US 20050138412A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2005/0138412 A1**

Griffin et al. (43) **Pub. Date: Jun. 23, 2005**

(54) **RESOURCE MANAGEMENT WITH POLICIES**

(76) Inventors: **Philip B. Griffin**, Longmont, CO (US); **Manish Devgan**, Broomfield, CO (US); **Alex Toussaint**, Broomfield, CO (US); **Rod McCauley**, Loveland, CO (US)

Correspondence Address:
FLIESLER MEYER, LLP
FOUR EMBARCADERO CENTER
SUITE 400
SAN FRANCISCO, CA 94111 (US)

(21) Appl. No.: **11/052,356**

(22) Filed: **Feb. 7, 2005**

Related U.S. Application Data

(63) Continuation of application No. 10/367,462, filed on Feb. 14, 2003.

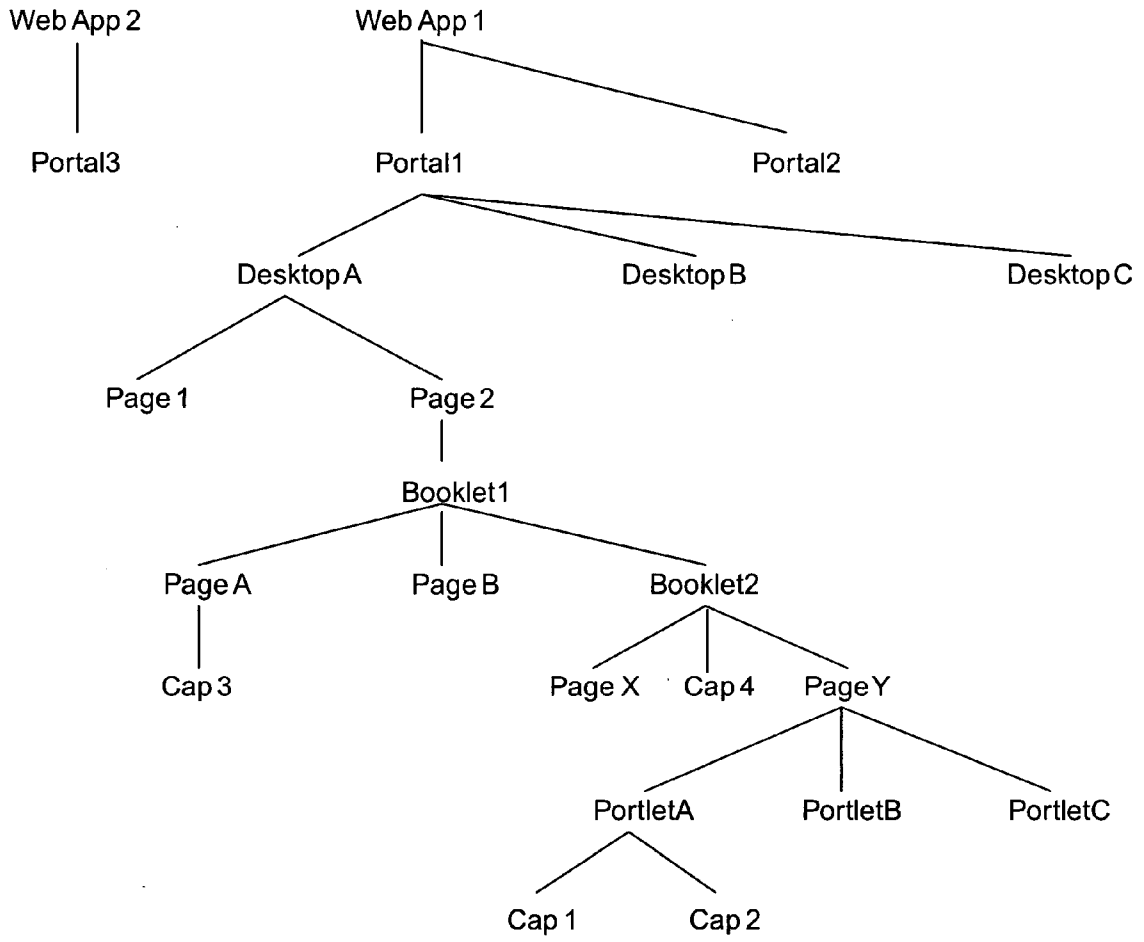
Publication Classification

(51) **Int. Cl.**[7] **H04L 9/00**

(52) **U.S. Cl.** **713/200**

(57) **ABSTRACT**

A system, method and media for controlling access to a resource in a distributed computing environment, comprising: receiving a request for a principal to access the resource; determining a policy that is appropriate for the principal given the resource; determining whether access to the resource is allowed given the policy.



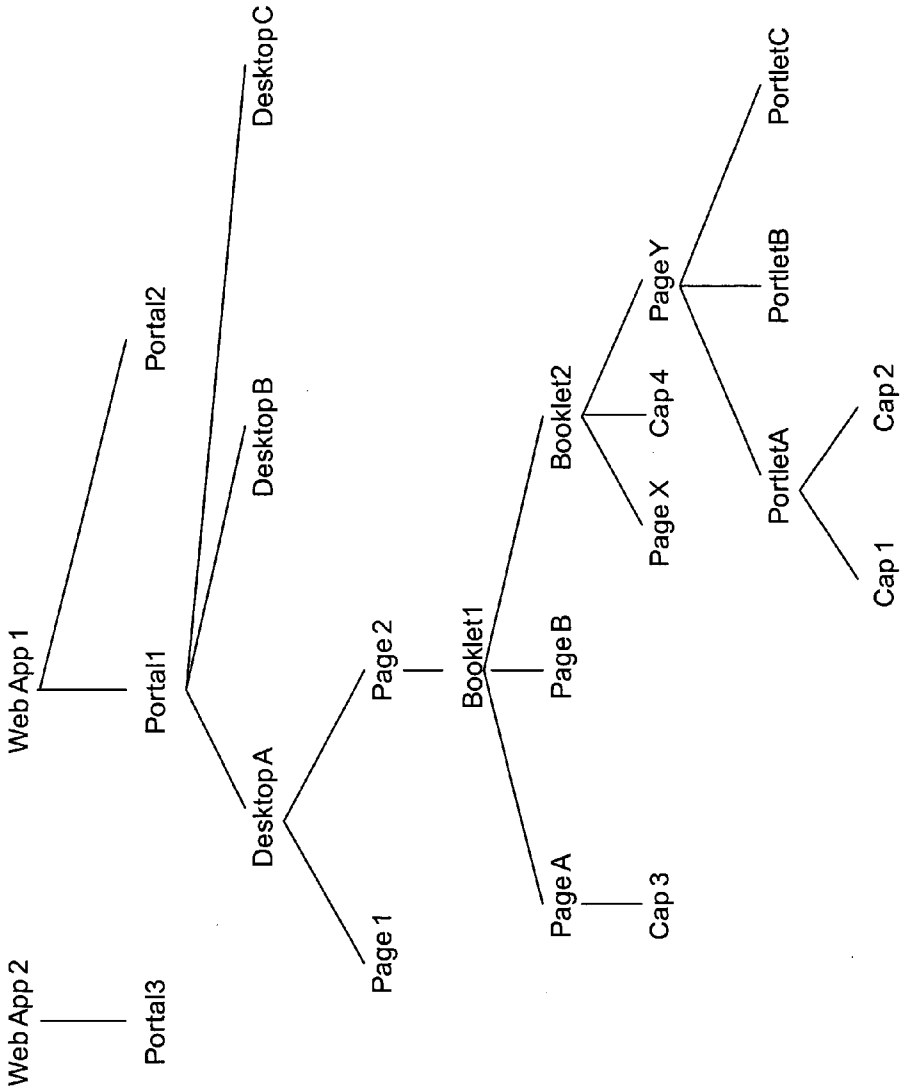

Web App 1
Web App 2
Portal1
Portal2
Portal3
DesktopA
DesktopB
DesktopC
Page 1
Page 2
Booklet1
Page A
Page B
Booklet2
Page X
Page Y
Cap 3
Cap 4
PortletA
PortletB
PortletC
Cap 1
Cap 2

Figure 1

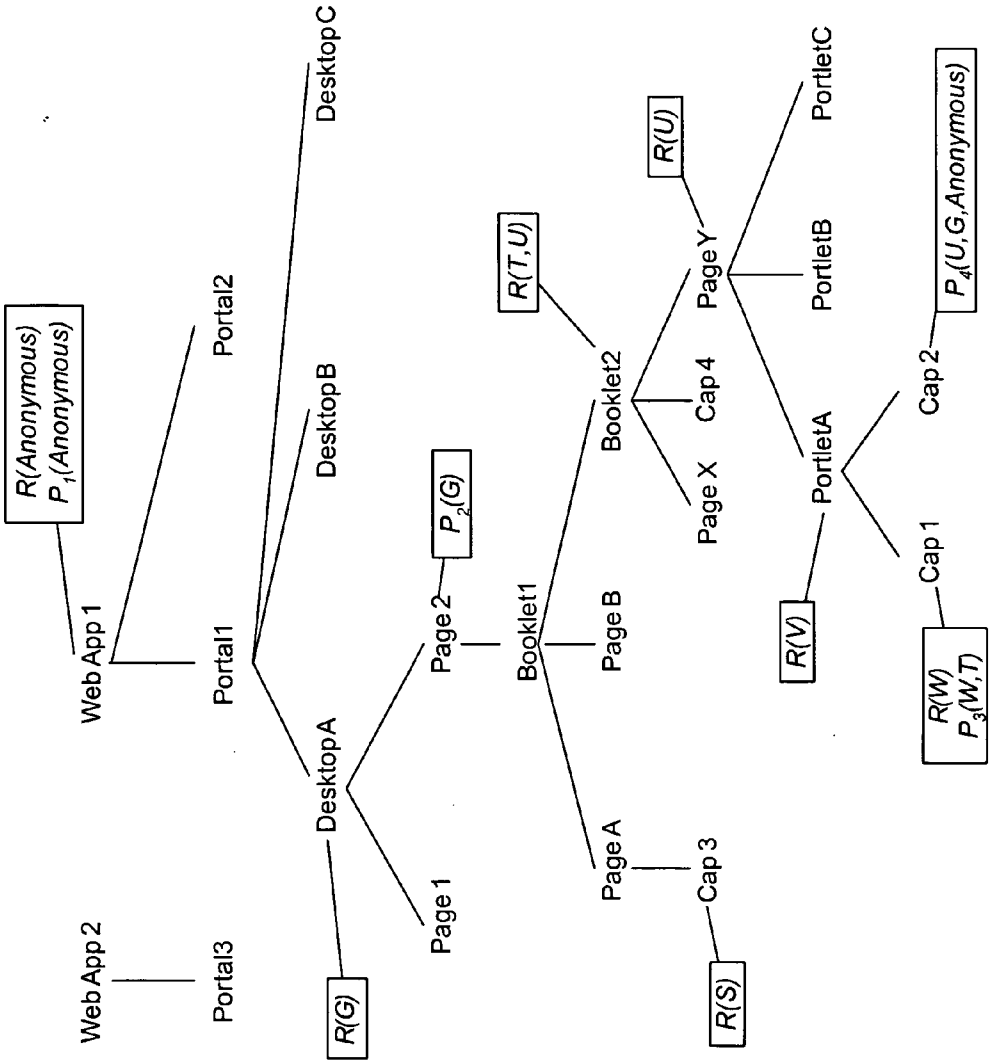

R(Anonymous)
P1(Anonymous)
Web App 1
Portal1
Portal2
DesktopA
DesktopB
DesktopC
Page 1
Page 2
P2(G)
R(G)
Web App 2
Portal3
Booklet1
Booklet2
R(T,U)
Page A
Page B
Page X
Page Y
Cap 3
Cap 4
R(U)
R(S)
PortletA
PortletB
PortletC
R(V)
Cap 1
Cap 2
R(W)
P3(W,T)
P4(U,G,Anonymous)

Figure 2

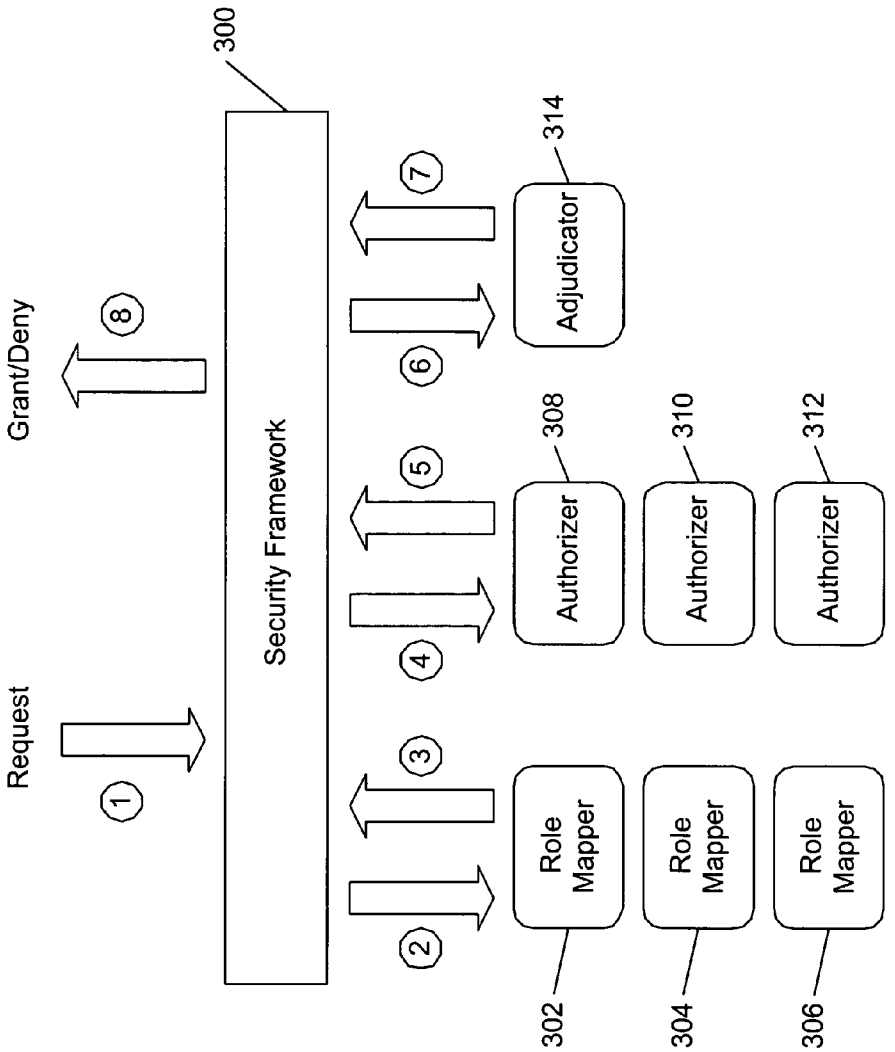

300
Security Framework
Request
Grant/Deny
1
2
3
4
5
6
7
8
Role Mapper
Role Mapper
Role Mapper
Authorizer
Authorizer
Authorizer
Adjudicator
302
304
306
308
310
312
314

Figure 3

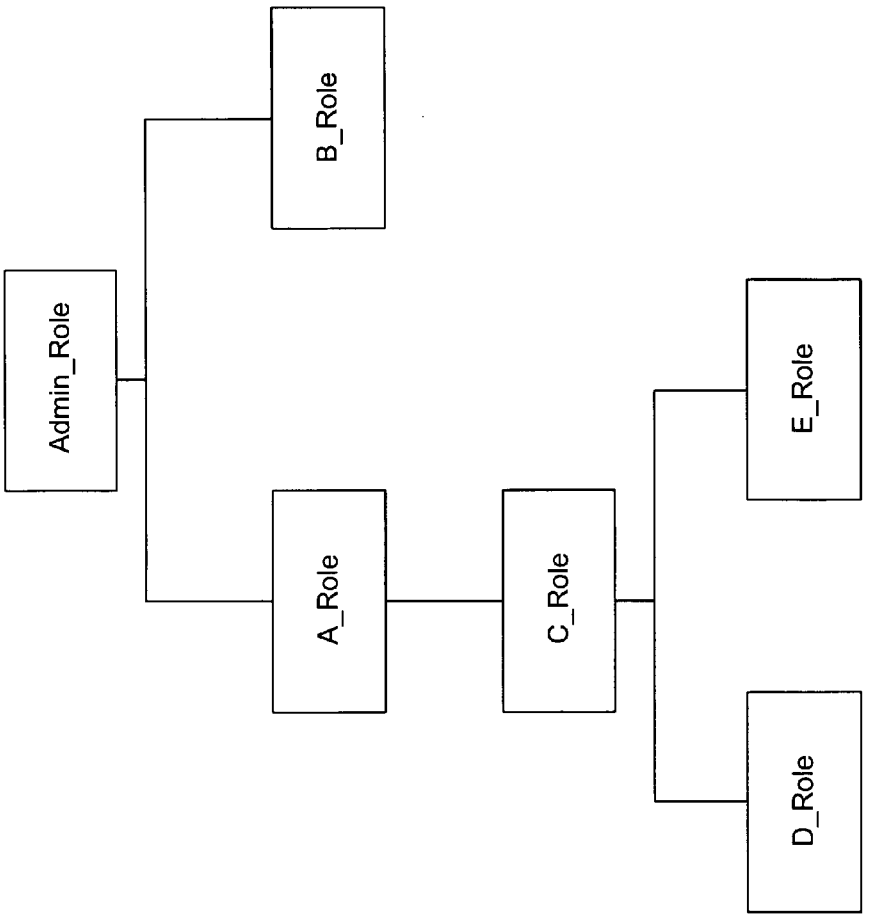


Figure 4

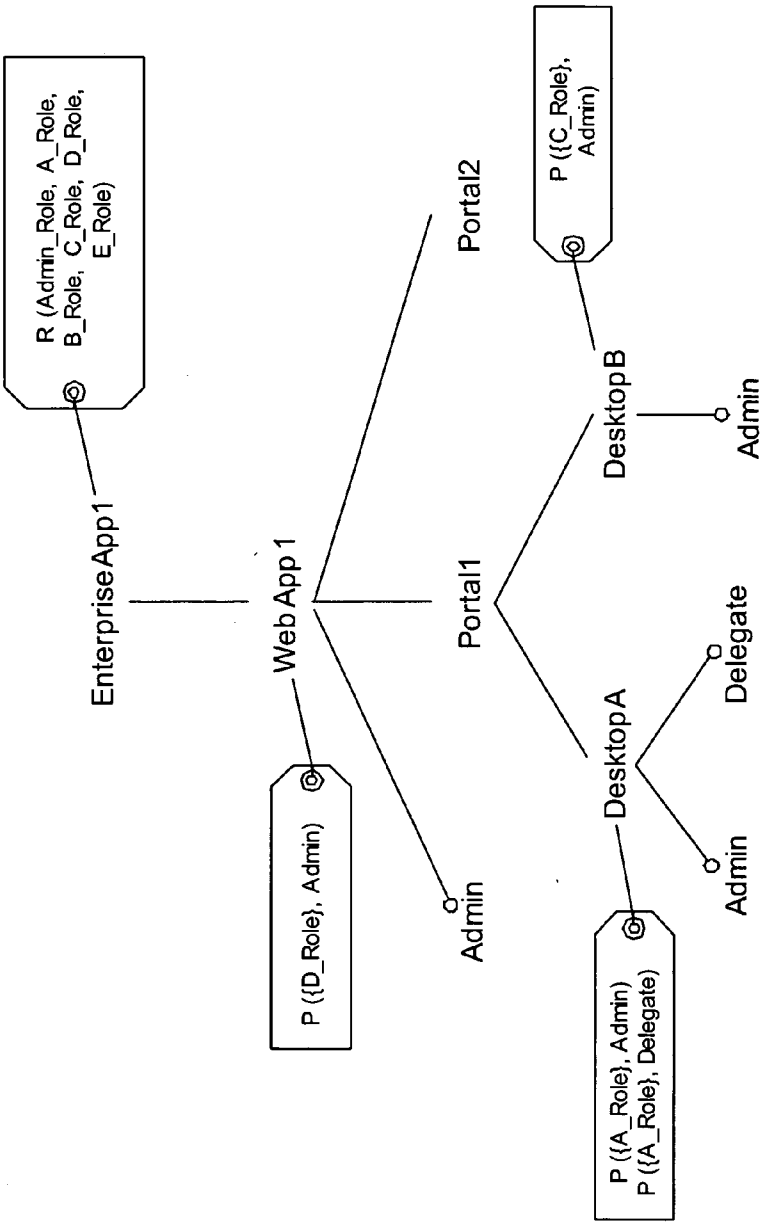

R (Admin_Role, A_Role, B_Role, C_Role, D_Role, E_Role)
EnterpriseApp1
Web App 1
P ({D_Role}, Admin)
Admin
Portal1
Portal2
P ({C_Role}, Admin)
DesktopB
Admin
DesktopA
Delegate
Admin
P ({A_Role}, Admin)
P ({A_Role}, Delegate)

Figure 5

RESOURCE MANAGEMENT WITH POLICIES

CLAIM OF PRIORITY

[0001] This application is a continuation of the following application which is included by reference in its entirety:

[0002] U.S. application Ser. No. 10/367,462 entitled METHOD FOR ROLE AND RESOURCE POLICY MANAGEMENT by Philip B. Griffin et al, filed Feb. 14, 2003 (Attorney Docket No. BEAS-1356US0).

CROSS REFERENCES

[0003] This application is related to the following co-pending applications which are hereby incorporated by reference in their entirety:

[0004] U.S. application Ser. No. 10/367,177 entitled SYSTEM AND METHOD FOR HIERARCHICAL ROLE-BASED ENTITLEMENTS, by Philip B. Griffin et al, filed Feb. 14, 2003 (Attorney Docket No. BEAS-1353US0);

[0005] U.S. application Ser. No. 10/367,190 entitled METHOD FOR DELEGATED ADMINISTRATION by Philip B. Griffin et al, filed Feb. 14, 2003 (Attorney Docket No. BEAS- 1358US0);

[0006] U.S. application Ser. No. 10/366,778 entitled METHOD FOR ROLE AND RESOURCE POLICY MANAGEMENT OPTIMIZATION by Philip B. Griffin et al, filed Feb. 14, 2003 (Attorney Docket No. BEAS-1357US0); and

[0007] U.S. application Ser. No. _____ entitled RESOURCE MANAGEMENT WITH ROLES by Philip B. Griffin et al, filed _____ (Attorney Docket No. BEAS-1356US 1).

COPYRIGHT NOTICE

[0008] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

FIELD OF THE DISCLOSURE

[0009] The present invention disclosure relates to authorization and control of resources in an enterprise application.

BACKGROUND

[0010] Enterprise applications can increase the availability of goods and services to customers inside and outside of an organization. One issue that accompanies deployment of an enterprise application is authorization or access control. Both customers and system administrators need to be privileged to perform certain actions (e.g., modifying a customer account) or to gain access to certain content. Typical authorization systems can be complex and time consuming to implement and maintain, especially if they are tied closely to the business logic in an enterprise application.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] **FIG. 1** is an illustration of a exemplary resource hierarchy in accordance to one embodiment of the invention.

[0012] **FIG. 2** is the exemplary hierarchy of **FIG. 1** further illustrating roles and security policies.

[0013] **FIG. 3** is a diagram of an authorization system in accordance to one embodiment of the invention.

[0014] **FIG. 4** is an illustration of a delegation role hierarchy in accordance to one embodiment of the invention.

[0015] **FIG. 5** is an illustration of exemplary delegation security policies in one embodiment of the invention.

DETAILED DESCRIPTION

[0016] The invention is illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" or "one" embodiment in this disclosure are not necessarily to the same embodiment, and such references mean at least one.

[0017] In one embodiment, an enterprise application includes one or more resources that facilitate the performance of business, scientific or other functions and tasks. In another embodiment, an enterprise application can be a Java™ 2 Enterprise Edition (J2EE) deployment unit that bundles together Web Applications, Enterprise Java™ Beans and Resource Adaptors into a single deployable unit. The Java™ programming language and its run-time libraries and environment are available from Sun Microsystems, Inc., of Santa Clara, Calif. Enterprise applications can include software, firmware and hardware elements. Software, firmware and hardware can be arbitrarily combined or divided into separate logical components. Furthermore, it will be apparent to those skilled in the art that such components, irregardless of how they are combined or divided, can execute on the same computer or can be arbitrarily distributed among different computers connected by one or more networks.

[0018] In one embodiment, a resource can correspond to any person, place or thing, including an object or an entity (e.g., a network, a computer, a computer user, a bank account, an electronic mail message, aspects of a computer operating system such as virtual memory, threads and file storage, etc.), a method or a process (e.g., balancing a checkbook, installing a device driver, allocating virtual memory, deleting a file, etc.), the occurrence or non-occurrence of an event (e.g., an attempt by a user to logon to a computer, a change in state, etc.) and an organization or association of resources (e.g., lists, trees, maps, hierarchies, etc.).

[0019] In one embodiment, resources can be classified into a hierarchical taxonomy (which itself can be a resource). By way of a non-limiting example, in an enterprise application, it may be necessary to refer to a particular resource such as a booklet. In order to reference the booklet, one needs to know which web page it is on, which portal the web page belongs to, which web application (or "web app") owns the web page, and which domain the web app belongs to. Each of these components is considered a resource and can be described as a resource path (e.g., a sequence of components separated by slashes):

[0020] domain/web_app/portal/desktop/page/booklet

[0021] The first resource is domain which lies at the "top" of the resource hierarchy. Working down the hierarchy, the

next component is web_app. The web_app is a "child" or "descendent" of domain and domain is a "parent" of web_app. The domain is superior to web_app and web_app is inferior to domain. Likewise, portal is a child of web_app and a parent of desktop. The page is a child of desktop with booklet as its child. The depth of the resource is the number of components in its path. For example, the depth of booklet is six (assuming that we are counting from 1) and the depth of portal is three. In one embodiment, the depth of a resource can be unlimited. In one embodiment, a resource can have properties or capabilities. By way of a non-limiting example, a booklet resource could have the ability to be customized by an end-user. The capability could be appended to the hierarchy as follows:

[0022] domain/web_app/portal/desktop/page/booklet.customize

[0023] **FIG. 1** is an illustration of an exemplary resource hierarchy in accordance to one embodiment of the invention. By way of a non-limiting example, this hierarchy can represent resources within an enterprise application. Web App **1** and Web App **2** are Web applications. A Web application resource is a part of an enterprise application that is accessible on the World Wide Web. Portal **1** and Portal **2** are portal resources and are children of Web App **1**. Portal **3** is a child of Web App **2**. In one embodiment, Web App **1** and Web App **2** can be children of one or more enterprise applications (not shown) which can be children of one or more domains (not shown). A portal is a point of access to data and applications that provides a unified and potentially personalized view of information and resources. Typically, a portal is implemented as one or more pages on a website (Page **1**, Page **2**, Page A, Page B, Page X, and Page Y). Portal pages can integrate many elements, such as applications, live data feeds, static information and multimedia presentations.

[0024] Desktop A, Desktop B and Desktop C contain one or more views of a portal that have been customized for a particular user or group of users. Pages within each desktop can contain portlets (Portlet A, Portlet B, and Portlet C) and booklets (Booklet **1** and Booklet **2**). A portlet is a self-contained application that renders itself on a portal page. In one embodiment, a booklet is a collection of one or more pages or booklets. Resource Web App **1**/Portal **1**/Desktop A/Page **2**/Booklet **1**/Page A has a capability Cap **3**. Likewise, Web App **1**/Portal **1**/Desktop A/Page **2**/Booklet **1**/Booklet **2** has a capability Cap **4** and Web App **1**/Portal **1**/Desktop A/Page **2**/Booklet **1**/Booklet **2**/Page Y/Portlet A has capabilities Cap **1** and Cap **2**.

[0025] Enterprise applications can control access to their resources and/or capabilities through the use of entitlements. In one embodiment, evaluation of an entitlement consists of determining a security policy by dynamically associating one or more roles with a principal. In one embodiment, a role can be based on rules that take into account information including knowledge about the principal, knowledge about a communication session, the current state of the system, and/or any other relevant information.

[0026] In one embodiment, a user represents a person who uses an enterprise application. A group can be an arbitrary collection of users. In one embodiment, members of a group share common traits such as job title, etc. A process can be a software or firmware computer program or portion thereof of any granularity (e.g., a task, thread, lightweight process, distributed object, Enterprise Java™ Bean, or any other computing operation). Users, groups and processes can be considered subjects. Subjects can be authenticated based on providing adequate proof (e.g., password, social security number, etc.) to an authentication system. Once authenticated, a subject can be considered a principal for purposes of evaluating entitlements. A principal is an identity assigned to a user, group or process as a result of authentication. A principal can also represent an anonymous user, group or process (e.g., a subject that has not been authenticated).

[0027] In one embodiment, a role definition contains one or more expressions that evaluate to true or false when evaluated for a given principal in a given context. In another embodiment, an expression can evaluate to a degree of certainty that access to a resource should be granted. Expressions may be nested within each other and can contain functions, arithmetic or logical operators, etc. In one embodiment, expressions are combined (e.g., with Boolean operators such as "and", "or", and "not") to form a Boolean expression that evaluates to true or false. If a role evaluates to true, then the principal in question is considered to satisfy the role.

[0028] Role expressions can be dynamically evaluated against a principal attempting to access a resource in a given context. A context can contain any information relevant to making a determination of whether a principal belongs in a role. By way of a non-limiting example, a context can include any of a principal's attributes (e.g., name, age, address, etc.) and/or information about a communication session. In another embodiment, a context can include information from a hypertext transfer protocol ("HTTP") or hypertext transfer protocol (secure) (HTTPS) request. This information can pertain to character encoding, remote user, authorization scheme, content length, server port, context path, request URI, request method, scheme, servlet path, content type, remote host, request protocol, locale, server name, remote address, query string, path information, etc. It will be apparent to those skilled in the art that a context can include any information which is relevant to evaluating an expression.

[0029] In one embodiment, expressions can include predicates. The invention disclosed herein is not limited to the present predicates discussed. A user predicate evaluates to true if the principal in question is the principal supplied as an argument to the predicate. The group predicate evaluates to true if the principal in question is a member of the specified group.

TABLE 1

Exemplary Roles

Role	Expression
Anonymous	Satisfied by all principals
BankManager	(User = Donna)
CustomerService	(User = Michael or Peter) or (Group = BankTellers)
LoanOfficer	(Group = Associate) and (Group = TrainingLevel2) and not (User = Bob)
BankManager	(User = Donna) and ((10/14/02 <= currentDate <= 10/25/02) or (11/14/02 <= currentDate <= 11/25/02))

TABLE 1-continued

Exemplary Roles

Role	Expression
Software	(Segment = JavaDeveloper)
SysAdmin	((User = Donna) and ((10/14/02 <= currentDate <= 10/25/02) or (11/14/02 <= currentDate <= 11/25/02))) or (Segment = SystemAdministrator)

[0030] Table 1 illustrates seven exemplary roles and their accompanying expressions. In one embodiment, the role "Anonymous" is a special role that is always satisfied. In another embodiment, the role of "Anonymous" is satisfied by an unauthenticated principal. The role of "BankManager" is met by a principal that is authenticated as user "Donna". The role of "CustomerService" is fulfilled by a principal authenticated as "Michael" or "Peter", or belonging to group "BankTellers". The "LoanOfficer" role is met by a principal that is a member of both the "Associate" group and the "TrainingLevel2" group, but is not "Bob". Roles can also be dynamic. By way of a non-limiting example, a role can be date and/or time dependent. In one embodiment, a time period can be specified using the currentDate predicate. The role of "BankManager" can be fulfilled by "Donna", but only between Oct. 14, 2002-Oct. 25, 2002 or Nov. 14, 2002-Nov. 25, 2002. It will be apparent to those skilled in the art that many such date or time predicates are possible (e.g., a predicate that is based on a date and a time, or one that is based on time only, etc.).

[0031] In addition to the predicates discussed above, a segment predicate (hereafter referred to as a "segment") can also be included in a role definition. A segment evaluates to true if the principal in question satisfies the segment's criteria. A segment can be defined in terms of one or more expressions or conditions which can be nested and include logical operators, mathematical operations, method calls, calls to external systems, function calls, etc. In another embodiment, a segment can be specified in plain language. By way of a non-limiting example:

[0032] When all of these conditions apply, the principal is a

[0033] JavaDeveloper:

[0034] Developer is equal to True

[0035] Skill level is equal to 'High'

[0036] Preferred language is equal to 'Java'

[0037] In this example, the segment being described is "ExperiencedJavaDeveloper". The condition "Developer is equal to True" will evaluate to true when information contained in or referenced through a context indicates that the principal in question is a user in the software development department of an organization. Likewise, the other conditions ("Skill level is equal to 'High"', "Preferred language is equal to 'Java"') could similarly be evaluated using information from or referenced through a context. In another embodiment, a condition can pertain to information about a communication session. It will be apparent to those skilled in the art that a condition can be based on any information, whether the information is connected with a particular principal or not. If the segment as a whole evaluates to true, the principal is said to have satisfied the segment. In Table **1**, by way of a non-limiting example, the role of "Software" is met by a principal that satisfies the "JavaDeveloper" segment. By way of a further non-limiting example:

[0038] When all of these conditions apply, the principal is a SystemAdministrator:

[0039] TimeofDay is between 12:00 am and 7:00 am

[0040] SystemLoad is 'Low'

[0041] AdminSkillLevel is at least 5

[0042] In this example, two conditions ("TimeofDay is between 12:00 am and 7:00 am" and "SystemLoad is 'Low"') are based on information unrelated to a particular principal. The segment evaluates to true for the principal in question if it is the middle of the night, the system is not busy, and the principal has level 5 administration skills. In Table 1, by way of a non-limiting example, the role of "SysAdmin" is met by "Donna", but only between Oct. 14, 2002-Oct. 25, 2002 or Nov. 14, 2002-Nov. 25, 2002, or by a principal that satisfies the "SystemAdministrator" segment.

[0043] In one embodiment, a segment can be persisted in Extensible Markup Language (XML). XML is a platform independent language for representing structured documents. Retrieving information stored in an XML document can be time consuming since the text comprising the XML document must be parsed. To save time, in another embodiment once a XML document representing a segment has been parsed, the information extracted therefrom can be cached to avoid the need to parse the file again.

[0044] **FIG. 2** is the exemplary hierarchy of **FIG. 1** further illustrating roles and security policies. Roles are designated by the letter 'R' followed by a parenthetical list of one or more roles. Likewise, policies are designated by the letter 'P' followed by a parenthetical list including a set of roles and an optional capability to which the policy applies. If no capability is present, the policy applies to the resource as a whole. In one embodiment, roles can be considered global in scope or can be associated with a particular resource. A global role is considered within the scope of any resource. In one embodiment, a role associated with a resource is within the scope of that resource. In another embodiment, the role is within the scope of the resource and all of its descendents. In yet another embodiment, the role is within the scope of the resource and all of its descendents unless a role with the same name is associated with a descendent. In this way, a "more local" role occludes a "less local" role of the name.

[0045] In **FIG. 2**, the role Anonymous is associated with the resource Web App **1**. In one embodiment, Anonymous is within the scope of Web App **1** and all resources beneath it in the hierarchy. Role G is associated with resource Desktop A and as such, is within the scope of Desktop A and its descendents. Role S is associated with resource Page A. Since Page A has no children (i.e., the attribute Cap **3** does not count as a child), the scope of role S is limited to Page A. Resource Booklet **2** is associated with roles T and U. In one embodiment, role T is within the scope of Booklet **2** and all of its descendents but the same does not hold true for role U. Since a descendent of Booklet **2** (i.e., Page Y) is asso-

ciated with another role by the same name, the role U associated with Booklet **2** is only within the scope of Booklet **2** and Page X. In one embodiment, the role U associated with Page Y however is within the scope of all of the descendents of Page Y (i.e., Portlet A, Portlet B, and Portlet C). Roles V and W are within the scope of Portlet A.

[**0046**] In one embodiment, a security policy (hereinafter referred to as a "policy") is an association between a resource, a set of roles, and an optional capability. Generally speaking, a policy grants access to the resource for all principals for which the set of roles evaluates to true. In one embodiment, a policy is satisfied if any of its roles evaluate to true for a given principal. In another embodiment, a policy is satisfied if all of its roles evaluate to true for a given principal. In another embodiment, a security policy integrity system can prevent removing or deleting roles that have policies which depend on them. Although one of skill in the art will recognize that there are many ways to implement such a system, one approach would be to keep track of the number of policies that depend on a particular role by using a reference count. Only when the reference count is equal to zero will the particular role be eligible for removal.

[**0047**] In yet a further embodiment, a policy's set of roles can be an expression including Boolean operators, set operators and roles for operands. A policy can be expressed as the tuple <resource, roles, [capability]>, wherein resource specifies the name of a resource and roles specifies a set of roles, and capability is an optional capability. While a policy is predicated on one or more roles, roles are predicated on users and groups. Therefore, one of skill in the art will appreciate that policies are in essence predicated on users, groups and/or segments. By way of illustration, there are four policies illustrated in **FIG. 2**:

[**0048**] P1=<Web App **1**, {Anonymous}>

[**0049**] P2=<Web App **1**/Portal **1**/Desktop A/Page **2**, {G}>

[**0050**] P3=<Web App **1**/ . . . /Page Y/Portlet A, {W, T}, Cap **1**>

[**0051**] P4=<web App **1**/ . . . /Page Y/Portlet A, {U, G, Anonymous}, Cap **2**>

[**0052**] By way of a non-limiting illustration, assume a principal p attempts to access resource Cap **1**. In order to do so, the security policy P3 on Cap **1** requires that p satisfy either role W or T. In one embodiment, all roles within the scope of Cap **1** (i.e., Anonymous, G, T, U, U, V, and W) are determined for p. If any of the roles that p satisfies match W or T, P3 is likewise satisfied and access to Cap **1** is granted for p.

[**0053**] By way of a further non-limiting illustration, assume principal p attempts to access capability Cap **2** for resource Portlet A. In order to do so, the security policy P4 on Cap **2** requires that p satisfy one of the roles U, G or Anonymous. In one embodiment, all roles within the scope of Portlet A (i.e., Anonymous, G, T, U, V and W) are determined for p. Note that in one embodiment, the role U associated with resource Booklet **2** is not in the scope of Portal A. Instead, the role having the same name but associated with the more "local" resource Page Y occludes it. Thus, if any of the roles that p satisfies match U, G or Anonymous, P4 is satisfied and access to Cap **2** is granted for p. However, since in one embodiment every principal satisfies the role Anonymous, P4 will always be satisfied.

[**0054**] By way of a further non-limiting example, assume p attempts to access capability Cap **4** associated with resource Booklet **2**. This resource has no policy. In one embodiment, access will be denied. In another embodiment, access will be granted. In yet a further embodiment, access will be granted if p satisfies a policy in a parent resource of Booklet **2**. Table 2 is a non-limiting illustration of a parent policy search using the resource hierarchy of **FIG. 2**. It is important to note, however, that the particular search order or the method of searching is irrelevant for purposes of this disclosure. In yet another embodiment, a resource without an explicit policy can include information regarding its parent policy and thus circumvent the need for a search.

TABLE 2

Exemplary Policy Search

Search Step	current Resource	Capability	Policy Found?
1	Web App 1/Portal 1/Desktop A/Page 2/Booklet 1/Booklet 2	Cap 4	No
2	Web App 1/Portal 1/Desktop A/Page 2/Booklet 1/Booklet 2		No
3	Web App 1/Portal 1/Desktop A/Page 2/Booklet 1	Cap 4	No
4	Web App 1/Portal 1/Desktop A/Page 2/Booklet 1		No
5	Web App 1/Portal 1/Desktop A/Page 2	Cap 4	No
6	Web App 1/Portal 1/Desktop A/Page 2		Yes

[**0055**] In one embodiment, the search for a policy proceeds as follows. The starting point for the search is the resource that owns the capability (i.e., Booklet **2**) to which the principal is attempting to access (i.e., Cap **4**). This is the current resource. If no policy exists at the current resource for the specific capability, in Step **2** we determine whether or not there is a policy merely on the resource itself. If no policy is found, in Step **3** the current resource is set equal to its parent (i.e., Booklet **1**). If the current resource has no policy for Cap **4**, we determine whether or not there is a policy on Booklet **1** itself. If no policy is found, in Step **5** the current resource is set equal to its parent (i.e., Page **2**). If no policy is found for Cap **4** at the current resource, we determine in Step **6** whether or not there is a policy on Page **2** itself. Since this is the case, the search stops at Step **6**. Web App **1**/Portal **1**/Desktop A/Page **2** has policy P2. Therefore if p satisfies role G, access to Cap **4** is granted for p.

[**0056**] In another embodiment, capabilities are associated with particular resource types. For example, booklets may have a type of capability (e.g., Cap **4**) that is not compatible with or available for other resource types (e.g., pages or desktops). Therefore, when searching for a policy as in Table 2, if a capability is not compatible for the current resource, that resource can be omitted from the search. In yet a further embodiment, if a policy is not found for a given resource type, a global library could be consulted to determine if there are any applicable global policies.

[**0057**] In another embodiment, roles and policies can reside in their own hierarchies, apart from the primary resource hierarchy. For applications that do not need to associate roles and/or policies with resources in the primary

hierarchy, such an approach can allow for a shallow role and/or policy tree, perhaps only with a single level. Searching smaller hierarchies can potentially reduce the time it takes to find all roles within scope and locate a policy.

[0058] **FIG. 3** is a diagram of an authorization system in accordance to one embodiment of the invention. Although this diagram depicts objects as functionally separate, such depiction is merely for illustrative purposes. It will be apparent to those skilled in the art that the objects portrayed in **FIG. 3** can be arbitrarily combined or divided into separate software, firmware or hardware components. Furthermore, it will also be apparent to those skilled in the art that such components, irregardless of how they are combined or divided, can execute on the same computer or can be arbitrarily distributed among different computers connected by one or more networks.

[0059] In one embodiment, security framework **300** is a modular security architecture having a published interface that allows for plug-in components. By way of a non-limiting example, a framework can be a library, a set of interfaces, distributed objects, or any other means for software, firmware and/or hardware components to intercommunicate. Connected to the framework are one or more role mapper components **(302-306)**. A role mapper maps (e.g., determines which roles are appropriate) a principal to one or more roles based on a resource hierarchy and a context. Each role mapper can implement its own specialized algorithms in this regard and use information and resources beyond that which is provided by the framework. Also connected to the framework are one or more authorizers **(308-310)**. An authorizer is responsible for determining if access to a resource can be granted based on whether a principal satisfies a resource policy. Each authorizer can implement its own specialized algorithms in this regard and use information and resources beyond that which is provided by the framework. Finally, adjudicator **314** resolves any difference in outcome between authorization modules and returns a final result (e.g., "grant", "deny" or "abstain"). In one embodiment, the adjudicator can take the logical "or" of the final results such that if any result is a "grant", the outcome of adjudication is "grant". In another embodiment, the adjudicator can take the logical "and" of the final results such that if any result is a "deny", the outcome of adjudication is "deny". In yet a further embodiment, the adjudicator can use a weighted average or other statistical means to determine the final outcome.

[0060] A process can interact with the framework in a number of ways which will be apparent to those skilled in the art. In one embodiment, a calling process provides a resource access request ① to the framework **300**. This request can include information about the principal, the resource to which access is requested, and any context information. In another embodiment, the request can contain references to this information. This information is then provided to one or more role mappers ② by the framework. Each role mapper determines which roles are appropriate for the principal based on their own criteria. In another embodiment, each role mapper can implement a cache to speed up searching for roles. Rather than traversing a resource tree to find all roles within scope, each role mapper can cache roles that were previously retrieved from a resource tree based on a key comprising the resource to which access is requested and the principal. After the initial retrieval from a resource tree, subsequent roles for a given resource-principal combination can be taken directly from the cache.

[0061] A set of satisfied roles is then returned to the framework in ③. The framework can provide the information from ① and ③ to the authorizer modules in ④. The authorization modules individually determine whether or not a policy is satisfied based on this information and their own criteria. In another embodiment, each authorizer can implement a cache to speed up searching for policies. Rather than traversing a resource tree to find a policy within scope, each authorizer can cache policies that were previously retrieved from a resource tree based on a key comprising the resource to which access is requested and the principal. After the initial retrieval from a resource tree, subsequent policies for a given resource-principal combination can be taken directly from the cache. The authorizer results (e.g., in terms of grant or deny decisions) are provided to the framework in ⑤ and provided to the adjudicator in ⑥. The adjudicator makes a final decision which it provides to the framework in ⑦. The framework then provides this decision to the calling process in ⑧.

[0062] As enterprise applications grow large and complex, so do the number of administrative tasks. One way to reduce the number of tasks that a system administrator is responsible for is to distribute the tasks among a number of administrators. Delegated administration allows a hierarchy of roles to manage administrative capabilities. By way of a non-limiting example, administrative capabilities can include the ability to manage customer accounts, the ability to delegate administrative capabilities, the ability to customize or personalize user interface elements (e.g., portals, booklets, desktops, portlets, etc.), the ability to perform administration of an enterprise application, etc. In another embodiment, any capability or property can be delegated. In one embodiment, delegation is an act whereby a principal in one role enables another hierarchically inferior role to have an administrative capability and/or further delegate an administrative capability. In one embodiment, a delegation role is identical to a role and can thusly be defined using predicates (e.g., user, group, currentDate, segment, etc.).

[0063] **FIG. 4** is an illustration of a delegation role hierarchy in accordance to one embodiment of the invention. In one embodiment, delegation roles can be organized into a delegation hierarchy to control the extent of delegation. In one embodiment, delegation roles can be associated with a single top-level resource, such as an enterprise application, and a delegation role hierarchy can be maintained separate from the resource hierarchy. A security policy can be associated with the enterprise application to limit which principals are allowed to alter the role definitions and the separately maintained role hierarchy. In another embodiment, a fictitious resource hierarchy that mirrors an arbitrary delegation role hierarchy can be utilized whereby each delegation role is associated with a resource corresponding to the delegation role's proper position in the hierarchy. A security policy can be associated with each resource to control which principals can modify the associated role. A security policy at the root of the hierarchy could limit which principals are allowed to modify the fictitious hierarchy itself.

[0064] Referring again to **FIG. 4**, role Admin_Role is at the top of the delegation role hierarchy. In one embodiment, the principal in this role has no limitations in its adminis-

trative capabilities or delegation authority. By way of a non-limiting example, a principal in the Admin_Role can modify the definition of delegation roles and the delegation hierarchy. In one embodiment, a principal in a delegation role can delegate administrative capabilities only to roles beneath it in a delegation hierarchy. Admin_Role has two children, A_Role and B_Role. A_Role has one child, C_Role, which as two children: D_Role and E_Role. By way of a non-limiting example, Admin_Role can delegate to all other roles beneath it in the hierarchy. Likewise, A_Role can delegate to C_Role, D_Role and E_Role. Whereas C_Role can only delegate to D_Role and E_Role. The leaves of the tree, D_Role, E_Role and B_Role cannot delegate since they have no children. In another embodiment, a node in the hierarchy can be related to more than one parent. This allows more than one superior role to delegate to an inferior role.

[**0065**] In one embodiment, a delegation can be represented by a security policy. The policy is associated with a delegated resource/capability and is based on the role to which the resource/capability was delegated. **FIG. 5** is an illustration of exemplary delegation security policies in one embodiment of the invention. Assume for this example that the delegation hierarchy of **FIG. 4** holds. Notice that the root resource in **FIG. 5**, Enterprise App **1** is associated with the following roles: Admin_Role, A_Role, B_Role, C_Role, D_Role and E_Role. The hierarchy depicted in **FIG. 5** could include other resources, roles and policies, but is limited for illustrative purposes. In one embodiment, a delegation creates a policy on the resource who's capability is being delegated. For example, resource Web App **1** has an Admin capability and an associated security policy P(D_Role). A principal in the role of C_Role, A_Role or Admin_Role created this policy by delegating to D_Role the Admin capability for Web App **1**. (It will be apparent to those of skill in the art that any capability can be delegated; i.e., not just Admin.) Thus, principals that satisfy D_Role can perform administration of Web App **1**. However, since Web App **1** does not have a delegation capability, a principal satisfying the D_Role cannot further delegate Web App **1**'s Admin capability.

[**0066**] Resource Desktop A has two capabilities, Admin and Delegate, each of which has a policy. The policy P(A_Role)attached to both indicates that a principal in the role of Admin_Role delegated to Role_A the capability to both administer Desktop A and further delegate this capability. Thus, a principal in Role_A can further delegate both the Admin and Delegate capabilities to hierarchically inferior delegation roles (i.e., C_Role, D_Role and E_Role). For example, resource Desktop B has a capability Admin that has a policy P(C_Role). This policy was put in place by a principal in the role of A_Role or Admin_Role. A principal in the role of C_Role will be able to administer Desktop B, but will not be able to further delegate this capability.

[**0067**] In one embodiment, a delegation to a node that is already delegated to by a principal in a hierarchically superior delegation role is not permitted. Referring to **FIGS. 4 and 5**, and by way of a non-limiting illustration, if resource Portal **2** had a policy P(A_Role), a principal in the role of C_Role would not be able to delegate Portal **2** since it had been delegated to a role superior to C_Role (i.e., A_Role).

[**0068**] In another embodiment, aspects of user group administration can be delegated. By way of a non-limiting example, user groups can by organized into a hierarchy by viewing them as children of an enterprise application resource. Capabilities that can be delegated include: user profile administration, the ability to view the members of group, and the ability to create, update and remove users and groups.

[**0069**] One embodiment may be implemented using a conventional general purpose or a specialized digital computer or microprocessor(s) programmed according to the teachings of the present disclosure, as will be apparent to those skilled in the computer art. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art. The invention may also be implemented by the preparation of integrated circuits or by interconnecting an appropriate network of conventional component circuits, as will be readily apparent to those skilled in the art.

[**0070**] One embodiment includes a computer program product which is a storage medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the features presented herein. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

[**0071**] Stored on any one of the computer readable medium (media), the present invention includes software for controlling both the hardware of the general purpose/specialized computer or microprocessor, and for enabling the computer or microprocessor to interact with a human user or other mechanism utilizing the results of the present invention. Such software may include, but is not limited to, device drivers, operating systems, execution environments/containers, and user applications.

[**0072**] The foregoing description of the preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. Embodiments were chosen and described in order to best describe the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention, the various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents.

What is claimed is:

1. A method for controlling access to a resource in a distributed computing environment, comprising:

receiving a request for a principal to access the resource;

determining a policy that is appropriate for the principal given the resource;

determining whether access to the resource is allowed given the policy;

wherein the policy is associated with a first resource in a hierarchy of resources; and

wherein the policy can supersede a second policy associated with a parent of the resource in the hierarchy of resources.

2. The method of claim 1 wherein:

a policy includes one or more expressions.

3. The method of claim 1 wherein the step of determining whether to allow access to the resource includes:

evaluating the policy.

4. The method of claim 1 wherein:

a resource is part of an enterprise application.

5. The method of claim 1 wherein:

a resource can inherit a policy.

6. The method of claim 1 wherein:

the policy evaluates to true or false for the principal.

7. The method of claim 1 wherein:

a policy includes one or more predicates.

8. The method of claim 1, further comprising:

responding to the request.

9. A machine readable medium having instructions stored thereon to cause a system to:

receive a request for a principal to access a resource;

determine a policy that is appropriate for the principal given the resource;

determine whether access to the resource is allowed given the policy;

wherein the policy is associated with a first resource in a hierarchy of resources; and

wherein the policy can supersede a second policy associated with a parent of the resource in the hierarchy of resources.

10. A system for controlling access to a resource in a distributed computing environment, comprising:

a security framework capable of receiving a request for a principal to access the resource;

a component coupled to the security framework and capable of determining whether access to the resource is allowed given the policy;

wherein the policy is associated with a first resource in a hierarchy of resources; and

wherein the policy can supersede a second policy associated with a parent of the resource in the hierarchy of resources.

11. The system of claim 10 wherein:

a policy includes one or more expressions.

12. The system of claim 10 wherein:

determining whether to allow access to the resource includes evaluating the policy.

13. The system of claim 10 wherein:

a resource is part of an enterprise application.

14. The system of claim 10 wherein:

a resource can inherit a policy.

15. The system of claim 10 wherein:

the policy evaluates to true or false for the principal.

16. The system of claim 10 wherein:

a policy includes one or more predicates.

17. The system of claim 10 wherein:

the security framework is capable of responding to the request.

* * * * *