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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 November 2012 (15.11.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/155096 A1

(51) International Patent Classification:
G06F 21/24 (2006.01)

(21) International Application Number:
PCT/US2012/037625

(22) International Filing Date:
11 May 2012 (11.05.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/485,025 11 May 2011 (11.05.2011) US
13/464,906 4 May 2012 (04.05.2012) US

(71) Applicant *(for all designated States except US)*: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, M/S 5OP7, Redwood Shores, California 94065 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **SRINIVASAN, Uppili** [US/US]; 45915 Sentinel Place, Fremont, California 94539 (US). **MOTUKURU, Vamsi** [IN/US]; 22 Raphael Drive, Monmouth Junction, New Jersey 08852 (US). **TURLAPATI, Ramana Rao S.** [US/US]; 34831 Hardwick Place, Fremont, California 94555 (US).

(74) Agents: **NICHOLES, Christian A** et al.; Kilpatrick Townsend & Stockton LLP, Two Embarcadero Center, 8th Floor, San Francisco, California 94111 (US).

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

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

Declarations under Rule 4.17:

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

Published:

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

(54) Title: ACCESS CONTROL ARCHITECTURE

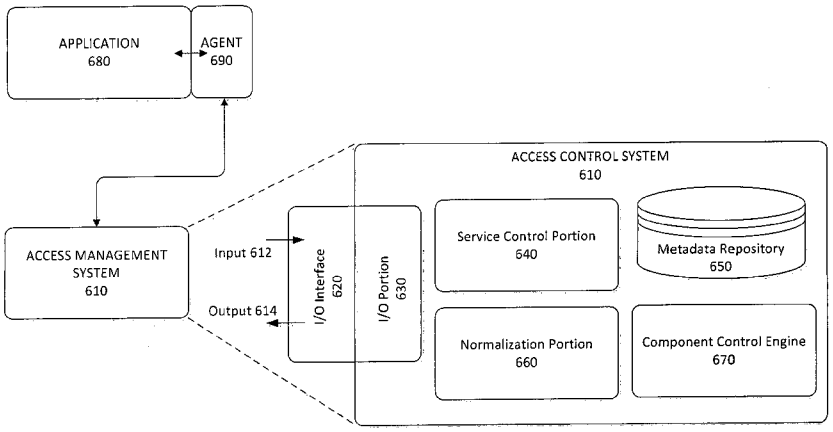


FIG. 6

(57) Abstract: An access control system architecture is provided. In one embodiment, the architecture comprises modular and decoupled components, which allow composability of heterogeneous solutions.

WO 2012/155096 A1

ACCESS CONTROL ARCHITECTURE

BENEFIT CLAIM

[0001] This application claims benefit of Provisional Appln. 61/485,025, filed May 11, 2011, the entire contents of which are hereby incorporated by reference as if fully set forth herein, under 35 U.S.C. §119(e).

FIELD OF THE INVENTION

[0002] The invention relates generally to secured computing, and more generally to access control for computing systems.

BACKGROUND

[0003] Modern businesses rely on a variety of applications and systems that control and generate information that is critical to business operations. Different applications often provide different services and information, and different users may require access to different levels of information within each system or application. The level of access that users are granted may depend on the role of the user. For example, a manager may need access to certain information about employees that report to him, but it may be improper for that manager to access the same information about those whom he reports to.

[0004] Earlier less sophisticated applications incorporated access control business logic directly into the application code. That is to say, each application would require users to have a separate account, separate policy logic, and separate permissions, for example. Furthermore, when a user is authenticated by one of these applications, this authentication remains unknown to other applications in the enterprise because the fact that authentication with the first application has taken place is not shared. Thus, there is no concept of trust between applications using different systems for authentication and access control. Engineers quickly realized that having an access control system for each application in an enterprise was much like having a gas station for each car, and determined that authentication and

access control would be more efficiently implemented and managed as a shared resource. These shared resources became known as an access control systems.

[0005] Access control systems often use policies and other business logic to make a determination regarding whether a particular access request should be granted to a particular resource. Upon making a determination that access should be granted, a token is provided to the requestor. This token is like a key that can be used to open a door that guards restricted data. For example, a user may attempt to access a human resources database to gather information about certain employees such as salary information. The user's web browser makes a request to the application, which requires authentication. If the web browser does not have a token, the user is asked to log in to the access control system. When the user is authenticated, the user's browser receives a cookie that represents a token that may be used to access the human resources application.

[0006] To facilitate the proliferation of access control systems, developers of such systems created agent development kits to allow application developers to easily create agents that are capable of interacting with an access control system on behalf of an application. These agents represent the logic required at the application side, and applications require less code to integrate with and use these agents than they would to include the access control logic directly into the application. However, agents are specific to the access control systems for which the agent toolkit was developed. Therefore, if an enterprise architect or engineer wishes to change the access control system used by a particular application, the agent associated with the application must also be replaced to conform to the requirements of the new access control system. Furthermore, access control systems may have different features, so that one access manager would not offer the services required by the applications in the enterprise, even if the agents were compatible. All of this makes access control systems "sticky," meaning that it is very difficult or cost-ineffective to switch applications from their reliance on one access control system to another.

[0007] As a result of the stickiness of access control systems, many enterprises either use multiple access control systems or use applications that are easily integrated with access control systems that are already employed by the enterprise. Applications are thus organized into "silos," and many of the applications are unable to take advantage of the services offered by access control systems that they are incompatible with. In such an enterprise, it becomes difficult to roll out new access control features, because integration must be performed for each access control system. In addition, users are often confused by the lack of universal

integration, since the demarcation between applications using the same access control system is not easily recognizable to the user.

[0008] In cases where applications are all using the same access control system, enterprise architects are often limited in their choice of applications out of a desire to maintain an existing access control system. Therefore, even if a particular application offers superior features, the architect will often choose a different application that is already compatible with or easily integrated with the access control system that is already deployed within the enterprise, sacrificing the superior features to retain consistency.

[0009] Present access control solutions face several challenges, and existing products have been under pressure to evolve to seamlessly support emerging enterprise and Internet access control requirements. This evolution has been problematic since products use designs that are impractical to extend to accommodate emerging requirements. As a result, addition of new features is often achieved via newer product components posing disruptive and time consuming integration and implementation process.

[0010] When it comes to securing enterprise resources and enabling access to these resources by enterprise users, companies have the freedom to choose and standardize their solution around protocol standards and implementation models that best suit their needs. But when it comes to extranet, there are business motivations to accommodate and attract an expanding base of user communities representing varying levels of trust (such as business partner communities, enterprise customers and general public Internet consumers), and associated interaction models.

[0011] Enterprises have to secure varying types of resources spanning different administrative models and problem domains. On the other hand users in an enterprise often need access to resources across many such domains.

[0012] Driven by regulatory compliance initiatives such as SOX or M&A activity, enterprise Access Control users need to transition from functional product silos to a process oriented environment that consolidates various legacy stacks such that, access control and control can be applied to all IT assets in a consistent manner. But lack of consistent consolidation architectures and migration frameworks often pose insurmountable implementation challenges.

[0013] Access Control vendors have traditionally addressed the above challenges with individual products specialized for specific problem domains. The common elements of

access control solutions, namely, types of tokens used to encode assertions, trust model among the parties involved, and the wire protocol standards are all inherently independent issues. But traditional products tend to create tight coupling among these elements.

[0014] FIG. 1 illustrates an example of different products tailored for different environments, each representing a specific coupling typical to the environment the product is targeted for. Because of the above tight coupling, multiple individual products need to be glued together to address the heterogeneous problem domains. This gluing/bridging ends up being the weakest link in the overall security posture since the individual products bridged are incompatible in their security models and design patterns.

[0015] These independent products are also often based on independent technology stacks, thus compounding the challenge of deploying integrated solutions for customers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] FIG. 1 illustrates an example of different products tailored for different environments;

[0017] FIG. 2 illustrates an access control model according to an embodiment of the present invention;

[0018] FIG. 3 depicts an access control architecture according to an embodiment of the present invention;

[0019] FIG. 4 depicts an access control architecture that enables different access control solutions to be authored or composed in a seamless manner through layering according to an embodiment of the present invention;

[0020] FIG. 5 illustrates a more general diagram of an access control architecture using a layered approach.

[0021] FIG. 6 is a simplified block diagram illustrating an access control system on which an embodiment may be implemented.

[0022] FIG. 7 is a flow diagram illustrating functions of an access control system on which an embodiment may be implemented.

[0023] FIG. 8 is a flow diagram illustrating functions of an access control system on which an embodiment may be implemented.

[0024] FIG. 9 shows an example of a migration pattern that is applicable to an access control product;

[0025] FIG.10 shows an example of a migration pattern that is applicable to an access control product;

[0026] FIG. 11 is a simplified block diagram illustrating physical components of a system environment that may be used in accordance with an embodiment of the present invention; and

[0027] FIG. 12 is a simplified block diagram of a computer system that may be used to practice an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0028] In the following description, for the purposes of explanation, specific details are set forth in order to provide a thorough understanding of embodiments of the invention. However, it will be apparent that the invention may be practiced without these specific details.

GENERAL OVERVIEW

[0029] According to an embodiment of the present invention, an access control architecture is provided that is capable of addressing the above-mentioned challenges. In one embodiment, instead of independent solution silos for different problem domains, the architecture comprises modular and decoupled components, allowing composability of heterogeneous solutions. In particular, the architecture enables composition of heterogeneous programming language implemented technology stacks.

[0030] In an embodiment, the access control architecture described herein enables support for emerging technologies and seamlessly accommodating associated new protocols as they evolve. In an embodiment, composite and heterogeneous solutions that are adaptable to a customer's deployment environment are enabled. An access control solution that supports different wire protocols simultaneously while accommodating varying trust models and provisioning models for different communities, simultaneously and seamlessly is described herein.

[0031] In an embodiment, the access control architecture described herein enables integrated cross-enterprise access control capable of spanning all data and assets of an

organization. In an embodiment, an integrated cross-enterprise access control solution that better manages security risk across the Enterprise is provided. In an embodiment, systematic migration of legacy technologies and consolidated control of legacy and new technology during the period of migration may be enabled.

[0032] In an embodiment, an access metadata repository maintains a repository of access metadata objects that describe data associated with access services. An access request is received from a requesting entity, and a request type associated with the access request is determined. A normalized request is generated, and a first functional component is selected to satisfy at least a portion of the first normalized access request based in part on the normalized access request and an access metadata object associated with the request type. In an embodiment, the first functional component generates at least a portion of a response, which is provided to the requesting entity. In an embodiment, the response is converted into a protocol-specific response before the response is provided to the requesting entity. In an embodiment, a computer-readable non-volatile storage medium comprises code segments which when loaded into one or more computers of a computer system causes the system to carry out the methods described below. In an embodiment, an access control system, comprises an access metadata repository of access metadata objects, wherein each access metadata object of a plurality of access metadata objects in the access metadata repository describes data associated with access services, an input/output portion configured to receive a first access request from a first requesting entity, a service control portion configured to determine a first request type associated with the first access request, a normalization portion configured to generate a first normalized access request, and a component control engine configured to select a first functional component to satisfy at least a portion of the first normalized access request based at least in part on the first normalized access request and an access metadata object associated with the first request type, wherein the access metadata repository, input/output portion, service control portion, normalization portion, and component control engine are implemented on one or more computing devices. In an embodiment, the component control engine is further configured to provide at least a portion of the first normalized access request to the first functional component, and the component control engine is further configured to cause the first functional component to generate at least a portion of a first response that conforms with the first request type associated with the first access request, and the service control portion is further configured to provide at

least the portion of the first response to the first requesting entity. In an embodiment, the first requesting entity is a first application that is a trusted application, the first access request includes a request for a security token associated with a second application, and generating at least a portion of a response includes generating a first token that authorizes the first application to make changes associated with the second application on behalf of a user of the first application. In an embodiment, the input/output portion is further configured to receive a second access request from a second requesting entity, wherein the second access request includes a request for a security token associated with an identity provider, the normalization portion is further configured to generate a second normalized access request, the component control engine is further configured to cause the first functional component to generate a second token that authorizes access to the third application, the first functional component is configured to generate the first token in response to receiving at least a portion of the first normalized request, and the first functional component is configured to generate the second token in response to receiving at least a portion of the second normalized request. In an embodiment, the request includes composite state information that identifies a state associated with a first functional component and a state associated with a second functional component. In an embodiment, the system further comprises an access policy repository configured to store access policy metadata for determining whether a request for access should be granted, and a second functional component configured to generate a response that includes information associated with the access policy metadata to the first requesting entity in response to determining that the first normalized access request does not meet the criteria specified by the access policy metadata. In an embodiment, the input/output portion is further configured to receive a second access request from a second requesting entity, the service control portion is further configured to determine a second request type associated with the second access request, the normalization portion is further configured to generate a second normalized access request, the component control engine is further configured to select the first functional component to satisfy at least a portion of the second normalized access request based at least in part on the second normalized access request and an access metadata object associated with the second request type, the normalization portion is further configured to provide at least a portion of the second normalized access request to the first functional component, and the component control engine is further configured to cause the first functional component to generate at least a portion of a second response

that conforms with the second request type associated with the second access request, and the service control portion is further configured to provide at least the portion of the first response to the first requesting entity. In an embodiment, an access control method, comprises maintaining an access metadata repository of access metadata objects, wherein each access metadata object of a plurality of access metadata objects in the access metadata repository describes data associated with access services, receiving a first access request from a first requesting entity, determining a first request type associated with the first access request, generating a first normalized access request, based at least in part on the first normalized access request and an access metadata object associated with the first request type, selecting a first functional component to satisfy at least a portion of the first normalized access request, wherein the method is performed by one or more computing devices. In an embodiment, the method further comprises providing at least a portion of the first normalized access request to the first functional component, generating at least a portion of a first response using the first functional component, the first response conforming with the first request type associated with the first access request, and providing at least the portion of the first response to the first requesting entity. In an embodiment, the first requesting entity is a first application that is a trusted application, the first access request includes a request for a security token associated with a second application, generating at least a portion of a response includes generating a first token that authorizes the first application to make changes associated with the second application on behalf of a user of the first application. In an embodiment, the method further comprises receiving a second access request from a second requesting entity, wherein the second access request includes a request for a security token associated with an identity provider, generating a second normalized access request, generating a second token that authorizes access to the third application, wherein the first token is generated by a token generation engine in response to receiving at least a portion of the first normalized request at the token generation engine, wherein the second token is generated by the token generation engine in response to receiving at least a portion of the second normalized request at the token generation engine. In an embodiment, the request includes composite state information that identifies a state associated with a first functional component and a state associated with a second functional component. In an embodiment, the method further comprises maintaining an access policy repository that stores access policy metadata for determining whether a request for access should be

granted, in response to determining that the first normalized access request does not meet the criteria specified by the access policy metadata, generating a response to the first requesting entity that includes information associated with the access policy metadata. In an embodiment, the method further comprises receiving a second access request from a second requesting entity, determining a second request type associated with the second access request, generating a second normalized access request, based at least in part on the second normalized access request and an access metadata object associated with the second request type, selecting the first functional component to satisfy at least a portion of the second normalized access request, providing at least a portion of the second normalized access request to the first functional component, generating at least a portion of a second response using the first functional component, the second response conforming with the second request type associated with the second access request, and providing at least the portion of the first response to the first requesting entity. In an embodiment, a system for performing access control comprises means for maintaining an access metadata repository of access metadata objects, wherein each access metadata object of a plurality of access metadata objects in the access metadata repository describes data associated with access services, means for receiving a first access request from a first requesting entity, means for determining a first request type associated with the first access request, means for generating a first normalized access request, means for, based at least in part on the first normalized access request and an access metadata object associated with the first request type, selecting a first functional component to satisfy at least a portion of the first normalized access request. In an embodiment, the system further comprises means for providing at least a portion of the first normalized access request to the first functional component, means for generating at least a portion of a first response using the first functional component, the first response conforming with the first request type associated with the first access request, and means for providing at least the portion of the first response to the first requesting entity. In an embodiment, the first requesting entity is a first application that is a trusted application, the first access request includes a request for a security token associated with a second application, and the means for generating at least a portion of a response includes means for generating a first token that authorizes the first application to make changes associated with the second application on behalf of a user of the first application. In an embodiment, the system further comprises means for receiving a second access request from a second requesting

entity, wherein the second access request includes a request for a security token associated with an identity provider, means for generating a second normalized access request, and means for generating a second token that authorizes access to the third application, wherein the first token is generated by a token generation engine in response to receiving at least a portion of the first normalized request at the token generation engine, wherein the second token is generated by the token generation engine in response to receiving at least a portion of the second normalized request at the token generation engine. In an embodiment,

the request includes composite state information that identifies a state associated with a first functional component and a state associated with a second functional component. In an embodiment, the system further comprises means for maintaining an access policy repository that stores access policy metadata for determining whether a request for access should be granted, and means for, in response to determining that the first normalized access request does not meet the criteria specified by the access policy metadata, generating a response to the first requesting entity that includes information associated with the access policy metadata. In an embodiment, the system further comprises means for receiving a second access request from a second requesting entity, means for determining a second request type associated with the second access request, means for generating a second normalized access request, means for, based at least in part on the second normalized access request and an access metadata object associated with the second request type, selecting the first functional component to satisfy at least a portion of the second normalized access request, means for providing at least a portion of the second normalized access request to the first functional component, means for generating at least a portion of a second response using the first functional component, the second response conforming with the second request type associated with the second access request, and means for providing at least the portion of the first response to the first requesting entity.

LAYERED ACCESS CONTROL ARCHITECTURE

[0033] FIG. 2 illustrates an access control model according to an embodiment of the present invention. The architecture depicted in FIG. 2 depicts a unique multi-layered architecture, with each layer comprising a category of building blocks. In the embodiment

depicted in FIG. 2 the building blocks include: Identity solutions 210, protocol facades 220, token handling engines 230, trust models 240, and shared services 250.

[0034] In an embodiment, a layering model is based on the above building blocks. An orchestration controller component is used to orchestrate messages between the layers that use these building blocks, allowing communication between layers to adapt to the communication capabilities available between the layers according to the needs between layers, rather than hard-coded communication between specific components. This helps to facilitate systematic development of new capabilities and composite solutions.

[0035] An extensible foundation representing a single set of canonical real-world physical entities and mechanisms to map protocol-defined entities to their corresponding canonical representations may be enabled in an embodiment. For example, an external policy engine may be communicatively coupled to the shared services layer of an access control system using the multi-layered access model described in an embodiment, thereby extending the functionality of all services provided by the access model. This capability allows the same physical entity (such as a service, an application client or user) to be part of composite solutions spanning problem domains and protocols.

[0036] In an embodiment, patterns for identifying and packaging shareable functional components are deployable in different form factors (such as embedded, distributed etc.) with the goal of enabling a broad spectrum of customer deployment environments. In addition, a set of dedicated service facades leveraging the same composable architecture, aimed at facilitating co-existence and consolidated control of legacy technologies and while allowing gradual migration of legacy to new technology may be featured in embodiments.

[0037] In an embodiment, the multi-layered access control system utilizes an infrastructure that decouples information type from information life-cycle to enable externalizing and sharing of control and enforcement across Access Control to be externalized and thus allowing Policy decisions and session infrastructure to be decoupled/externalized. This design element facilitates consistent control and enforcement of policies across problem domains and administrative boundaries (Web Single Sign On (SSO) for enterprise resources, ID federation for partners, ID propagation and delegated access to web services, etc.).

[0038] Current access control products are based on a "silo" approach, as shown in FIG. 1. Referring to FIG. 1, silo A 110, silo B 120, silo C, 130, and silo D 140 all comprise a static set of elements that may not be altered. For example, customer portal applications may use

access control solutions that only authenticate using OpenID, and may not utilize the functionality of any silo other than silo D 140. In the approach depicted in FIG. 1, each product is essentially an island. As a result, it is practically impossible to define, manage and enforce security in a consistent manner across the various silos. This fragmentation exposes chinks in the security armor of an enterprise and the growing use of the Internet has significantly increased the potential for enterprise IT assets to be comprised. To solve the above, what is needed is a transition of products from functional silos to compose-able units. This is realized through a new architecture that facilitates the transition in a phased manner for existing solutions while simultaneously delivering broad heterogeneous solutions based on emerging standards.

[0039] FIG. 3 depicts an access control architecture according to an embodiment of the present invention. The embodiment depicted in FIG. 3 provides a multi-protocol, composite, server architecture. In an embodiment, the building blocks are used in conjunction with a layering model, allowing products and composite solutions to function in a manner that is independent of any specific language, platform or technology.

[0040] The functional layering shown in FIG. 3 allows for composing any existing access control (AC) product (existing and future) while ensuring code reuse in an embodiment. Protocol bindings 310 represent the logic that maps authentication protocol messages to standard messaging and transport protocols. Protocol bindings 310 are responsible for marshaling and unmarshalling protocol response and requests respectively. Controller 320 represents the core business logic orchestration needed to fulfill protocol requests by invoking the functional components. Shared services for access (SSA) 330 represents the shared server functionality that is intended to be used by all AM products. It is composed of high-level stateful engines (foundation components) and low-level stateless engines (foundation APIs). Control is passed from the Protocol Binding layer 310 to the Controller 320 to the SSA layer 330.

[0041] In an embodiment, patterns for identification and packaging of shareable functionality make it possible for this functionality to be made available in different form factors to support a broad spectrum of solutions. The architecture focuses on interfaces and layering between layers and the entities/modules that implement the layer. Each layer provides a specific set of functionality with the lower layers providing facilities for use by the layer above. In an embodiment, this layered approach enables composition of functionality

independent of packaging form factors to build composite solutions (for example, as shown in FIG. 4).

[0042] Embodiments facilitate the building of different products by allowing the different entities/modules in the various layers through composition (e.g., as shown in FIG. 4). As a result, new features/functionality and manageability may be introduced in an incremental manner.

[0043] FIG. 4 depicts an architecture that includes various solutions 410, packaged products 420, standalone entities 430, single sign-on authentication (SSA) 440, and other platforms APIs, connectors, and plugins 450. This illustration provides an example of the flexibility offered by a layered approach, rather than a "silo" approach to access control.

[**0044**] FIG. 5 illustrates a more general diagram of an access control architecture using a layered approach. The components of an access control architecture include five types of entities in an embodiment: service entities 510, protocol bindings 520, a controller 530, integrated functional components 540, and functional components with APIs 550.

[**0045**] Service entities 510 represent a set of functional entities packaged together as a standalone software component. Service entity(ies) are capable of being administered and managed by a customer.

[**0046**] Protocol binding layer 520 represents the logic that maps authentication protocol messages to standard messaging and transport protocols. Protocol bindings are is responsible for marshalling and unmarshalling protocol responses and requests.

[**0047**] In an embodiment, protocol binding may perform security processing and wire data optimizations (e.g. MTOM) that depend on the wire ordering of bytes, and does not perform any semantic processing of the protocol request/response. Protocol binding layer 520 leverages the transport and message protocol binding facilities provided by the underlying platform, and may be realized in programming constructs such as handlers, interceptors, plugins and library APIs.

[**0048**] Controller 530 represents the core business logic orchestration needed to fulfill protocol requests by invoking the functional components. In an embodiment, the controller layer 530 is decoupled from the protocol binding layer, and is transport binding agnostic. The controller layer Includes the logic to process the request by invoking the Functional Components (Engines) and Foundation APIs. Product specific business logic is isolated at

the controller layer 530 in an embodiment to enable the functional components to remain unaware of details that are unnecessary to perform their individual functions. Thus, messages passed from the controller layer 530 to the functional component layers 540 and 550 are normalized to include only the information required by each functional compoenent in an embodiment. Thus, product independent/generic logic is pushed to the engines/foundation APIs to promote modularity and code reuse across various access control components.

[0049] Functional components represent units of well defined functionality. Each functional component is tightly coupled and logically separate from other functional components. In an embodiment, component-to-component communication is only allowed through public interfaces. Functional components have an associated configuration that is user visible and is intended to be used in conjunction with other functional entities as part of a bigger service entity in service entities layer 510. A component can be decomposed into sub-components to facilitate modularity and parallel development. Integrated functional components 540 and functional components with APIs 550 differ in their level of integration to the system. For example, an integrated functional component may access an API associated with another functional component in order to extend the functionality of the access control system. This illustrates the extensible nature of a layered access control architecture. The following "engines" represent examples of functional entities that may be implimented in an embodiment:

[0050] Authentication Engine: The authentication engine is responsible for establishing the identity of a user by collecting and validating the credentials of a user using a specific authentication protocol. A User interacts with the controller (View), which delegates the authentication logic (Model + Controller) to the authentication engine. The user interaction for credential collection is performed by the protocol/web/presentation tier components like the controller, agents, and interceptors.

[0051] The authentication engine includes all functions necessary for applying the various authentication schemes in various combinations; supporting authentication models including the policies and control flags required by those models; customizing and extending all aspects of the authentication process including techniques, rules and protocols; and Interacting with the agent tier to drive credential collection.

[0052] Authorization Engine: The authorization engine is Responsible for the process of establishing the identity of a claimant using a specified authentication protocol. User

interaction is performed for credential collection by the protocol/web/presentation tier components like the NG-AM controller, agents, and interceptors.

[0053] In an embodiment, the authorization engine includes the functions necessary for: applying the various authentication schemes in various combinations; supporting the JAAS authentication model including the policies and control flags; customizing and extending all aspects of the authentication process including techniques, rules and protocols; and interacting with the agent tier to drive credential collection.

[0054] In an embodiment, the authorization engine supports existing schemes. Centralized and distributed identity stores are also supported in an embodiment. In an embodiment, the authorization engine supports multiple authentication schemes per resource in a combination with different control flags, and is agnostic of credential collection specifics such as protocol binding and agent environment. The authorization engine may delegate authorization services to remote or external authorization services in an embodiment.

[0055] In an embodiment, the authorization engine supports support impersonation of users by another authenticated user (e.g. administrators or support personal). The authorization engine is also capable of supporting multiple agent interaction models for credential collection in an embodiment.

[0056] In an embodiment, the authorization engine may leverage other Engine(s) and/or Foundation API(s) for needed functionality, such as session persistence, token generation/validation, or any other functionality associated with the system. In addition, the authorization engine supports migration of authentication policies and use-cases in an embodiment.

[0057] SSO Engine: The SSO engine is responsible for providing the single sign-on (SSO) experience to a user. This means that a user may "sign-in" to one application and use that token to access other applications. It accomplishes this by managing the user session lifecycle, which involving facilitates Global logout by orchestrating logout across all RPs in the valid user session.

[0058] The SSO engine supports logout orchestration and client state handling in an embodiment. In addition, the SSO engine provides support for web agents. For example, web agents may be used to "guard" access to web-based applications. When an access request is made to the application, the web agent intercepts the request, and passes the request

to the access control system. After the request is normalized, the normalized request is sent to the SSO engine, which is in charge of persisting and locating client session state.

[0059] The SSO engine supports session indexing, domain scoped and resource scoped trust models, and cookie-based multi-domain and multi-zone SSO in an embodiment. In an embodiment, SSO engine leverages other Engine(s) and/or Foundation API(s) for needed functionality (e.g. session persistence, token generation).

[0060] Federation Engine: The federation engine is responsible for managing account linking with partners as well as providing federated session control services. The federation engine uses federation protocols to enable standards-based SSO across domains. The federation engine also orchistrates logout flows in an SSO environment.

[0061] The federation engine supports multiple federation protocols in an embodiment. Federation data may be stored persistently in a repository such as a database system or other storage. Various authentication mechanisms are supported by the federation engine in an embodiment, including Infocard based authentication. However, other features and functionality may be externalized. For example, security processing may be externalized in an embodiment.

[0062] Token Engine: The token engine is responsible for managing the entire token life cycle for all tokens including generation, validation, cancellation and renewal of security tokens and credentials. It includes both local and external / delegated operations with Native SSO bridges used for external operations. Maintaining a single token control engine (rather than multiple token engines across silos) ensures Improved security, maintainability and elimination of code duplication across products. The token engine also includes the functions for locating/resolving security tokens and credentials of user(s) associated with a token across protocols/components/servers. In an embodiment, Token Processing Foundation API is responsible for handling the mechanics or structural aspects of credential and token processing.

[0063] In an embodiment, the token engine includes Multiple token generation, validation, cancellation, and/or renewal modules for any given token type. In addition, the token engine supports Username, SAML and X.509 (Validation) without requiring integration with or leveraging any existing security products/infrastructures, as these features are contained within the token engine. The token engine supports the ability for clients to submit proof tokens and other identity proofing data as input, and also supports challenge response and

negotiation protocols like SPNEGO that involve multiple interactions in an embodiment. In embodiments, the token engine is capable of leveraging FIPS 140-2 cryptographic modules for token processing, and any X.509 token processing performed by the token engine supports the Federal processing rules as specified by NIST. The token engine also publishes metadata and supports dynamic discovery of engine functionality at runtime. In addition, the token engine is capable of being invoked by J2SE and J2EE clients and is portable across J2EE servers. An extensible plugin model for typed token lifecycle operations may also be supported.

[**0064**] In an embodiment, Structural aspects (building) of token processing are delegated to the token processing foundation API. Additional custom token types and customized token handling may also be available via the token processing foundation API in an embodiment. Security processing of tokens is externalized in an embodiment and may leverage the Partner Trust metadata engine, other Engine(s) and/or Foundation API(s) for needed functionality.

[**0065**] Session Control Engine: The session control engine is responsible for managing user session and token context information with support for user/admin initiated and time-out based events. The session control engine is capable of orchestrating a global lockout through the use of sessions. The session control engine is also responsible for managing tokens that are responsible for authenticating an entity throughout a session. The control of tokens that are used to maintain a session is called token lifecycle control. The session control engine is in charge of creating, updating and deleting user sessions, locating/resolving sessions associated with a token across protocols/components/servers, and creating, updating and deleting token context.

[**0066**] The session control engine may store active sessions in a database or repository. In an embodiment, inactive session information remains available for a configurable time after session expiration. For example, the time for which a session may remain available may be entered via a user interface.

[**0067**] In embodiments, the session control engine may also: be capable of notifying registered listeners on specific session events, support In-Memory and RDBMS session persistence, support distributed in-memory session persistence, be indexed by a Username, GUID and Username-ProviderID, be capable of supporting session indexing.

[**0068**] Card Control Engine: Cards (including iCards) enable people to organize their digital identities and to select one they want to use for any given interaction. The card

control engine is responsible for managing the lifecycle of CardSpace compliant and other card types. It is key functionality needed to support user centric identity models and is invoked by the IDP's controller upon Card related protocol messages.

[0069] Trust Policy Control Engine: The trust policy control engine enables control and access of trust policy information and decisions based on these policies. The trust policy control engine is responsible for the control fo trust relationships with peer entities involved in interactions. Trust policy is directed to trust between entities, and not between users. For example, two applications may establish a trust relationship with one another, enabling those applications to share data with one another.

[0070] The trust policy control engine consists of the functions necessary for: Making trust determinations about interacting entities; Managing the trust relationships; and Resolving/Converting protocol specific entity identifiers to a canonical representation.

[0071] Partner Metadata Control Engine: The partner metadata control engine enables control and access of meta-data related to partners and consolidates metadata for sentralized control for better control and compliance. Invoked upon both admin and protocol operations.

[0072] The partner metadata control engine consists of the functions necessary for creating, updating, and seleting metadata in an embodiment. It may also include functions for locating/retrieving metadata needed to interact with a partner using a specific protocol to access a specific target. The partner metadata control engine maintains a registry for all partner information, and uses a lightweight storage mechanism for high performance. Each metadata object is self-describing, and therefore the partner metadata control engine may store any type of metadata required that conforms to this aspect of the metadata.

[0073] Native SSO Bridges: The native sso bridges performe functions related to active integration with 3rd party SSO/Policy Services that employ SSO Policy Server specific semantics and complexity.

[0074] Foundation APIs 350 represent additional functionality that is available to the access control system via APIs, rather than via direct integration.

STRUCTURAL AND FUNCTIONAL OVERVIEW

[0075] FIG. 6 is a simplified block diagram illustrating an access control system 610 on which an embodiment may be implemented. In the embodiment shown in FIG. 6, access

control system 610 is a collection of entities 630, 640, 650, 660, and 670, each of which may be implemented in logic gates such as software logic gates, hardware logic gates, or any combination thereof. Access control system 610 includes an input/output (I/O) interface 620 in an embodiment. In another embodiment, I/O interface 620 is not part of access control system 610, but is coupled to access control system 610. I/O interface 620 may be configured to couple access control system 610 to a network or a user input device such as a keyboard, mouse, or any other user input device. I/O interface 620 may also be configured to access control system 610 to other devices or means of providing or interpreting signals or data such as an input 612, including a network, display device, or transmission media device capable of transmitting or displaying an output 614. In an embodiment, I/O interface 620 may represent multiple I/O interfaces.

[0076] Input 612 may include input from a web application such as application 680 or an agent such as agent 690 in an embodimnent. Agent 690 may be configured to intercept access requests from a user such as an access request issued from a user's web browser software or other software or hardware. These requests may be directed in whole or in part to access control system 610 in the form of input 612.

[0077] In an embodiment, access control system 610 includes an I/O portion 630 configured to receive input 612 from I/O interface 620. I/O portion 630 may be configured to store input 612 or information associated with input 612 in non-transitory media, such as volatile or non-volatile storage media. For example, I/O portion 623 may include logging logic.I/O portion 623 is communicatively coupled to service control portion 640, metadata repository 650, normalization portion 660, and component control engine 670 in an embodiment.

[0078] In an embodiment, service control portion 640. Service control portion 640 manages service entities associated with service entities layer 510. These service entities may include protocol-specific listeners that listen for requests of a particular type, for example. When a request is received by a service entity, that request is passed to normalization portion 660 in an embodiment.

[0079] In an embodiment, access control system 610 includes normalization portion 660.Normalization portion 660 impliments the features discussed above with respect to the protocol binding layer 520. Normalization portion 660 receives protocol-specific requests from service control portion 640 in an embodiment. Normalization portion 660 then normalizes the request by providing only non-protocol-specific information to component control engine. In addition, normalization portion 660 may provide metadata frommetadata

repository 650 to component control engine 670 in order to describe the response detailes expected from component control engine 670 in response to the normalized request. As used herein, the term "normalized request" is a request that is different than the original request on which the normalized request is based." metadata repository 650 is used to store metadata and other data in an embodiment.

[0080] In an embodiment, access control system 610 includes component control engine 670. Component control engine 670 orchistrates the performance of all activities associated with incoming requests. Specifically, component control engine determines, based on the information provided by normalization portion, which functional components 540 and 550 are required to satisfy a request, and directs those components to perform their respective functions.

EXAMPLE CONFIGURATION METADATA

[0081] In an embodiment, normalization portion 660 provides configuration metadata to component control engine 670. The configuration metadataand other metadata may be referred to as access metadat objects. This metadata is stored in metadata repository 650 or other tangible storage media in an embodiment. The purpose of the metadata is to provide details to component control engine 670 that helps component control engine 670 fulfill and orchistrate requests. Specifically, each metadata object may represent a particular type of request, and include instructions to component control engine 670 regarding how to handle the request, including which functional components should be used to honor a request and what type of response is expected by normalization portion 660 at the protocol bindings layer 520.

[0082] Athough any configuration metadata format or data source may be used to accomplish the communication between layers described above, the following XML code represents a sample configuration schema that may be used in an embodiment:

```
<Configuration>
     <Providers>
       <!- Describes how to customize processing
       <!- Describes how to override the default logic
        <Provider>
          <!- Describe a Provider that provides
              specific business logic or security functionality
         </Provider>
     </Providers>
      <Setting>
          <!- Describes how to hook into my surrounding infrastructures
          <!- Describes message exchange semantics
```

```
            <!- Describes the generic functionality
            <!- Describes capabilities and how to implement them
            <Setting Type="">
                  …. Any <Setting> or parent <Setting> defined information
            </Setting>
      </Setting>
      <SettingHandlers>
        <SettingHandler>  Responsible for processing a specific
          <Setting> Type
          </Setting>
       </SettingHandler>
 </SettingHandlers>
 </Configuration>
```

[0083] A more specific example that uses this configuration schema follows:

```
<Configuration Version="1.0" ProductName="Oracle NG-SSO"
 ProductVersion="11gR2" ReleaseVersion="11.2.1.0.0"
 Description="The Oracle NG-SSO Product Configuration>

<Providers>
            <Provider Name="DefaultAuthN" Version="1.0"
                  Type="Authentication" SubType="Step"
                  ImplementationClass="oracle...MyAuthNProviderImpl">
            </Provider>
            <Provider Name="CredMapperPlugin" Version="1.0"
                  Type="Authentication" SubType="Plugin"
                  SubSubType="CredentialMapper"
                  ImplementationClass="oracle...MyCredMapperImpl">
            </Provider>
            <Provider Name="DefaultAuthz" Version="1.0"
                  Type=Authorization SubType="Jaas-Oz"
                   ImplementationClass="oracle...MyAuthzImpl">
            </Provider>
            <Provider Name="MyClaims" Version="1.1"
                  Type="ClaimsProvider" SubType=""
                  ImplementationClass=""><Provider>
            </Provider>
</Providers>
<Setting Name="MyPartnerTrustManager">
            <Setting Name="Partner.1" Type=Map>
                  <Setting Name=URI Type=URI>http://abc.com</setting>
                  <Setting Name=Keys Type=string>Asymmetric</Setting>
                  <Setting Name=CertValues Type=string>abc</Setting>
            </Setting>
            <Setting Name=AdminPassword Type=Password">%$#^&</Setting>
      </Setting>
</Setting>
<SettingHandlers>
  <SettingHandler Name="Map" Type="Map" Class="oracle….MapHandler">
 <SettingHandler Name="Boolean" Type="Boolean"
Class="oracle….BooleanHandler">
 <SettingHandler Name="Keystore" Type="KeyStore"
Class="oracle….KeyStoreHandler">
 <SettingHandler Name="String" Type="String" Class="oracle….StringHandler">
 <SettingHandler Name="Path" Type="Path" Class="oracle….PathHandler">
 <SettingHandler Name="Enum" Type="Enum" Class="oracle….EnumHandler">
 <SettingHandler Name="Password" Type="Password"
  Class="oracle….PasswdHandler">
 <SettingHandler Name="URI" Type="String" Class="oracle….URIHandler">
```

```
</SettingHandlers>
</Configuration>
```

[0084] The information to be captured by the configuration is more important than the form of the configuration information. In other words, it is less important that the configuration be implimented in XML as shown above, and more important that enough information is captured in the configuration file to be instructive to component control engine 670. The information that may be captured in embodiments includes product behavior, product bindings, product backend integration, and product extensions.

[0085] Examples of product behavior include authorization, metadata & trust model information, including authorization information such as Windows native auth/ username/X.509/, and other common functionality such as debug/timeout/audit/throttling information. Examples of product bindings include security information such as HTTP basic/digest, OAP (Oracle Access Protocol)/SAML/OpenID Simple Sign, Infocard/Higgins information. Bindings may also include non security information such as metadata, HTTP transport, message encoding, or other similar information. Product backend integration information may include information about user data stores, partner trust, ID admin and access control infrastructures, and audit, logging, and monitoring information. Product extensions may include bindings, integration, and behavior extensions.

HANDLING ACCESS CONTROL REQUESTS

[0086] FIG. 7 is a flow diagram illustrating functions of an access control system on which an embodiment may be implemented. At step 710, a request is received. For example, the request may be a request from an application such as application 680 or an agent such as agent 690. The request may include a request for authentication, authorization, or any other service supported by functional component layers 540 and 550. The request may be received by access control system 610 as input 612 via I/O interface 620 in an embodiment. For example, I/O interface 620 may provide the request to I/O portion 630 for processing in an embodiment.

[0087] At step 720, a normalized request is generated. For example, normalization portion 660 generates an altered version of the request that may include more, less, or different information than the original request. The request may include protocol-specific information that is not included in the normalized request in an embodiment.

[0088] At step 730, a functional component is selected to satisfy at least a portion of the normalized request. For example, if the request is a request for a token, the token engine may be selected by component control engine 670 as the functional component to satisfy the request. If the request requires additional services from other functional components, those components may also be selected to participate in honoring the request by component control engine 670 in an embodiment.

[0089] FIG. 8 is a flow diagram illustrating more specific functions of an access management system on which an embodiment may be implemented. These specific functions are associated with steps 720 and 730 of FIG. 7.

[0090] At step 810, a determiniation of what type of request has been received is made by service control portion 640. For example, the request may be a web SSO request. At step 820, the normalized request is generated based on the request type. For example, the normalized request may include a portion of the web SSO request, without protocol-specific information. At step 830, the normalized request is sent to component control engine 670, along with acess metadata from metadata repository 650 that describes the information needed to orchistrate the generation of a response to the request.

[0091] At step 840, the normalized request is received from normalization portion 660 at component control engine 670. The request may or may not include access metadata. If the request does not include access metadata, access metadata may be retrieved from metadata repository 650 in an embodiment. At step 850, a functional component is selected to satisfy at least a portion of the normalized request based on the normalized request and the access metadata. At step 860, the selected component generates at least a portion of a response to the normalized request.

[0092] In an embodiment, the response that is generated in response to the normalized request is provided by componnt control portion 670 to normalization portion 660, which then provides a complete response to service control portion 640. The complete response may include several partial responses that were generated by different functional components in an embodiment. For example, a response may include authentication and authorization information, alonge with a token, all generated by different functional components. The response may be sent to the requesting entity using the expected protocol-specific format, as protocol-specific information is provided by component control engine based on the configuration metadata.

[0093] In an embodiment, the requesting entity may be an application or an agent that performs services for an application. For example, an agent may intercept certain types of requests such as authorization requests, and handle those requests by communicating with the access control system 610. The request may be a request for a token that will allow the application to access a second application. In response to the request, the token engine may generate a tokenthat authorizes the first application to make changes associated with the second application on behalf of a user of the first application.

[0094] In an embodiment, a request may include a request for a token associated with an identity provider. Normalization portion 660 generates a normalized access request, which is provided to component control engine 670, which selects the token engine to satisfy the request. Thus, the same token engine that was used to generate a token for the first request is also used to generate a token for a second, different type of request.

[0095] In an embodiment, some or all functional engines retain some state. That state is provided to each functional engine in the form of a token or cookie sometimes or every time the server is accessed. For example, the user may access a resource and establishes the cookie with the functional component that provides access to the resource. The cookie is given back to access control system 610 when another access occurrs. In an embodiment, multiple functional engines may be assoicated with a portion of the same state information (i.e., the same cookie).This state information comes through the protocol binding layer, and is then sent to the controller. The controller provides state information access to all of the functional engines. In an embodiment, the request includes composite state information that identifies a state associated with a first functional component and a state associated with a second functional component.

[0096] In an embodiment, an access policy repository is configured to store access policy metadata for determining whether a request for access should be granted. In an embodiment, a policy control functional component is configured to generate a response that includes information associated with the access policy metadata to the first requesting entity in response to determining that the first normalized access request does not meet the criteria specified by the access policy metadata. For example, the metadata may specify an error message associated with a policy failure. This error message may be returned in response to determining that the request does not meet criteria specified by the access policy metadata.

[0097] An access control policy is a deliberate plan of action to guide decisions and achieve intended access control outcomes. An access control policy is applicable to the process of authentication, authorization and auditing. Authentication policies are used to establish the identity of the requestor, while authorization policies are used to determine whether or not to allow the requestor to perform a requested action on a target.

[0098] There are at least two types of policy enforcement. These include information based policy enforcement and decision service based policy enforcement. In information based policy enforcement, policy evaluation is performed by the enforcement point. In addition, policy object types and schemas are published, and the enforcement point is interested and aware of the structure and representation of the policies. In decision based policy enforcement, policy evaluation is outsourced by the Policy Enforcement Point to the decision service. In addition, the enforcement point is interested only in the decision, and is not interested in or aware of the structure or represntation of the policies used for enforcement.

[0099] In an embodiment, policy enforcement may be implimented in an access control solution using the layered approach described herein. For example, a policy engine may be implimented as an integrated functional component or a functional component with an API in embodiments. In an embodiment, policy decisions associated with authorization, risk, fraud, and trust brokering are contemplated for a wide variety of information access and services. Any policies may be implimented because of the plug-in friendly layered approach in an embodiment, but the following have been specifically considered: AuthN Rules – Credential Collection Rules, Chaining Rules, Forced AuthN; SSO – Purpose, Scope, Session Lifetime; I-Card – Types, Claims, Data Handling; Trust – Issuers, Recipients; Federation – Account Linking Rules, Profile Support, NameID Selection; Message Formatting – Use of Signing/Encryption, Encoding andOptimization; Auditing and Logging – Mapping of Levels to Resources; Partner – Use of Metadata; and Security Tokens – Issuance/Exchange, Token Renewal, Cancellation, Delegation and Impersonation Rules.

CANONICAL MIGRATION FRAMEWORK

[0100] Conventional solutions support migration through a process that either requires downtime for full upgrade or support coexistence features in combination with an extensive product certification process to ensure that two given versions of a AM component/product can work together. The certification process is not scalable since it is very time consuming

and expensive. Being mission critical systems, the downtime approach is a non-starter for AM components/products. Thus, a new migration architecture is provided that facilitates a deployment model and design pattern enabling different versions of AM components/products to coexist is needed to address the zero downtime requirements. In addition, the ability to isolate the coexistence modules such that they can be removed/decommissioned later when they are not needed any more is desirable.

[0101] More specifically, in one embodiment, a canonical migration architecture is provided that allows existing and legacy access control solutions to be migrated in a phased and minimally disruptive manner. It also decouples client and server types and versions from each other thereby allowing mixing-n-matching legacy and newer clients and/or servers in existing deployments. This allows for future-proofing a customer's current investment in a specific version of an Access Control product by allowing point upgrades of individual pieces/products with the assurance that existing infrastructure will continue to work.

[0102] FIG. 9 and FIG. 10 show examples of two migration patterns that are applicable to an access control product. In an embodiment, the access control system 610 facilitates incremental migration of AM deployments while ensuring availability of the AM deployment during the migration process, which is critically necessary since any downtime of the AM deployment results in Enterprise Apps between inaccessible.

[0103] In FIG. 9, customers want to use the expressiveness of the next generation SSO policies for a subset of the legacy apps. In an embodiment, customers will need to migrate legacy policies to the new NG-SSO/current solution policies using the policy data migration adapters. The legacy policy server will delegate to the above policy processing to the co-existence proxy via the plugin adapter. For example, a browser 905 may connect to a legacy policy enforcement point 910 to request access to relying party 920. In response the legacy policy enforcement point 910 communicates with a protocol adapter 940 that interacts with coexistence proxy 950. Coexistence proxy 950 accesses the NG-SSO server 945. Meanwhile, legacy policies 935 may be migrated to NG-SSO server 945 using a policy data migration adapter 930. When the NG-SSO policy enforcement point 915 is associated with a relying party such as relying party 925, interaction may be made directly with the NG-SSO server 945. Support for existing policy enforcement points may remain until the migration is complete in order to facilitate a less disruptive migration to the NG-SSO server 945.

[0104] In FIG. 10, Customers migrate legacy policies to NG-SSO policies using the policy data migration adapter. Customers want to continue using legacy PEPs and migrate to NG-

SSO PEPs in an incremental manner. Legacy PEPs now communicate with the Coexistence Proxy via the Protocol Adapter which acts as a protocol bridge between the legacy and NG-SSO Servers. For example, a browser 1005 may connect to a legacy policy enforcement point 1010 to request access to relying party 1020. In response the legacy policy enforcement point 1010 communicates with a policy server 1030, which then communicates with plug-in adapter 1035. Plug-in adapter 1035 interacts with coexistence proxy 1045, and coexistence proxy 950 accesses the NG-SSO server 945. Meanwhile, legacy policies may be migrated from policy server 1030 to NG-SSO server 1040 using a policy data migration adapter 1050. When the next generation single sign on policy enforcement point 1015 is associated with a relying party such as relying party 1025, interaction may be made directly with the NG-SSO server 1040. Support for existing policy enforcement points may remain until the migration is complete in order to facilitate a less disruptive migration to the NG-SSO server 1040.

[0105] An embodiment of the present invention provides a single product that is capable of supporting the combined feature set of the multiple Access Control products. This architecture enables convergence of existing products are based on different centers, have incompatible architectures and are built using different languages and technologies. In an embodiment, an access control architecture improves scalability and maintainability. Beneficially, certain embodiments discussed provide a layering model and an orchestration controller component that facilitates systematic development of new capabilities and composite solutions. Beneficially, certain embodiments of the invention provide an extensible foundation representing a single set of canonical real-world physical entities and mechanisms to map protocol-defined entities to their corresponding canonical representations, thereby allowing the same physical entity (such as a service, an application client, or user) to be part of composite solutions spanning problem domains and protocols. Beneficially, certain embodiments provide patterns for identifying and packaging shareable functional components to be deployable in different form factors (such as embedded, distributed, etc.), thereby enabling a broad spectrum of customer deployment environments. Beneficially, certain embodiments provide a set of dedicated service facades leveraging the same composable architecture, facilitating co-existence and consolidated management of legacy technologies while allowing gradual migration of legacy to new technology. Beneficially, certain embodiments provide an infrastructure that decouples information type from information life-cycle to enable externalizing and sharing of management and enforcement across the access control system to be externalized, thus allowing policy decisions and session infrastructure to be decoupled and externalized. Beneficially, certain

embodiments facilitate consistent management and enforcement of policies across problem domains and administrative boundaries (e.g., Web SSO for enterprise resources, ID federation for partners, ID propagation and delegated access to web services, etc.).

HARDWARE OVERVIEW

[0106] Fig. 11 is a simplified block diagram illustrating physical components of a system environment 1100 that may be used in accordance with an embodiment of the present invention. This diagram is merely an example, which should not unduly limit the scope of the claims. One of ordinary skill in the art would recognize many variations, alternatives, and modifications.

[0107] As shown, system environment 1100 includes one or more client-computing devices 1102, 1104, 1106, 1108 communicatively coupled with a server computer 1110 via a network 1112. In one set of embodiments, client-computing devices 1102, 1104, 1106, 1108 may be configured to run one or more components of a graphical interface described above.

[0108] Client-computing devices 1102, 1104, 1106, 1108 may be general purpose personal computers (including, for example, personal computers and/or laptop computers running various versions of Microsoft Windows and/or Apple Macintosh operating systems), cell phones or PDAs (running software such as Microsoft Windows Mobile and being Internet, e-mail, SMS, Blackberry, and/or other communication protocol enabled), and/or workstation computers running any of a variety of commercially-available UNIX or UNIX-like operating systems (including without limitation the variety of GNU/Linux operating systems). Alternatively, client-computing devices 1102, 1104, 1106, and 1108 may be any other electronic devices capable of communicating over a network (e.g., network 1112 described below) with server computer 1110. Although system environment 1100 is shown with four client-computing devices and one server computer, any number of client-computing devices and server computers may be supported.

[0109] Server computer 1110 may be a general-purpose computer, specialized server computer (including, e.g., a LINUX server, UNIX server, mid-range server, mainframe computer, rack-mounted server, etc.), server farm, server cluster, or any other appropriate arrangement and/or combination. Server computer 1110 may run an operating system including any of those discussed above, as well as any commercially available server operating system. Server computer 1110 may also run any of a variety of server applications and/or mid-tier applications, including web servers, Java virtual machines, application

servers, database servers, and the like. In various embodiments, server computer 1110 is adapted to run one or more Web services or software applications that provide the diagnostics functionality described above. For example, server computer 1110 may be configured to execute the various methods described in the various flowcharts described above.

[0110] As shown, client-computing devices 1102, 1104, 1106, 1108 and server computer 1110 are communicatively coupled via network 1112. Network 1112 may be any type of network that can support data communications using any of a variety of commercially-available protocols, including without limitation TCP/IP, SNA, IPX, AppleTalk, and the like. Merely by way of example, network 1112 may be a local area network (LAN), such as an Ethernet network, a Token-Ring network and/or the like; a wide-area network; a virtual network, including without limitation a virtual private network (VPN); the Internet; an intranet; an extranet; a public switched telephone network (PSTN); an infrared network; a wireless network (e.g., a network operating under any of the IEEE 802.11 suite of protocols, the Bluetooth protocol known in the art, and/or any other wireless protocol); and/or any combination of these and/or other networks. In various embodiments, the client-computing devices 1102, 1104, 1106, 1108 and server computer 1110 are able to access the database 1114 through the network 1112. In certain embodiments, the client-computing devices 1102, 1104, 1106, 1108 and server computer 1110 each has its own database.

[0111] System environment 1100 may also include one or more databases 1114. Database 1114 may correspond to an instance of integration repository as well as any other type of database or data storage component described in this disclosure. Database 1114 may reside in a variety of locations. By way of example, database 1114 may reside on a storage medium local to (and/or resident in) one or more of the computers 1102, 1104, 1106, 1108, 1110. Alternatively, database 1114 may be remote from any or all of the computers 1102, 1104, 1106, 1108, 1110 and/or in communication (e.g., via network 1112) with one or more of these. In one set of embodiments, database 1114 may reside in a storage-area network (SAN) familiar to those skilled in the art. Similarly, any necessary files for performing the functions attributed to the computers 1102, 1104, 1106, 1108, 1110 may be stored locally on the respective computer and/or remotely on database 1114, as appropriate. In one set of embodiments, database 1114 is a relational database, such as Oracle 11g available from Oracle Corporation that is adapted to store, update, and retrieve data in response to SQL-formatted commands. In various embodiments, database 1114 stores data that is used for providing diagnostic capabilities as described above.

[0112] Fig. 12 is a simplified block diagram of a computer system that may be used to practice an embodiment of the present invention. Computer system 1200 may serve as a processing system 102 depicted in Fig. 1. In various embodiments, computer system 1200 may be used to implement any of the computers 1102, 1104, 1106, 1108, 1110 illustrated in system environment 1100 described above. As shown in Fig. 12, computer system 1200 includes a processor 1202 that communicates with a number of peripheral subsystems via a bus subsystem 1204. These peripheral subsystems may include a storage subsystem 1206, comprising a memory subsystem 1208 and a file storage subsystem 1210, user interface input devices 1212, user interface output devices 1214, and a network interface subsystem 1216.

[0113] Bus subsystem 1204 provides a mechanism for letting the various components and subsystems of computer system 1200 communicate with each other as intended. Although bus subsystem 1204 is shown schematically as a single bus, alternative embodiments of the bus subsystem may utilize multiple busses.

[0114] Network interface subsystem 1216 provides an interface to other computer systems, networks, and portals. Network interface subsystem 1216 serves as an interface for receiving data from and transmitting data to other systems from computer system 1200.

[0115] User interface input devices 1212 may include a keyboard, pointing devices such as a mouse, trackball, touchpad, or graphics tablet, a scanner, a barcode scanner, a touch screen incorporated into the display, audio input devices such as voice recognition systems, microphones, and other types of input devices. In general, use of the term "input device" is intended to include all possible types of devices and mechanisms for inputting information to computer system 1200.

[0116] User interface output devices 1214 may include a display subsystem, a printer, a fax machine, or non-visual displays such as audio output devices, etc. The display subsystem may be a cathode ray tube (CRT), a flat-panel device such as a liquid crystal display (LCD), or a projection device. In general, use of the term "output device" is intended to include all possible types of devices and mechanisms for outputting information from computer system 1200.

[0117] Storage subsystem 1206 provides a non-transitory computer-readable storage medium for storing the basic programming and data constructs that provide the functionality of the present invention. Software (programs, code modules, instructions) that when executed by a processor provide the functionality of the present invention may be stored in

storage subsystem 1206. These software modules or instructions may be executed by processor(s) 1202. Storage subsystem 1206 may also provide a repository for storing data used in accordance with the present invention. Storage subsystem 1206 may comprise memory subsystem 1208 and file/disk storage subsystem 1210.

[0118] Memory subsystem 1208 may include a number of memories including a main random access memory (RAM) 1218 for storage of instructions and data during program execution and a read only memory (ROM) 1220 in which fixed instructions are stored. File storage subsystem 1210 provides persistent (non-volatile) storage for program and data files, and may include a hard disk drive, a floppy disk drive along with associated removable media, a Compact Disk Read Only Memory (CD-ROM) drive, an optical drive, removable media cartridges, and other like storage media.

[0119] Computer system 1200 can be of various types including a personal computer, a portable computer, a workstation, a network computer, a mainframe, a kiosk, a server or any other data processing system. Due to the ever-changing nature of computers and networks, the description of computer system 1200 depicted in Fig. 12 is intended only as a specific example for purposes of illustrating the preferred embodiment of the computer system. Many other configurations having more or fewer components than the system depicted in Fig. 12 are possible.

WHAT IS CLAIMED IS:

1. A access control system, comprising:

an access metadata repository of access metadata objects, wherein each access metadata object of a plurality of access metadata objects in the access metadata repository describes data associated with access services;

input/output portion configured to receive a first access request from a first requesting entity;

service control portion configured to determine a first request type associated with the first access request;

normalization portion configured to generate a first normalized access request; and

component control engine configured to select a first functional component to satisfy at least a portion of the first normalized access request based at least in part on the first normalized access request and an access metadata object associated with the first request type;

wherein the access metadata repository, input/output portion, service control portion, normalization portion, and component control engine are implemented on one or more computing devices.

2. The system of Claim 1, wherein:

the component control engine is further configured to provide at least a portion of the first normalized access request to the first functional component; and

the component control engine is further configured to cause the first functional component to generate at least a portion of a first response that conforms with the first request type associated with the first access request; and

the service control portion is further configured to provide at least the portion of the first response to the first requesting entity.

3. The system of any of Claims 1-2, wherein:

the first requesting entity is a first application that is a trusted application;

the first access request includes a request for a security token associated with a second application; and

generating at least a portion of a response includes generating a first token that authorizes the first application to make changes associated with the second application on behalf of a user of the first application.

4. The system of Claim 3, wherein:

the input/output portion is further configured to receive a second access request from a second requesting entity, wherein the second access request includes a request for a security token associated with an identity provider;

the normalization portion is further configured to generate a second normalized access request;

the component control engine is further configured to cause the first functional component to generate a second token that authorizes access to the third application;

wherein the first functional component is configured to generate the first token in response to receiving at least a portion of the first normalized request; and

wherein the first functional component is configured to generate the second token in response to receiving at least a portion of the second normalized request.

5. The system of any of Claims 1-4, wherein the request includes composite state information that identifies a state associated with a first functional component and a state associated with a second functional component.

6. The system of any of Claims 1-5, further comprising:

an access policy repository configured to store access policy metadata for determining whether a request for access should be granted;

a second functional component configured to generate a response that includes information associated with the access policy metadata to the first requesting entity in response to determining that the first normalized access request does not meet the criteria specified by the access policy metadata.

7. The system of any of Claims 1-6, wherein:

the input/output portion is further configured to receive a second access request from a second requesting entity;

the service control portion is further configured to determine a second request type associated with the second access request;

the normalization portion is further configured to generate a second normalized access request;

the component control engine is further configured to select the first functional component to satisfy at least a portion of the second normalized access request based at least in part on the second normalized access request and an access metadata object associated with the second request type;

the normalization portion is further configured to provide at least a portion of the second normalized access request to the first functional component; and

the component control engine is further configured to cause the first functional component to generate at least a portion of a second response that conforms with the second request type associated with the second access request; and

the service control portion is further configured to provide at least the portion of the first response to the first requesting entity.

8. A access control method, comprising:

maintaining an access metadata repository of access metadata objects, wherein each access metadata object of a plurality of access metadata objects in the access metadata repository describes data associated with access services;

receiving a first access request from a first requesting entity;

determining a first request type associated with the first access request;

generating a first normalized access request;

based at least in part on the first normalized access request and an access metadata object associated with the first request type, selecting a first functional component to satisfy at least a portion of the first normalized access request;

wherein the method is performed by one or more computing devices.

9. The method of Claim 8, further comprising:

providing at least a portion of the first normalized access request to the first functional component;

generating at least a portion of a first response using the first functional component, the first response conforming with the first request type associated with the first access request;and

providing at least the portion of the first response to the first requesting entity.

10. The method of any of Claims 8-9, wherein:

the first requesting entity is a first application that is a trusted application;

the first access request includes a request for a security token associated with a second application;

generating at least a portion of a response includes generating a first token that authorizes the first application to make changes associated with the second application on behalf of a user of the first application.

11. The method of Claim 10, further comprising:

receiving a second access request from a second requesting entity, wherein the second access request includes a request for a security token associated with an identity provider;

generating a second normalized access request;

generating a second token that authorizes access to the third application;

wherein the first token is generated by a token generation engine in response to receiving at least a portion of the first normalized request at the token generation engine;

wherein the second token is generated by the token generation engine in response to receiving at least a portion of the second normalized request at the token generation engine.

12. The method of any of Claims 8-11, wherein the request includes composite state information that identifies a state associated with a first functional component and a state associated with a second functional component.

13. The method of any of Claims 8-12, further comprising:

maintaining an access policy repository that stores access policy metadata for determining whether a request for access should be granted;

in response to determining that the first normalized access request does not meet the criteria specified by the access policy metadata, generating a response to the first requesting entity that includes information associated with the access policy metadata.

14. The method of any of Claims 8-13, further comprising:

receiving a second access request from a second requesting entity;

determining a second request type associated with the second access request;

generating a second normalized access request;

based at least in part on the second normalized access request and an access metadata object associated with the second request type, selecting the first functional component to satisfy at least a portion of the second normalized access request;

providing at least a portion of the second normalized access request to the first functional component;

generating at least a portion of a second response using the first functional component, the second response conforming with the second request type associated with the second access request; and

providing at least the portion of the first response to the first requesting entity.

15. A system for performing access control, comprising:

means for maintaining an access metadata repository of access metadata objects, wherein each access metadata object of a plurality of access metadata objects in the access metadata repository describes data associated with access services;

means for receiving a first access request from a first requesting entity;

means for determining a first request type associated with the first access request;

means for generating a first normalized access request;

means for, based at least in part on the first normalized access request and an access metadata object associated with the first request type, selecting a first functional component to satisfy at least a portion of the first normalized access request.

16. The system of Claim 15, further comprising:

means for providing at least a portion of the first normalized access request to the first functional component;

means for generating at least a portion of a first response using the first functional component, the first response conforming with the first request type associated with the first access request;and

means for providing at least the portion of the first response to the first requesting entity.

17. The system of any of Claims 15-16, wherein:

the first requesting entity is a first application that is a trusted application;

the first access request includes a request for a security token associated with a second application;

means for generating at least a portion of a response includes means for generating a first token that authorizes the first application to make changes associated with the second application on behalf of a user of the first application.

18. The system of Claim 17, further comprising:

means for receiving a second access request from a second requesting entity, wherein the second access request includes a request for a security token associated with an identity provider;

means for generating a second normalized access request;

means for generating a second token that authorizes access to the third application;

wherein the first token is generated by a token generation engine in response to receiving at least a portion of the first normalized request at the token generation engine;

wherein the second token is generated by the token generation engine in response to receiving at least a portion of the second normalized request at the token generation engine.

19. The system of any of Claims 15-18, wherein the request includes composite state information that identifies a state associated with a first functional component and a state associated with a second functional component.

20. The system of any of Claims 15-19, further comprising:

means for maintaining an access policy repository that stores access policy metadata for determining whether a request for access should be granted;

means for, in response to determining that the first normalized access request does not meet the criteria specified by the access policy metadata, generating a response to the first requesting entity that includes information associated with the access policy metadata.

21. The system of any of Claims 15-20, further comprising:

means for receiving a second access request from a second requesting entity;

means for determining a second request type associated with the second access request;

means for generating a second normalized access request;

means for, based at least in part on the second normalized access request and an access metadata object associated with the second request type, selecting the first functional component to satisfy at least a portion of the second normalized access request;

means for providing at least a portion of the second normalized access request to the first functional component;

means for generating at least a portion of a second response using the first functional component, the second response conforming with the second request type associated with the second access request;and

means for providing at least the portion of the first response to the first requesting entity.

22. A computer-readable non-volatile storage medium storing a plurality of instructions executable by one or more processors, the plurality of instructions when executed by the processor to perform the method according to any one of claims 8-14.

23. A computer-readable non-volatile storage medium comprising code segments which when loaded into one or more computers of a computer system causes the system to carry out the method of any of claims 8-14.

24. A system, comprising:

a storage device that stores access metadata objects, wherein each access metadata object of a plurality of access metadata objects in the storage device describes data corresponding to access services; and

a computing device that

receives a first access request from a first client,

determines a first request type of the first access request,

generates a first normalized access request, and

selects a first functional component to satisfy at least a portion of the first normalized access request based at least on the first normalized access request and an access metadata object stored in the storage device and corresponding to the first request type.

25. A method, comprising:

receiving, by a computing device, a first access request from a first client;

determining, by the computing device, a first request type of the first access request;

generating, by the computing device, a first normalized access request; and

based at least on the first normalized access request and an access metadata object stored in a storage device and corresponding to the first request type, selecting, by the computing device, a first functional component to satisfy at least a portion of the first normalized access request, wherein each access metadata

object of a plurality of access metadata objects in the storage device describes data corresponding to access services.

26. A computer-readable non-volatile storage medium storing instructions that cause a processor to:

receive a first access request from a first client;

determine a first request type of the first access request;

generate a first normalized access request; and

based at least on the first normalized access request and an access metadata object stored in a storage device and corresponding to the first request type, select a first functional component to satisfy at least a portion of the first normalized access request, wherein each access metadata object of a plurality of access metadata objects in the storage device describes data corresponding to access services.

27. A program that causes a processor to:

receive a first access request from a first client;

determine a first request type of the first access request;

generate a first normalized access request; and

based at least on the first normalized access request and an access metadata object stored in a storage device and corresponding to the first request type, select a first functional component to satisfy at least a portion of the first normalized access request, wherein each access metadata object of a plurality of access metadata objects in the storage device describes data corresponding to access services.

FIG. 1

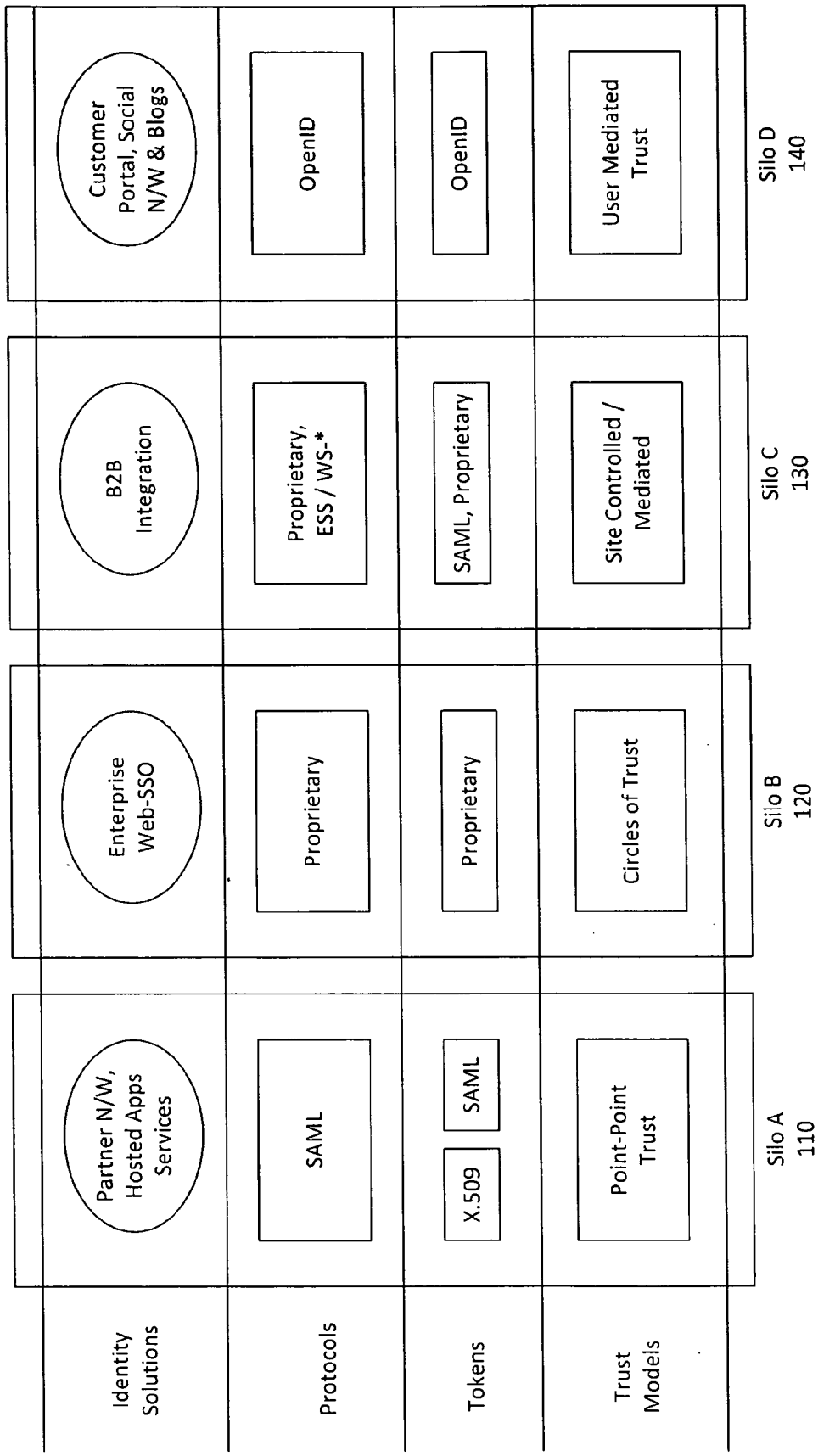

Identity Solutions
Protocols
Tokens
Trust Models
Partner N/W, Hosted Apps Services
SAML
X.509
SAML
Point-Point Trust
Silo A 110
Enterprise Web-SSO
Proprietary
Proprietary
Circles of Trust
Silo B 120
B2B Integration
Proprietary, ESS / WS-*
SAML, Proprietary
Site Controlled / Mediated
Silo C 130
Customer Portal, Social N/W & Blogs
OpenID
OpenID
User Mediated Trust
Silo D 140

FIG. 2

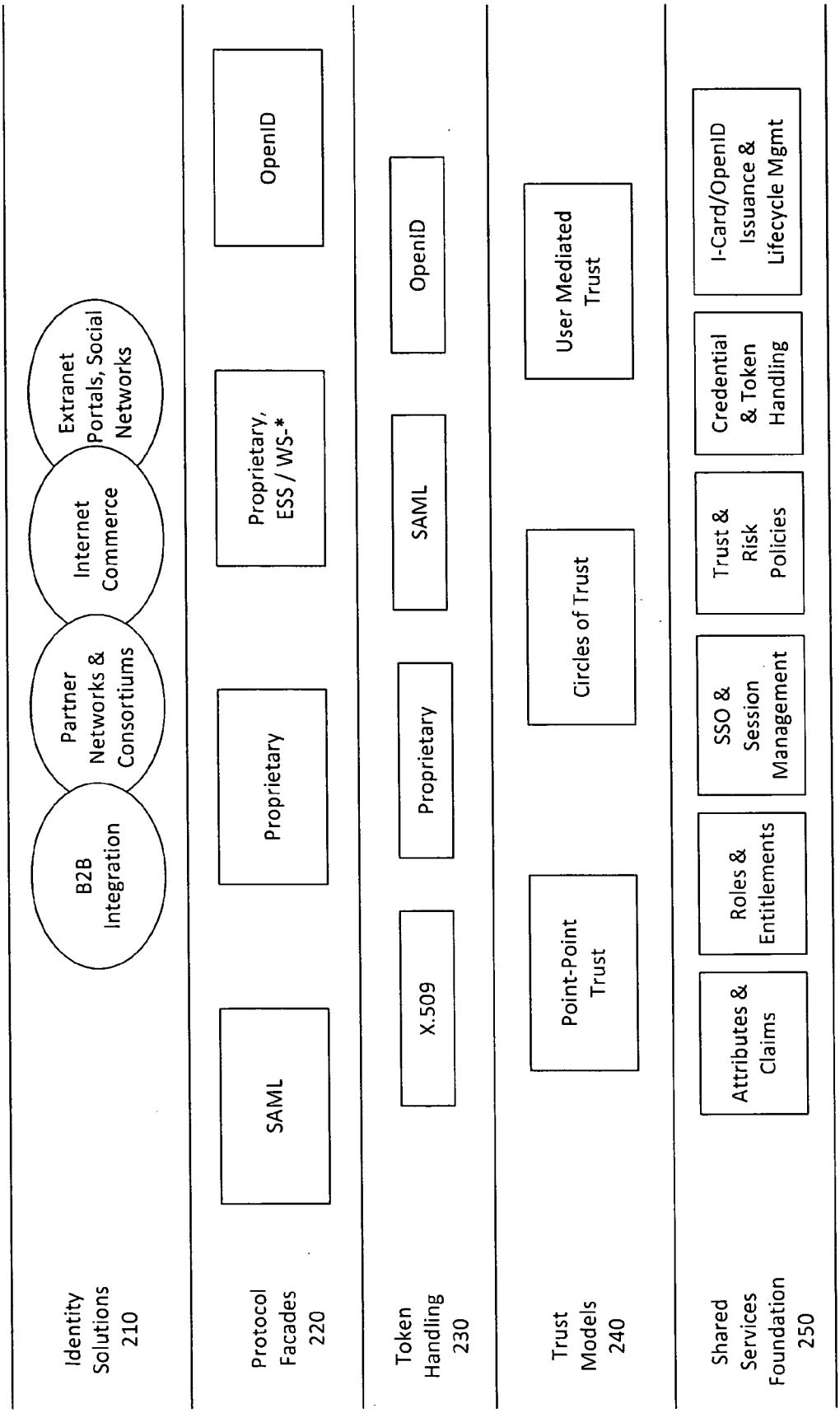

Identity Solutions 210
B2B Integration
Partner Networks & Consortiums
Internet Commerce
Extranet Portals, Social Networks
Protocol Facades 220
SAML
Proprietary
Proprietary, ESS / WS-*
OpenID
Token Handling 230
X.509
Proprietary
SAML
OpenID
Trust Models 240
Point-Point Trust
Circles of Trust
User Mediated Trust
Shared Services Foundation 250
Attributes & Claims
Roles & Entitlements
SSO & Session Management
Trust & Risk Policies
Credential & Token Handling
I-Card/OpenID Issuance & Lifecycle Mgmt

FIG. 3

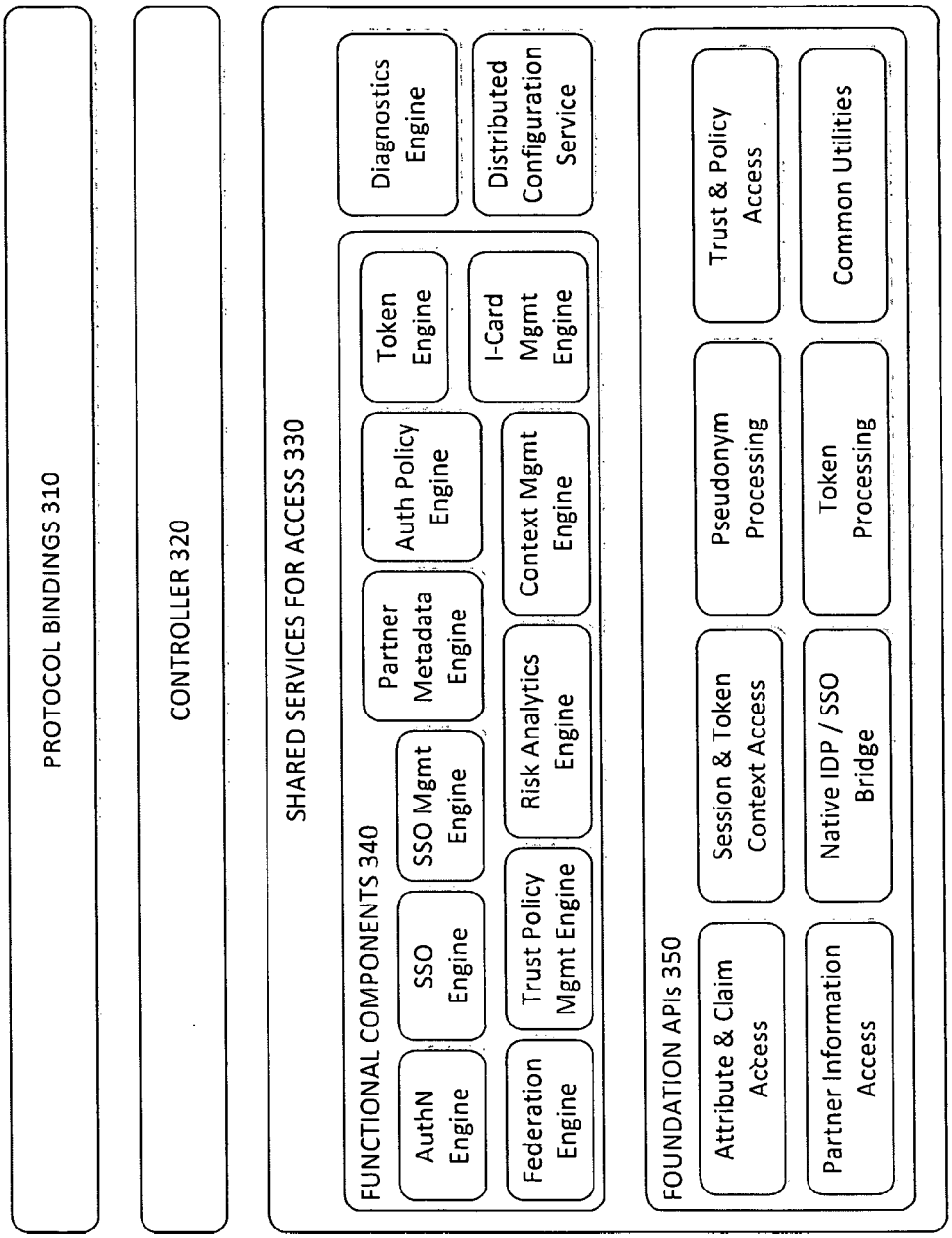

PROTOCOL BINDINGS 310
CONTROLLER 320
SHARED SERVICES FOR ACCESS 330
FUNCTIONAL COMPONENTS 340
AuthN Engine
SSO Engine
SSO Mgmt Engine
Partner Metadata Engine
Auth Policy Engine
Token Engine
Federation Engine
Trust Policy Mgmt Engine
Risk Analytics Engine
Context Mgmt Engine
I-Card Mgmt Engine
Diagnostics Engine
Distributed Configuration Service
FOUNDATION APIs 350
Attribute & Claim Access
Session & Token Context Access
Pseudonym Processing
Trust & Policy Access
Partner Information Access
Native IDP / SSO Bridge
Token Processing
Common Utilities

FIG. 4

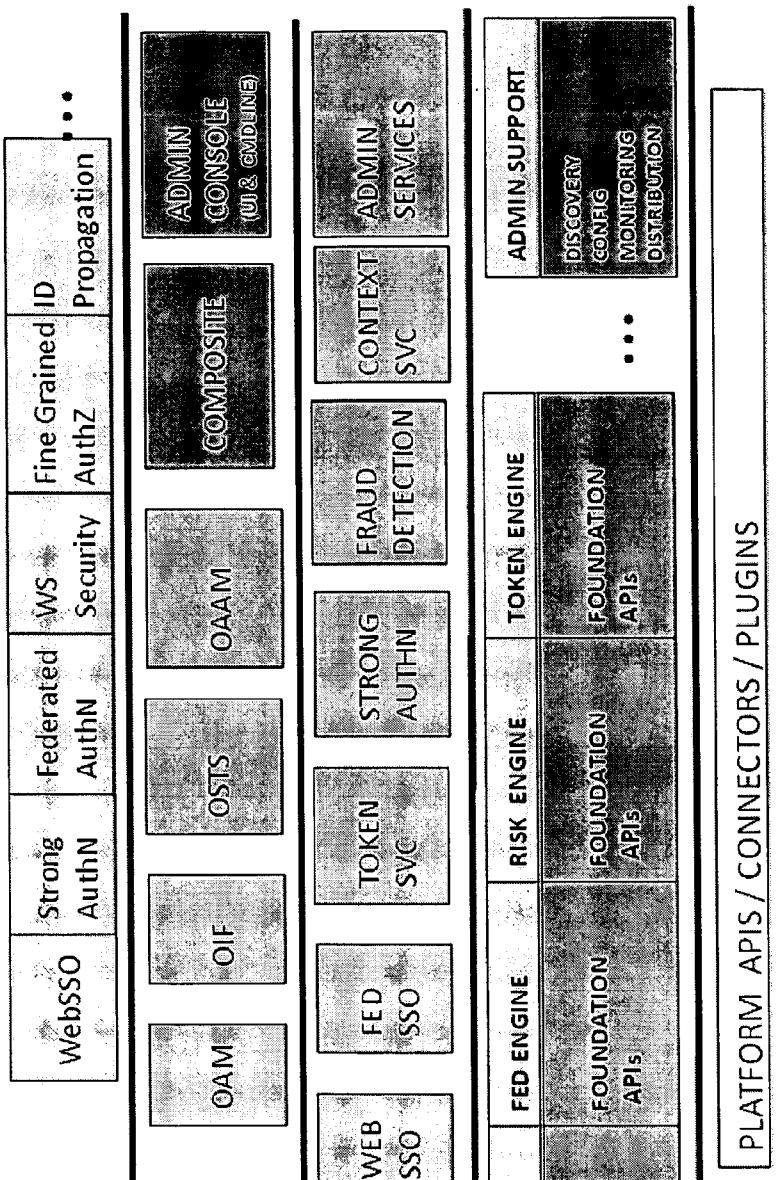

410
SOLUTIONS
WebSSO
Strong AuthN
Federated AuthN
WS Security
Fine Grained AuthZ
ID Propagation
420
PKGD PRODUCTS
OAM
OIF
OSTS
OAAM
COMPOSITE
ADMIN CONSOLE (UI & CMDLINE)
430
STANDALONE ENTITIES
WEB SSO
FED SSO
TOKEN SVC
STRONG AUTHN
FRAUD DETECTION
CONTEXT SVC
ADMIN SERVICES
440
SSA
AUTHN ENGINE
FED ENGINE
RISK ENGINE
TOKEN ENGINE
ADMIN SUPPORT
FOUNDATION APIs
FOUNDATION APIs
FOUNDATION APIs
DISCOVERY CONFIG MONITORING DISTRIBUTION
450
PLATFORM APIS / CONNECTORS / PLUGINS

FIG. 5

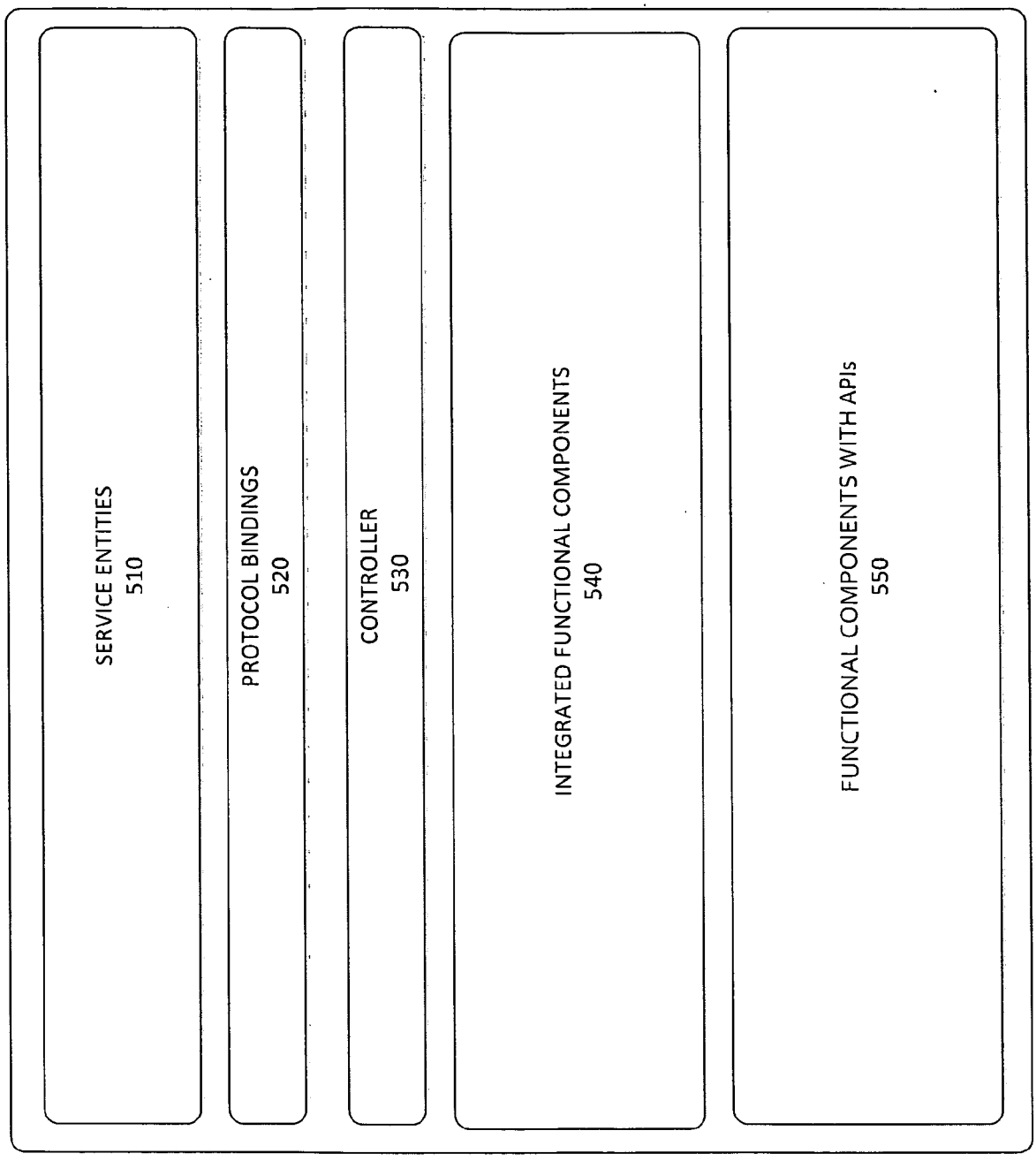

SERVICE ENTITIES
510
PROTOCOL BINDINGS
520
CONTROLLER
530
INTEGRATED FUNCTIONAL COMPONENTS
540
FUNCTIONAL COMPONENTS WITH APIs
550

FIG. 6

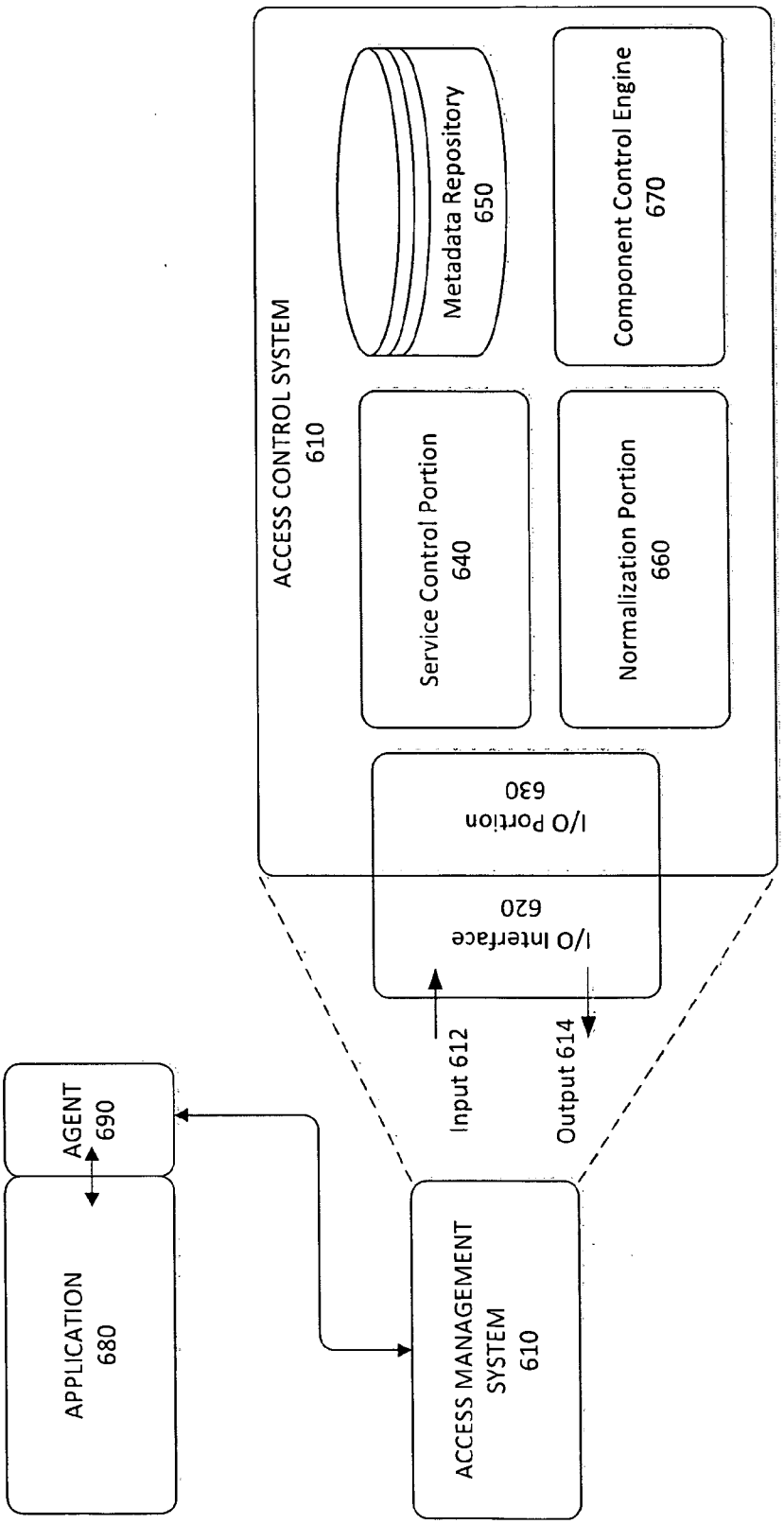

ACCESS CONTROL SYSTEM
610
Metadata Repository
650
Component Control Engine
670
Service Control Portion
640
Normalization Portion
660
I/O Portion
630
I/O Interface
620
Input 612
Output 614
AGENT
690
APPLICATION
680
ACCESS MANAGEMENT
SYSTEM
610

FIG. 7

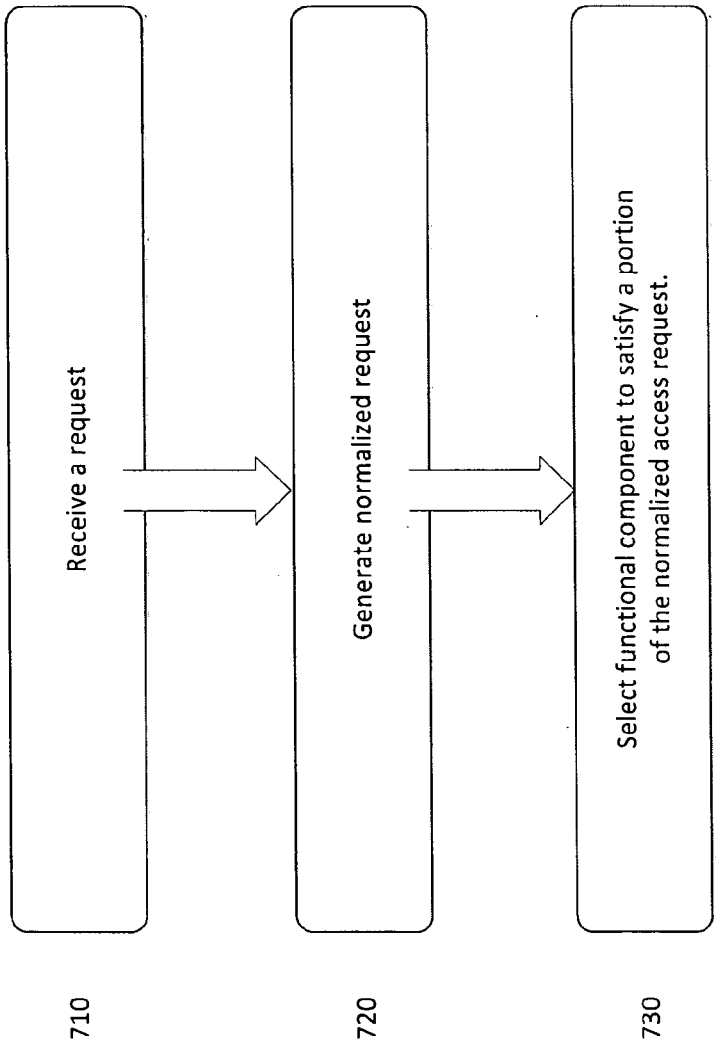

Receive a request
Generate normalized request
Select functional component to satisfy a portion of the normalized access request.
710
720
730

FIG. 8

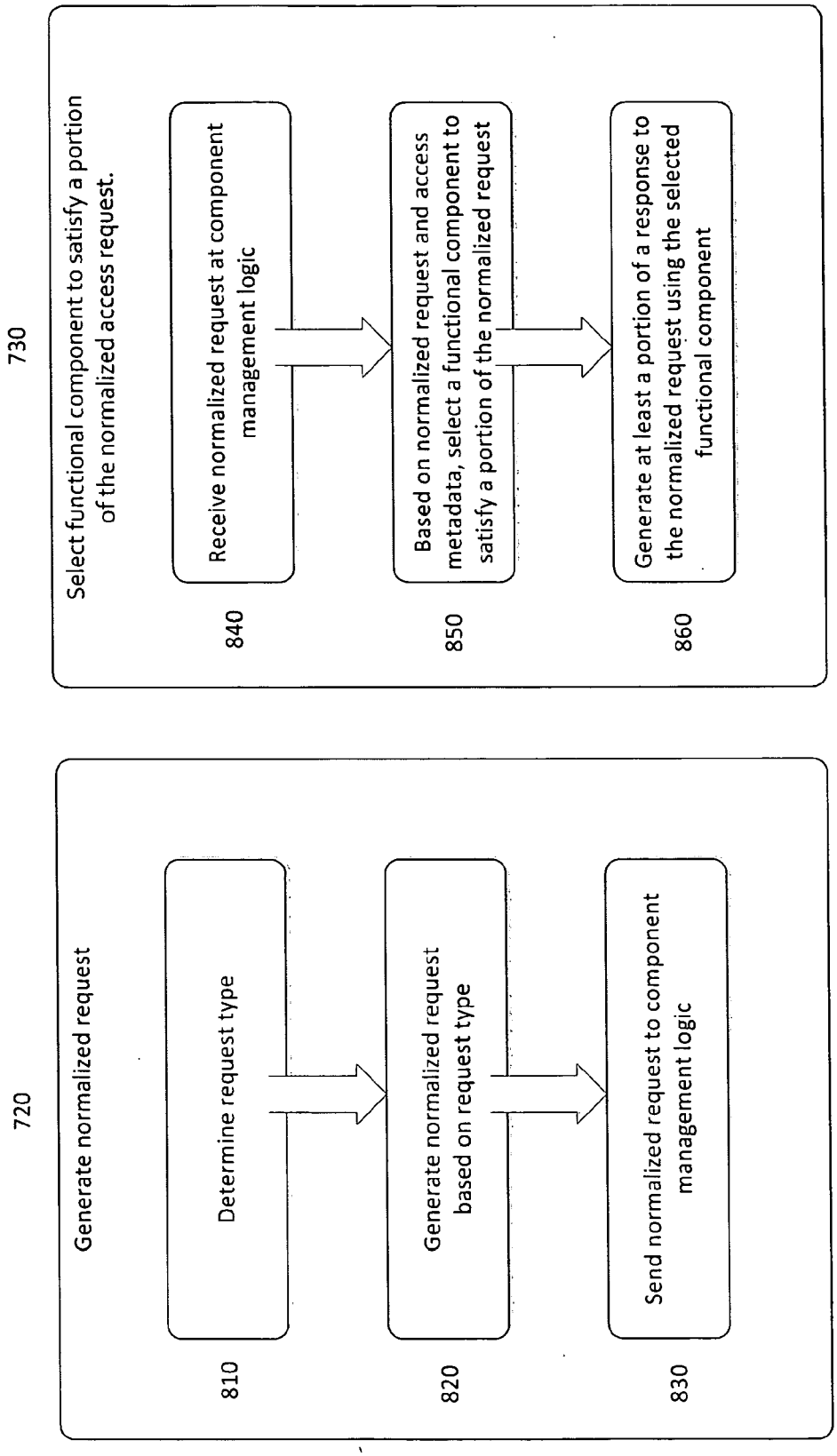

720
Generate normalized request
Determine request type
810
Generate normalized request based on request type
820
Send normalized request to component management logic
830
730
Select functional component to satisfy a portion of the normalized access request.
Receive normalized request at component management logic
840
Based on normalized request and access metadata, select a functional component to satisfy a portion of the normalized request
850
Generate at least a portion of a response to the normalized request using the selected functional component
860

FIG. 9

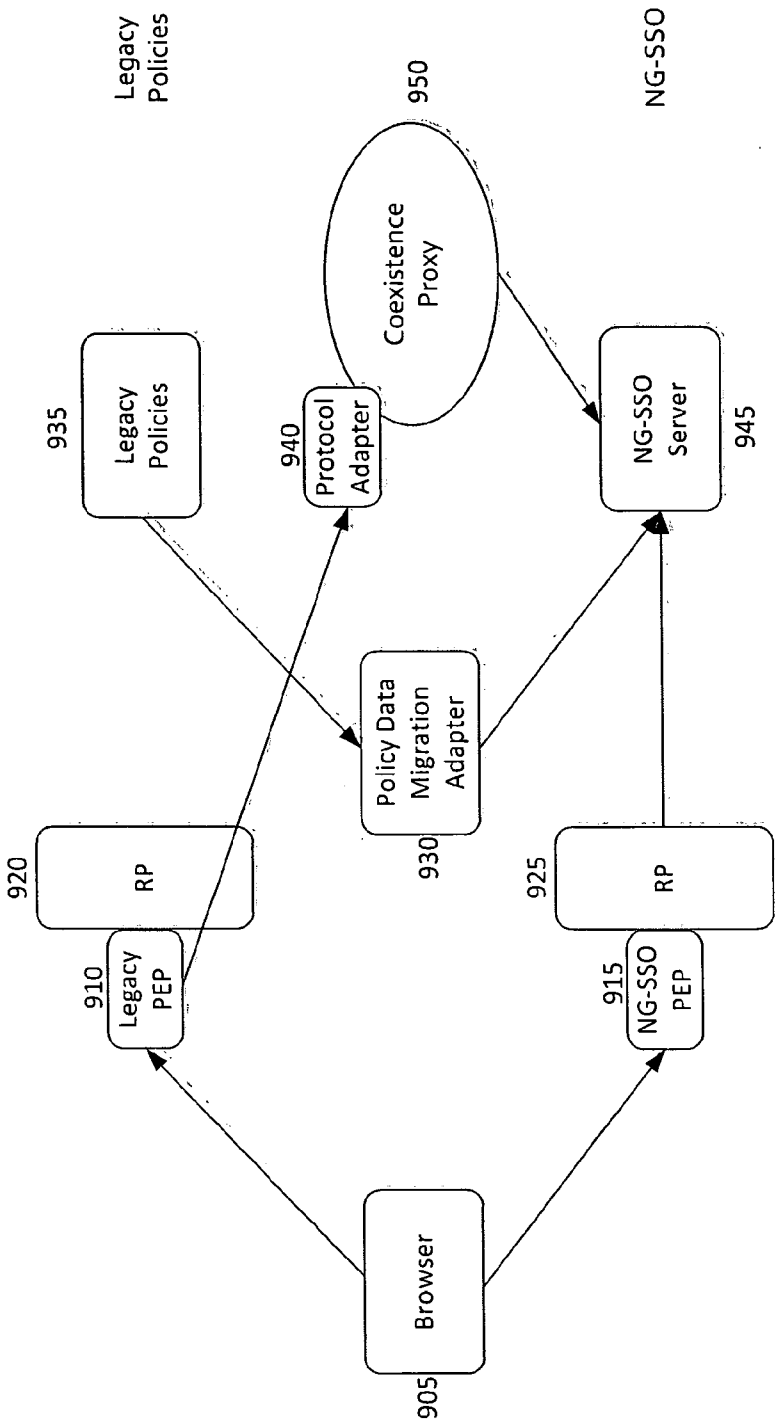

Legacy Policies
950
NG-SSO
Coexistence Proxy
935
Legacy Policies
940
Protocol Adapter
NG-SSO Server
945
Policy Data Migration Adapter
930
920
RP
925
RP
910
Legacy PEP
915
NG-SSO PEP
Browser
905

FIG. 10

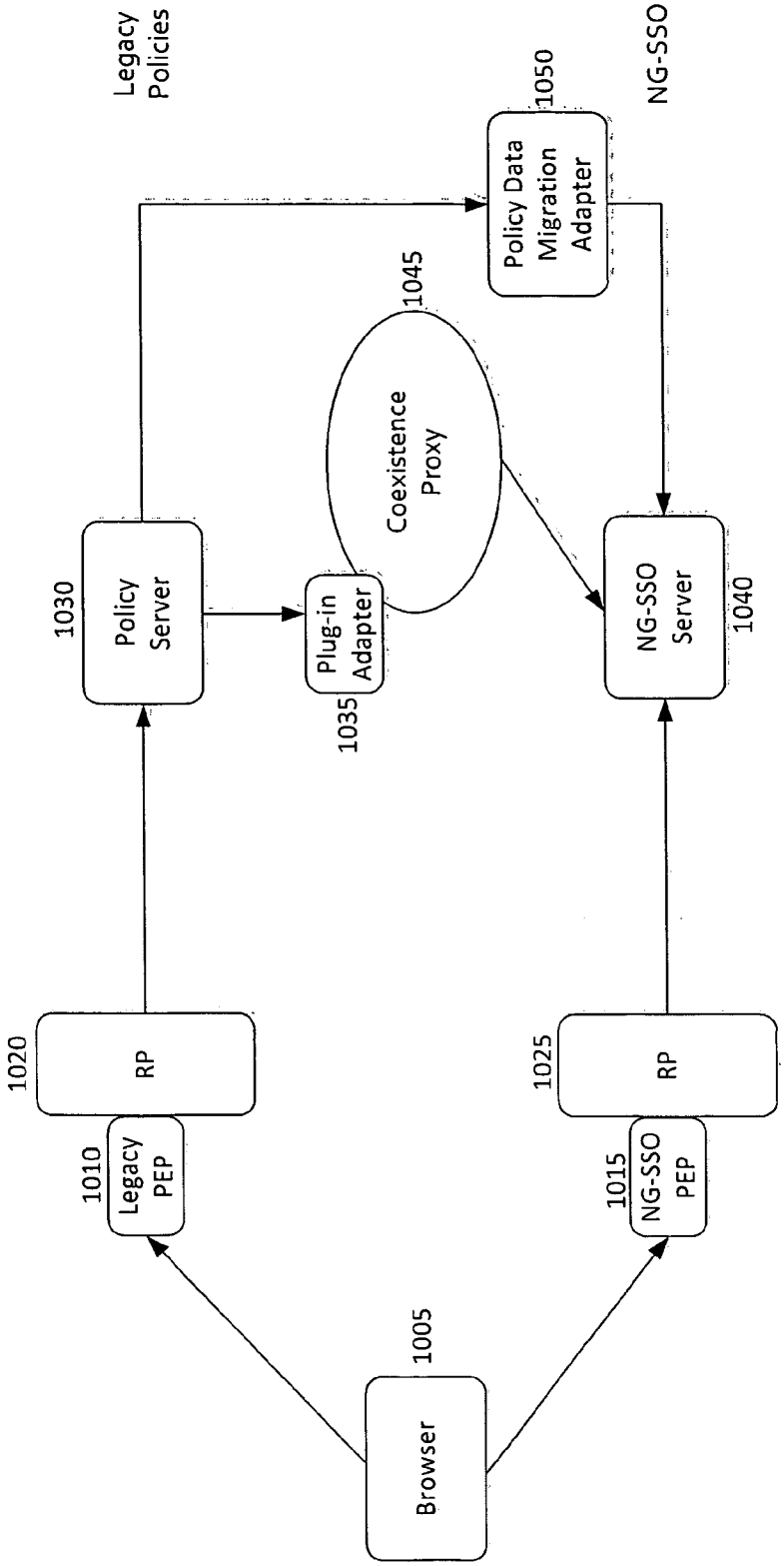

Legacy Policies
NG-SSO
1050
Policy Data Migration Adapter
1045
Coexistence Proxy
1030
Policy Server
Plug-in Adapter
1035
NG-SSO Server
1040
1020
RP
1025
RP
1010
Legacy PEP
1015
NG-SSO PEP
1005
Browser

FIG. 11

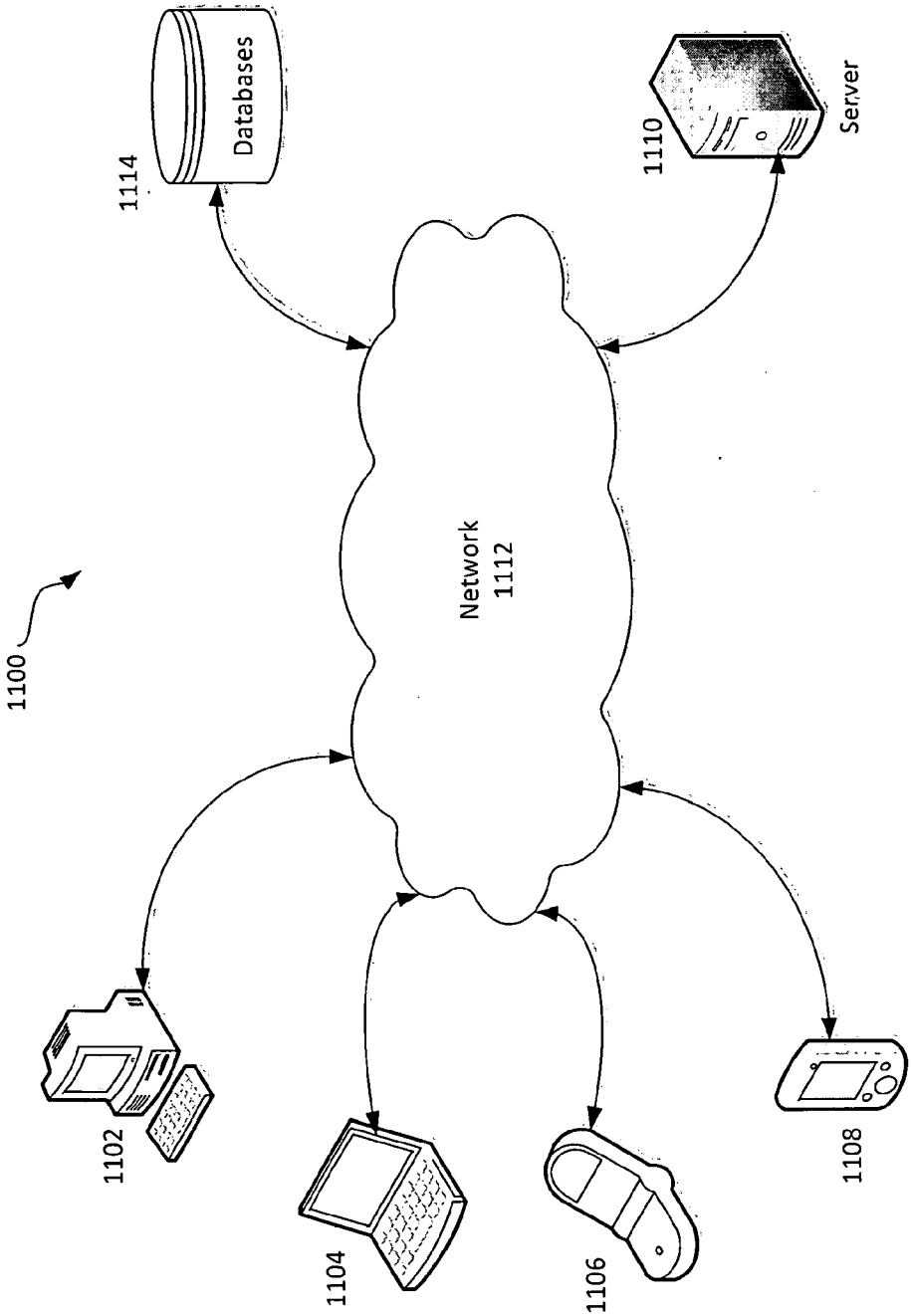
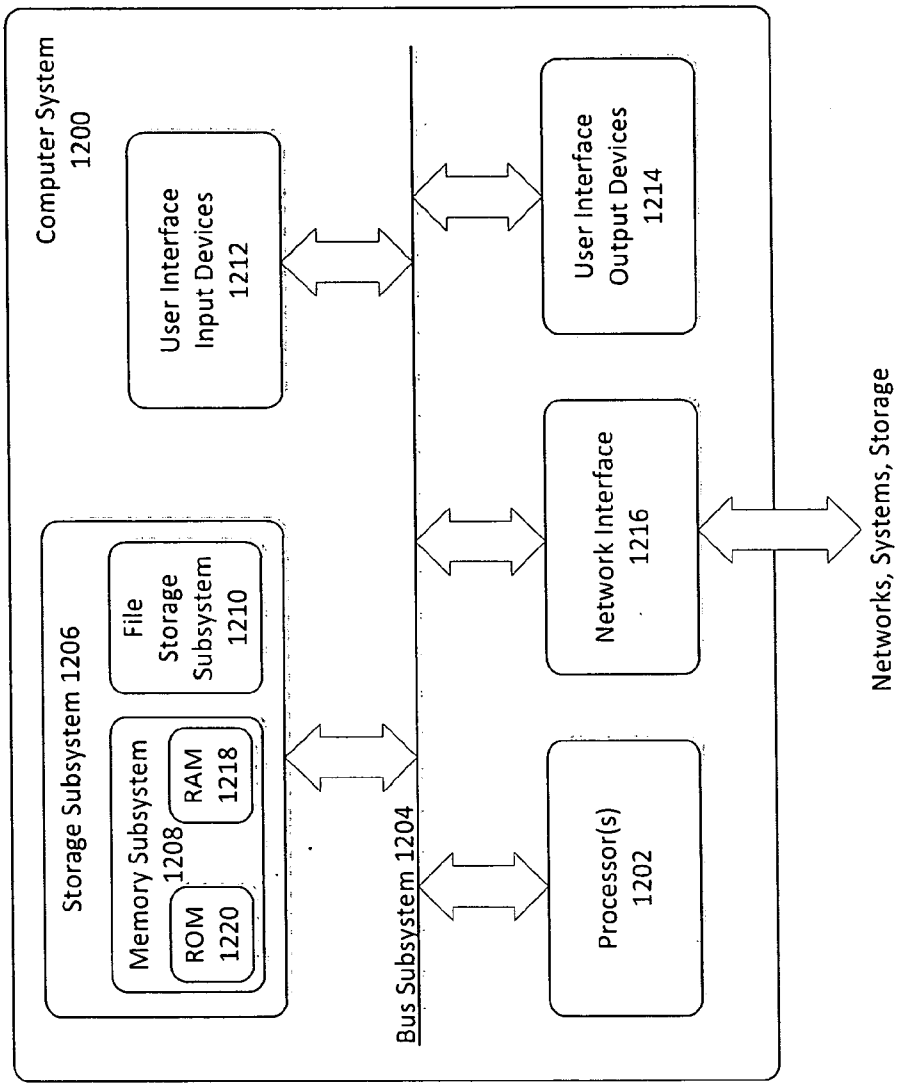

Databases
1114
1110
Server
Network
1112
1100
1102
1104
1106
1108

FIG. 12


Computer System
1200
User Interface
Input Devices
1212
User Interface
Output Devices
1214
Storage Subsystem 1206
File
Storage
Subsystem
1210
Memory Subsystem
1208
RAM
1218
ROM
1220
Network Interface
1216
Networks, Systems, Storage
Bus Subsystem 1204
Processor(s)
1202

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2012/037625

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/24
ADD.

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

B. FIELDS SEARCHED

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

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2007/240231 A1 (HASWAREY BASHIR A [US] ET AL) 11 October 2007 (2007-10-11) paragraph [0001] paragraph [0030] - paragraph [0033]; figures 1, 2 paragraph [0040]; figure 5 paragraph [0045]; figure 9 -----	1-27

☐ 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
17 July 2012	24/07/2012

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

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2012/037625

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007240231 A1	11-10-2007	US 2007240231 A1	11-10-2007
		WO 2007117818 A2	18-10-2007