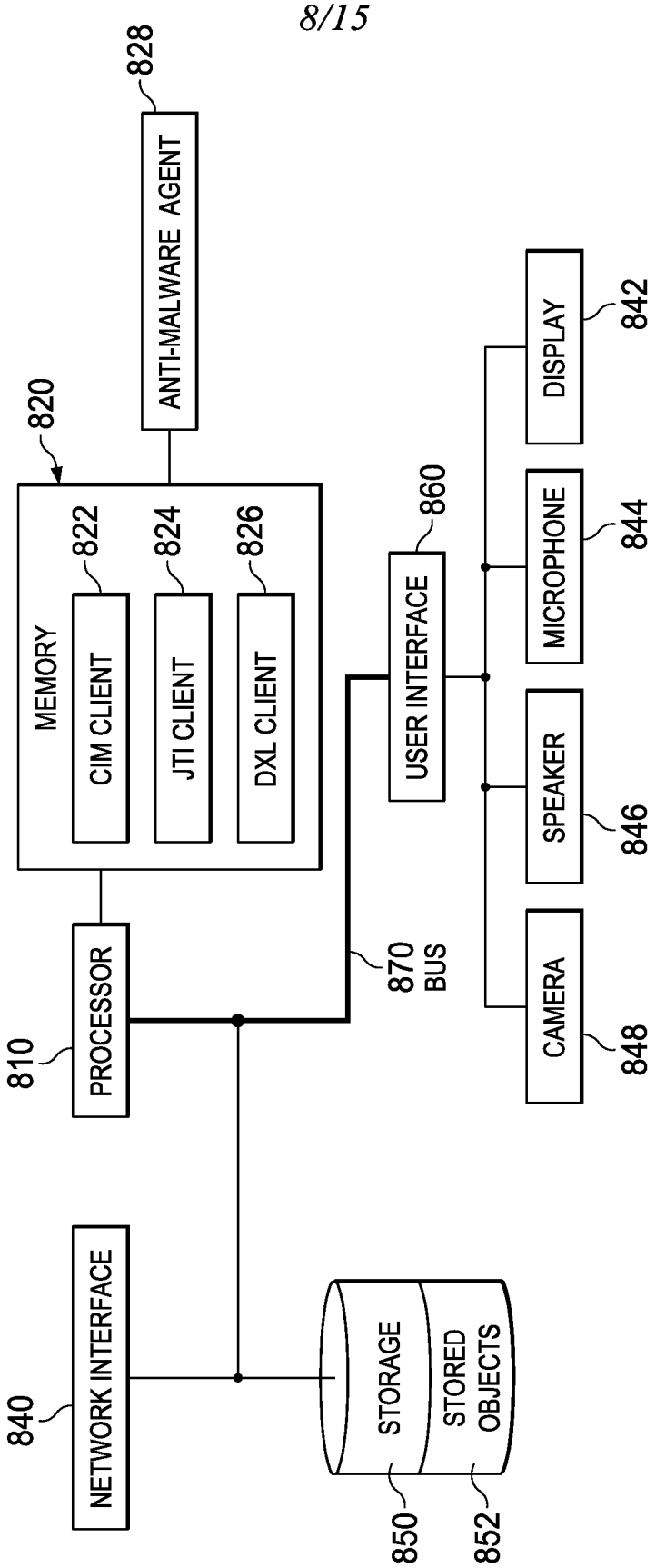
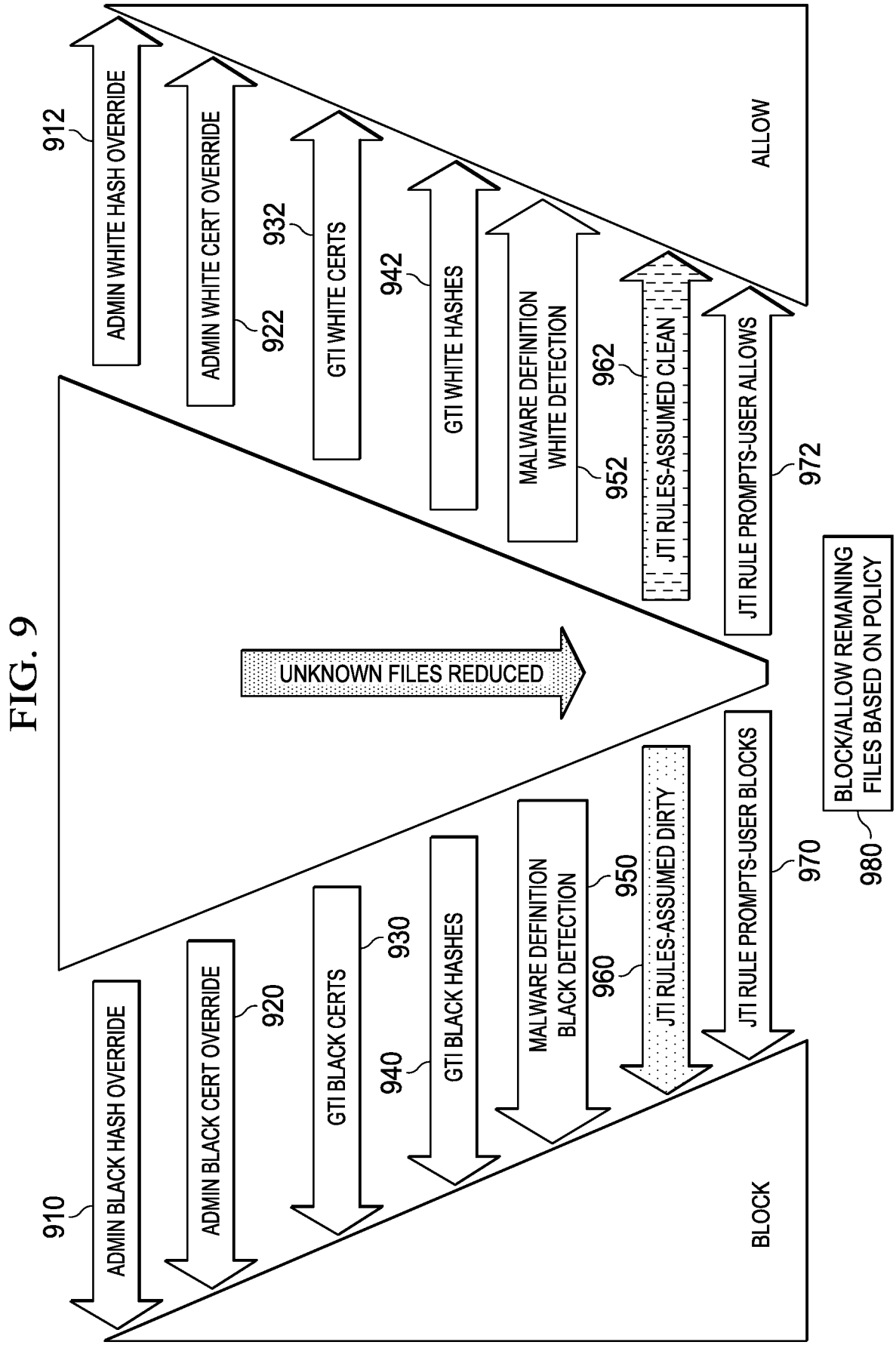

828
ANTI-MALWARE AGENT
820
MEMORY
822
CIM CLIENT
824
JTI CLIENT
826
DXL CLIENT
860
USER INTERFACE
842
DISPLAY
844
MICROPHONE
846
SPEAKER
848
CAMERA
870
BUS
810
PROCESSOR
840
NETWORK INTERFACE
STORAGE
STORED OBJECTS
850
852

FIG. 9


ALLOW
912
ADMIN WHITE HASH OVERRIDE
922
ADMIN WHITE CERT OVERRIDE
932
GTI WHITE CERTS
942
GTI WHITE HASHES
952
MALWARE DEFINITION WHITE DETECTION
962
JTI RULES-ASSUMED CLEAN
972
JTI RULE PROMPTS-USER ALLOWS
UNKNOWN FILES REDUCED
980
BLOCK/ALLOW REMAINING FILES BASED ON POLICY
970
JTI RULE PROMPTS-USER BLOCKS
960
JTI RULES-ASSUMED DIRTY
950
MALWARE DEFINITION BLACK DETECTION
940
GTI BLACK HASHES
930
GTI BLACK CERTS
920
ADMIN BLACK CERT OVERRIDE
910
ADMIN BLACK HASH OVERRIDE
BLOCK

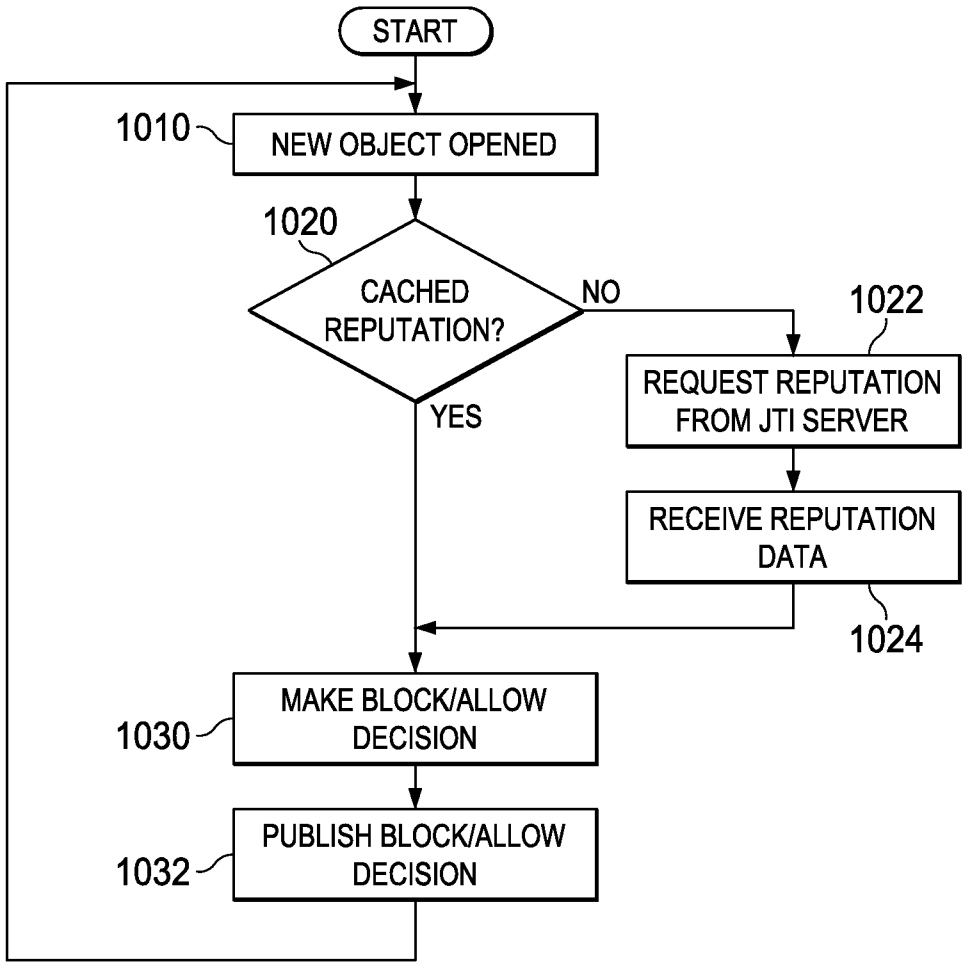

START
1010
NEW OBJECT OPENED
1020
CACHED REPUTATION?
NO
1022
REQUEST REPUTATION FROM JTI SERVER
RECEIVE REPUTATION DATA
1024
YES
1030
MAKE BLOCK/ALLOW DECISION
1032
PUBLISH BLOCK/ALLOW DECISION

FIG. 10

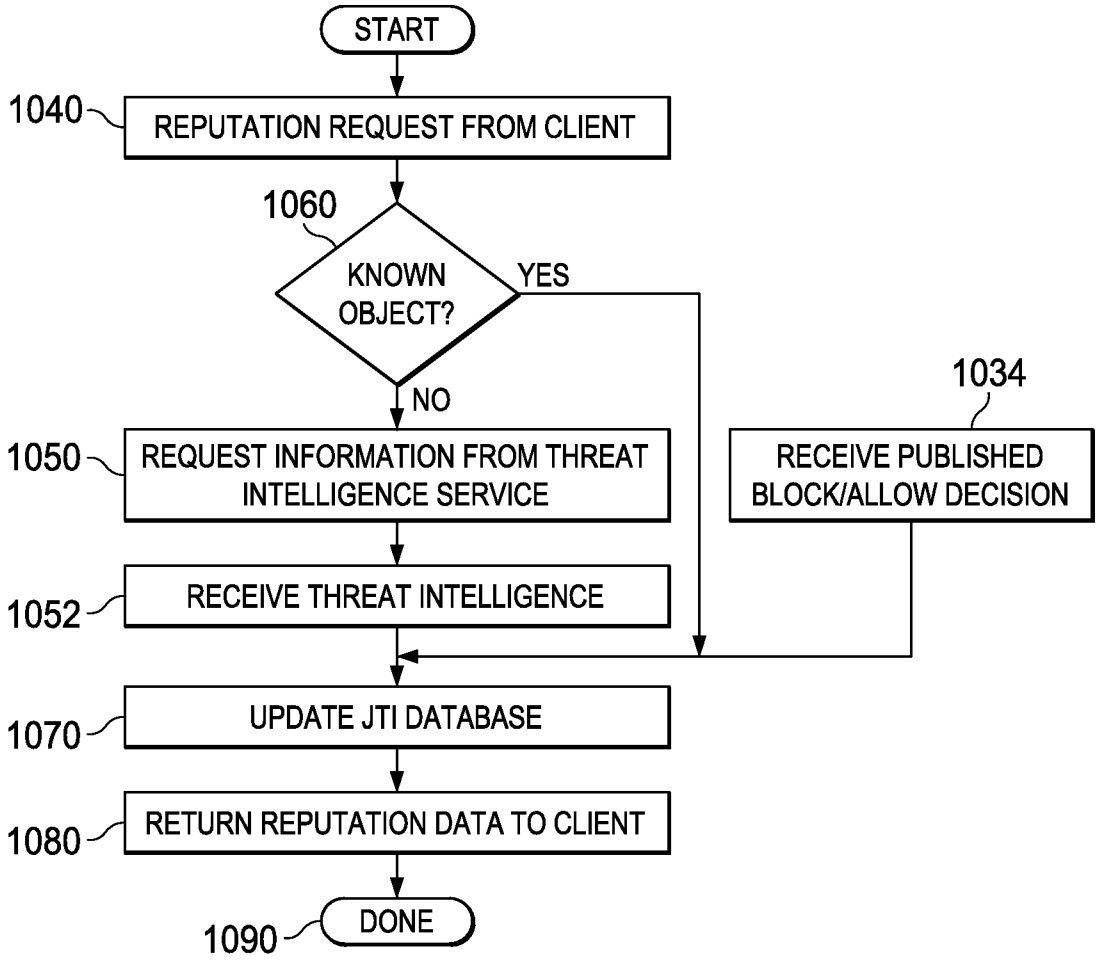

START
1040
REPUTATION REQUEST FROM CLIENT
1060
KNOWN OBJECT?
YES
NO
1034
RECEIVE PUBLISHED BLOCK/ALLOW DECISION
1050
REQUEST INFORMATION FROM THREAT INTELLIGENCE SERVICE
1052
RECEIVE THREAT INTELLIGENCE
1070
UPDATE JTI DATABASE
1080
RETURN REPUTATION DATA TO CLIENT
1090
DONE

FIG. 10A

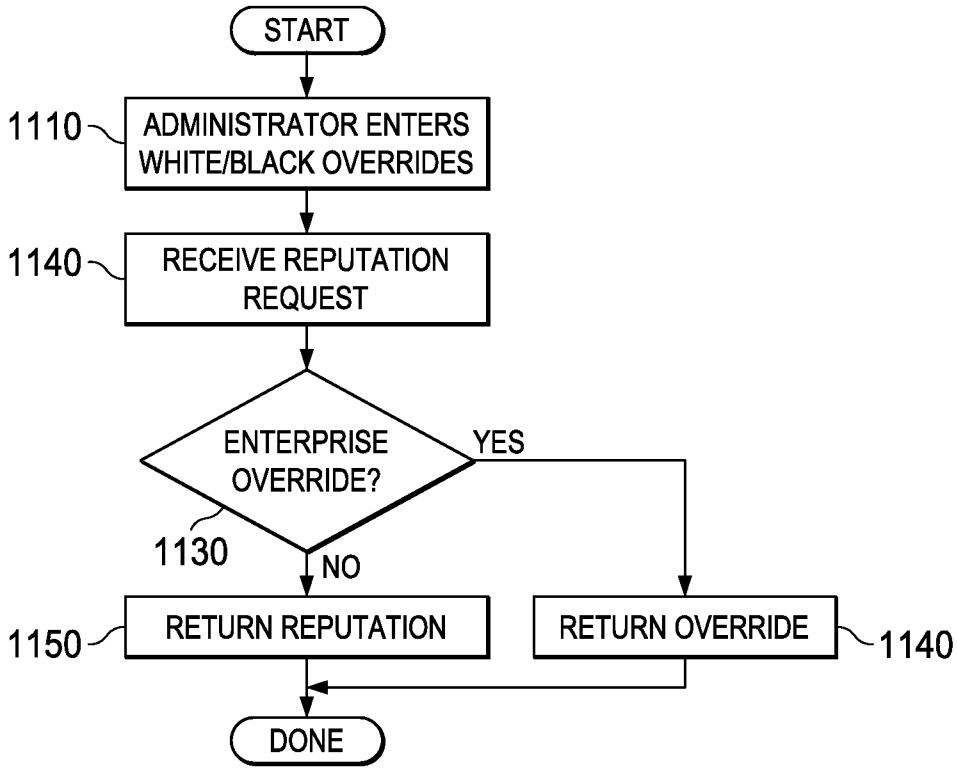

START
1110
ADMINISTRATOR ENTERS WHITE/BLACK OVERRIDES
1140
RECEIVE REPUTATION REQUEST
ENTERPRISE OVERRIDE?
YES
1130
NO
1150
RETURN REPUTATION
RETURN OVERRIDE
1140
DONE

FIG. 11

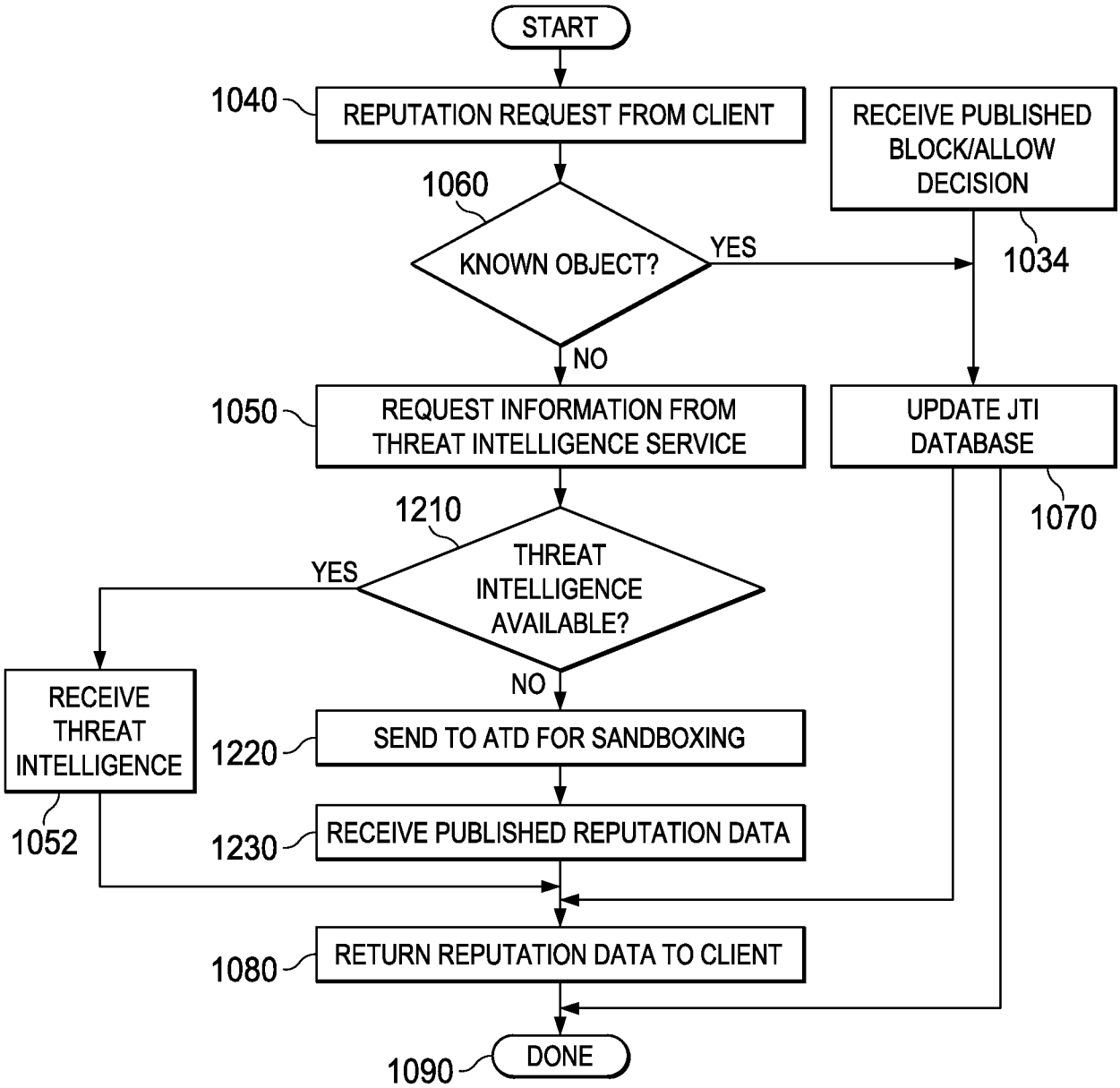

START
1040
REPUTATION REQUEST FROM CLIENT
1060
KNOWN OBJECT?
YES
NO
RECEIVE PUBLISHED BLOCK/ALLOW DECISION
1034
1050
REQUEST INFORMATION FROM THREAT INTELLIGENCE SERVICE
UPDATE JTI DATABASE
1070
1210
THREAT INTELLIGENCE AVAILABLE?
YES
NO
RECEIVE THREAT INTELLIGENCE
1052
1220
SEND TO ATD FOR SANDBOXING
1230
RECEIVE PUBLISHED REPUTATION DATA
1080
RETURN REPUTATION DATA TO CLIENT
1090
DONE

FIG. 12

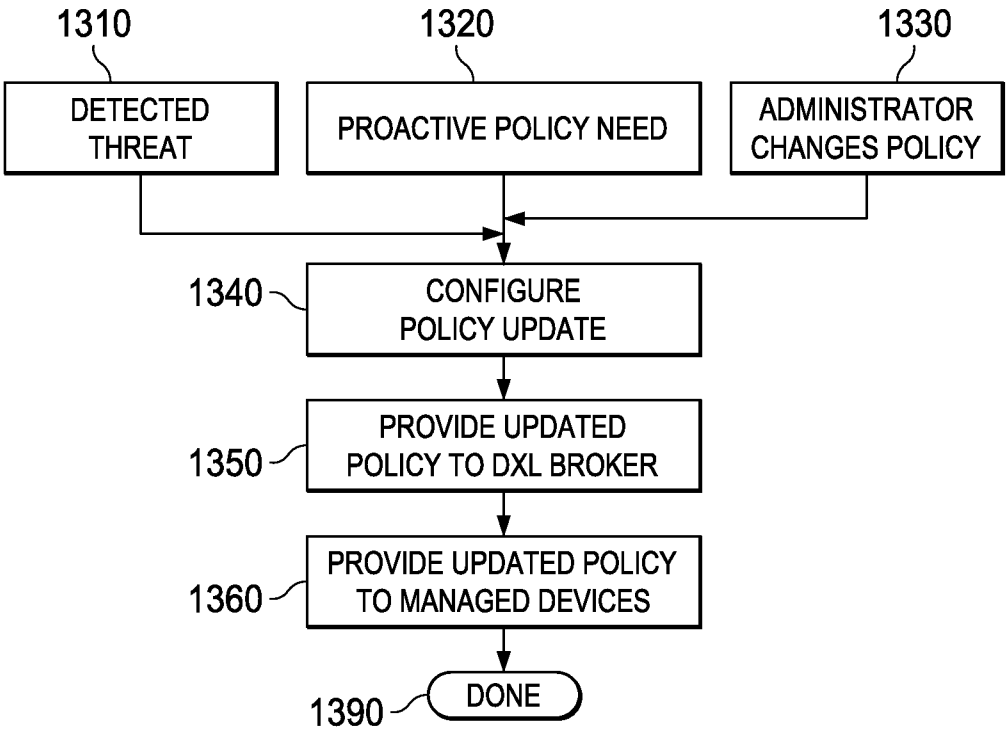

1310
1320
1330
DETECTED THREAT
PROACTIVE POLICY NEED
ADMINISTRATOR CHANGES POLICY
1340
CONFIGURE POLICY UPDATE
1350
PROVIDE UPDATED POLICY TO DXL BROKER
1360
PROVIDE UPDATED POLICY TO MANAGED DEVICES
1390
DONE

FIG. 13

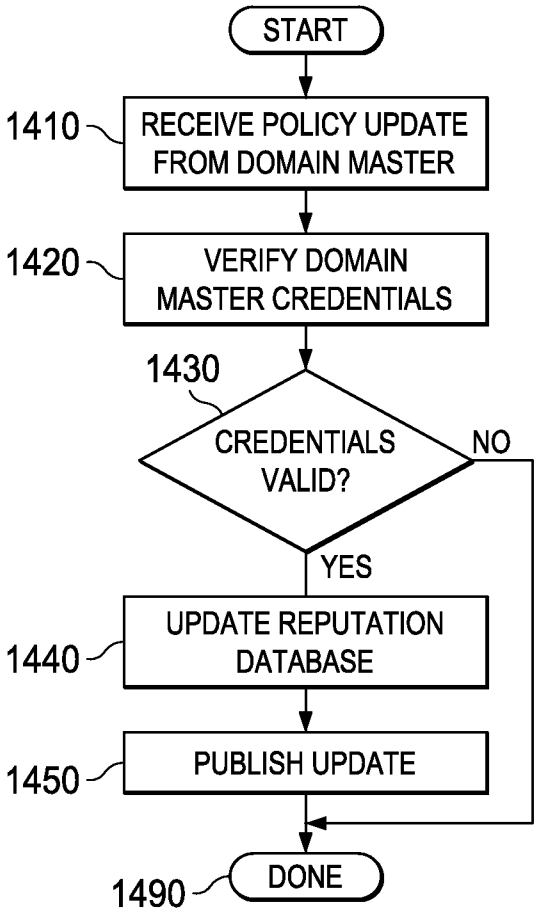

START
1410
RECEIVE POLICY UPDATE FROM DOMAIN MASTER
1420
VERIFY DOMAIN MASTER CREDENTIALS
1430
CREDENTIALS VALID?
NO
YES
1440
UPDATE REPUTATION DATABASE
1450
PUBLISH UPDATE
1490
DONE

FIG. 14

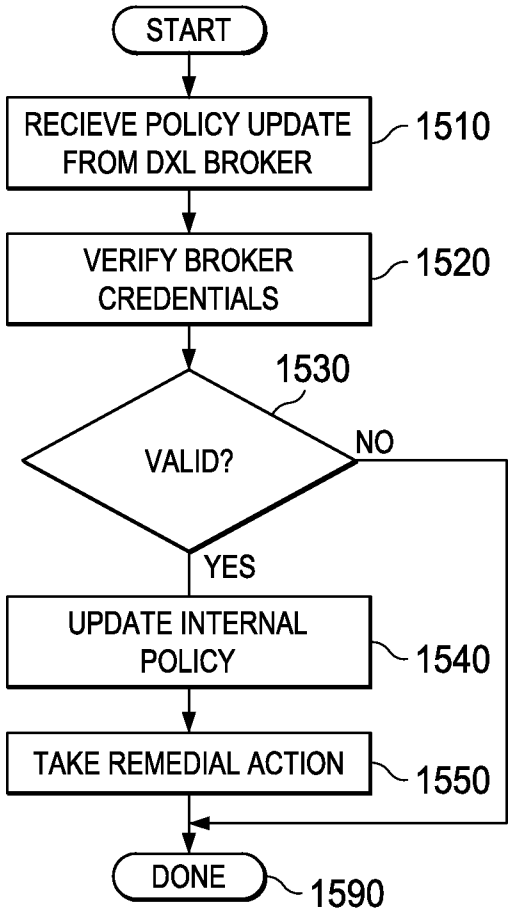

START
RECIEVE POLICY UPDATE FROM DXL BROKER
1510
VERIFY BROKER CREDENTIALS
1520
1530
VALID?
NO
YES
UPDATE INTERNAL POLICY
1540
TAKE REMEDIAL ACTION
1550
DONE
1590

FIG. 15

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/076712

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/62(2013.01)i, G06F 17/30(2006.01)i

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)
G06F 21/62; G06F 15/16; H04K 1/00; G06F 17/30; G06F ; H04L 29/06; H04L 9/00; G06F 21/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & keywords: data exchange layer, policy, update, malware, message, domain master

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2009-0300045 A1 (JAY CHAUDHRY et al.) 03 December 2009 See abstract, paragraphs [0042]-[0047], [0100]-[0107] and figures 2, 5.	1,3,8,10,15,17
A		2,9,16
Y	US 2011-0113105 A1 (CHERYL ECKARDT) 12 May 2011 See paragraphs [0021], [0033]-[0038] and figures 1, 8.	1,3,8,10,15,17
A	US 2006-0143685 A1 (KARAN VASISHTH et al.) 29 June 2006 See paragraphs [0069]-[0072] and figure 4.	1-3,8-10,15-17
A	WO 2004-072800 A2 (PROGRESS SOFTWARE CORP. et al.) 26 August 2004 See paragraphs [0034]-[0041], [0071] and figures 2, 5.	1-3,8-10,15-17
A	US 2012-0102317 A1 (PRIYADARSHEE DEEPTARAG MATHUR et al.) 26 April 2012 See paragraphs [0047]-[0055], [0125]-[0133] and figures 1, 5.	1-3,8-10,15-17

☐ 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
18 June 2014 (18.06.2014)	**18 June 2014 (18.06.2014)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701,
Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
AHN, Jeong Hwan
Telephone No. +82-42-481-8440

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2013/076712

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009-0300045 A1	03/12/2009	CN 102106114 A	22/06/2011
		EP 2283611 A2	16/02/2011
		EP 2283611 A4	20/02/2013
		US 2011-0138468 A1	09/06/2011
		US 2011-0202695 A1	18/08/2011
		US 7899849 B2	01/03/2011
		US 8185510 B2	22/05/2012
		WO 2009-154945 A2	23/12/2009
		WO 2009-154945 A3	01/04/2010
US 2011-0113105 A1	12/05/2011	US 8380797 B2	19/02/2013
US 2006-0143685 A1	29/06/2006	US 2010-0175105 A1	08/07/2010
		US 7607164 B2	20/10/2009
		US 8171522 B2	01/05/2012
WO 2004-072800 A2	26/08/2004	GB 0517847 D0	12/10/2005
		GB 2417160 A	15/02/2006
		GB 2417160 B	20/12/2006
		WO 2004-072800 A3	29/12/2004
US 2012-0102317 A1	26/04/2012	US 2012-102329 A1	26/04/2012
		WO 2012-054899 A2	26/04/2012
		WO 2012-054899 A3	19/07/2012
		WO 2012-054903 A2	26/04/2012
		WO 2012-054903 A3	05/07/2012

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2013/076712

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.: 4-7, 11-14, 18-21
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:

1. ☐ As all required addtional 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 any 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.:

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.