The Resource Access Decision Facility

In Search of an Architecture for Enterprise Access Management

www.2ab.com

Overview

Information is distributed; when there is a need to access medical, financial, legal or criminal information about a person, the task is ominous. The same is true of corporate and government information. The sources of the information are vast, and the changing environment – mergers & acquisitions in the commercial world and the need to collaborate between agencies and establish coalitions across governments – creates a catalyst for providing as much information as possible on-line. This fact makes distributed object (component) technology particularly appropriate for the development of a service architecture that accommodates identification of people, location of information about people and controlled access to such information.

Increasingly, the biggest problem that organizations face is the management of the security infrastructure that controls access to sensitive and/or confidential information. It is important to understand that there are many aspects of Security. Securing a network from unauthorized access is important but will do very little to control access to information that is classified and/or confidential when a person already has access to the network. Access Control ensures that once a person is identified and provided entry to your network that they are only allowed access to the information that they are authorized to see. These security policies are often based on roles, clearances, entitlements and/or relationships of (or between) individuals and/or groups of individuals.

Managing secure implementations through a well-defined architecture that respects separation of concerns is crucial to implementation of a security architecture that can be understood by business leaders, managed by security administrators and audited against increasingly stringent requirements of legislators and consumers. This paper will focus on only a single aspect of security - the necessary capabilities of a framework that facilitates access control decisions based on application domain factors while maintaining the separation (or de-coupling) of authorization logic from application business logic. This is the area that has traditionally been known as “application-level” security. Government, healthcare, financial and telecommunication domains require a framework that supports fine-grain resource control as demanded by the privacy and confidentiality constraints of federal legislation.

Unfortunately, application developers (ISV or end-user) have enough challenges meeting the functional requirements of providing business software; they do not want to spend precious development resources on access control logic, nor is this their area of expertise. Furthermore, enterprises deploying business applications cannot continue the proliferation of access control mechanisms – each often unique to the application. They need to define enterprise policies and then define how those policies apply to control of the secured resources stewarded by their business applications. Users want this to be done consistently across all software components (whether purchased or built internally). That is, a standardized access control framework is necessary so that access control can be “plugged in” to business applications. The Object Management Group has developed a standard for fine-grain access control that provides such a framework. This paper explores the Resource Access Decision Facility (RAD) and how it can be leveraged to simplify Enterprise Access Management.

The problem addressed by the Resource Access Decision Facility (RAD) is not a new one. The catalyst for this facility was a realization that access control was becoming increasingly unmanageable in enterprise application integration environments. Vendors are spending an increasing percentage of their development time building access control into their applications. This is accomplished in a variety of ways with the obvious problem that each time an enterprise purchases a software component, they are also purchasing an access control mechanism that must be deployed and administered as part of their security infrastructure. This fact has made it impossible for enterprises to design and implement consistent application resource access control policy. The requirements of many government regulations, such as the U.S. Health Insurance Portability & Accountability Act (HIPPA) and U.S. Title 47 in telecommunications, make it mandatory that this problem be solved.

The OMG Resource Access Decision Facility (RAD) provides for the de-coupling of authorization logic from application logic, allowing applications with such requirements to be independent from a particular access control policy. RAD provides a number of key design features that will be discussed in this paper. It is important to understand that although the Resource Access Decision Facility was initially based on the CORBA® platform, the model and design approach can be successfully used in any distributed computing environments. 2AB's iLock Security Services product suite leverages this model for fine-grain access control within JAVA and J2EE (jLock), Web Services (webLock) and CORBA (orbLock) application environments.

The RAD design extends the underlying security infrastructure that provides authentication of users and provides the ability of an application to protect any resources stewarded by application logic. It supports the naming of resources and the definition of patterns for resource names in a standardized format to facilitate management of fine-grain access control policy at the level of granularity required by an application end-user community. It also allows the definition of arbitrary operations on these resources and the independent protection of those operations. The framework provides administrative interfaces that allow access control policy engines to be "plugged in," thus accommodating integration of existing policy engines and/or user-written policy evaluators. A "plug-in" can also provide dynamic security attribution to support policy that is based on transient relationships. A typical example of a dynamic attribute is "primary care physician"; a security attribute that is based on the relationship between a physician and the person for which clinical information is requested at a point in time. This "plug-in" framework approach enables elaborate and consistent access control policies across heterogeneous software components. The RAD framework was designed to accommodate environments where multiple policies govern access to a resource (such as an administrative policy and a legal policy). In such environments, it is necessary to understand how to combine policies to make access decisions. This feature is also part of the RAD "plug-in" architecture.

Why do we need RAD?

Before we explore the capabilities of RAD in more detail, let's look at why we need such a facility and how these capabilities extend what is provided by a Security Service. A typical distributed computing solution integrates a variety of technologies (see Figure 1).

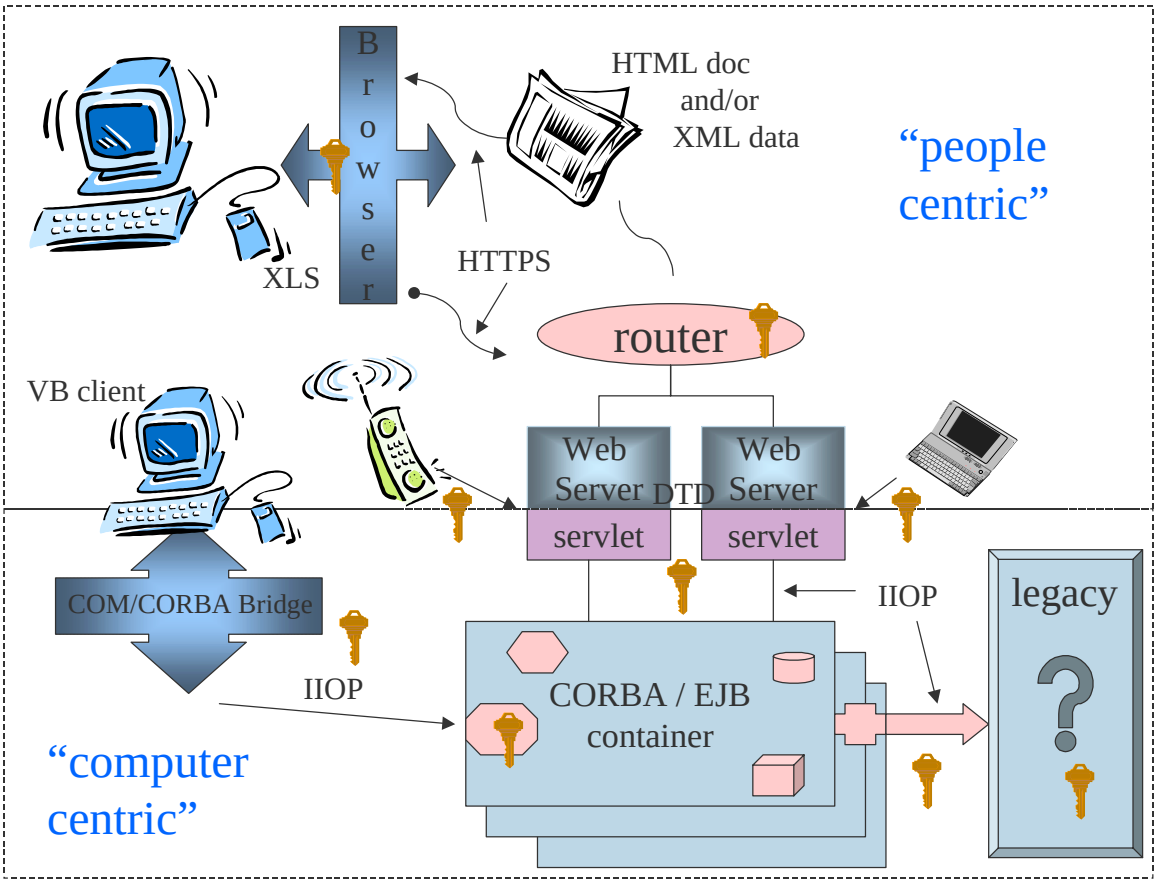

Figure 1 –Typical Distributed Computing Solution

Access control policy can be injected at any point in the architecture. The RAD facility was originally intended to be used by services providing application features, but increasingly RAD facilities are being used on the client side where transitions are made between Web and J2EE and/or CORBA® technologies. It has proven particularly useful for environments where data defined in Extended or Hypertext Markup Languages (XML or HTML) documents contain multiple secured resource that may have diverse access control requirements.

In service architectures, an operation on an object may need to reference many secured resources to provide the requested service. This cannot be secured with a traditional security service. It is also common to design abstract interfaces where the identity of the secured resource is not known until the operation is invoked. This is because the information necessary to identify the secured resource is carried in a parameter value. For example, different parts of a government file may require diverse clearance levels because of the sensitivity of the information. These security policies may need to change dynamically in war situations. A common health related example is a request for a clinical observation – a clinical observation is an abstract concept that may be realized as many diverse secured resource types that can be accessed via a common interface. The sensitivity of a 'clinical observation" is not consistent; for example, Anthrax related information may be more sensitive than other observations. In addition, Web Services, EJB's and/or CORBA® objects often function as wrappers around legacy systems that do not lend themselves easily to object-oriented access control mechanisms. It is also true that security requirements in many domains mandate domain-specific factors such as the relationship between a user and the person for whom information is requested. Increasingly, these requirements are causing vendors to embed access control systems within their business applications. These complex security requirements mandate access

control policies that are more sophisticated and of finer granularity than the general ones used in security services of existing distribution platforms.

What is RAD?

RAD is a framework that facilitates access control decisions based on application domain factors while maintaining the separation (or de-coupling) of authorization logic from application business. The RAD facility is not a replacement or substitution for an infrastructure security service. The RAD service is used in conjunction with other security infrastructure to provide enhanced access decisions. Access to authenticated credentials from a security infrastructure that supports delegation is the foundation that any application needs to provide application-level security and is required in a RAD environment. The designers of the RAD service wanted to be certain that it could be deployed with diverse security infrastructures. For this reason, the Resource Access Decision Facility and the underlying security infrastructure are loosely coupled. The only dependency that exists is that the application must be able to extract authenticated credentials (security attributes) from the security service and format them as an OMG Security::AttributeList. Vendors, such as 2AB, provide such translation tools; the most common being the extraction and transformation of credentials carried by digital certificates. Usage of a RAD facility removes the requirement that the application developer understand which security attributes are necessary to allow access to a secured resource. That is, the access control mechanism may be developed, purchased and administered separate from the application and integrated at deployment time.

This RAD approach allows secured resources to be named by the application. Those names are exposed to the RAD policy engine (potentially mapped by the vendor to the native format of a resource name in the policy engine). This allows an enterprise a consistent way to identify and administer access policies across diverse application environments. It also means that different enterprises (or organizations within an enterprise) may make different choices in terms of the policy engine that they use with the RAD framework. In fact, multiple policy engines may be used with a single implementation of the RAD framework as a result of the support for multiple policy evaluators.

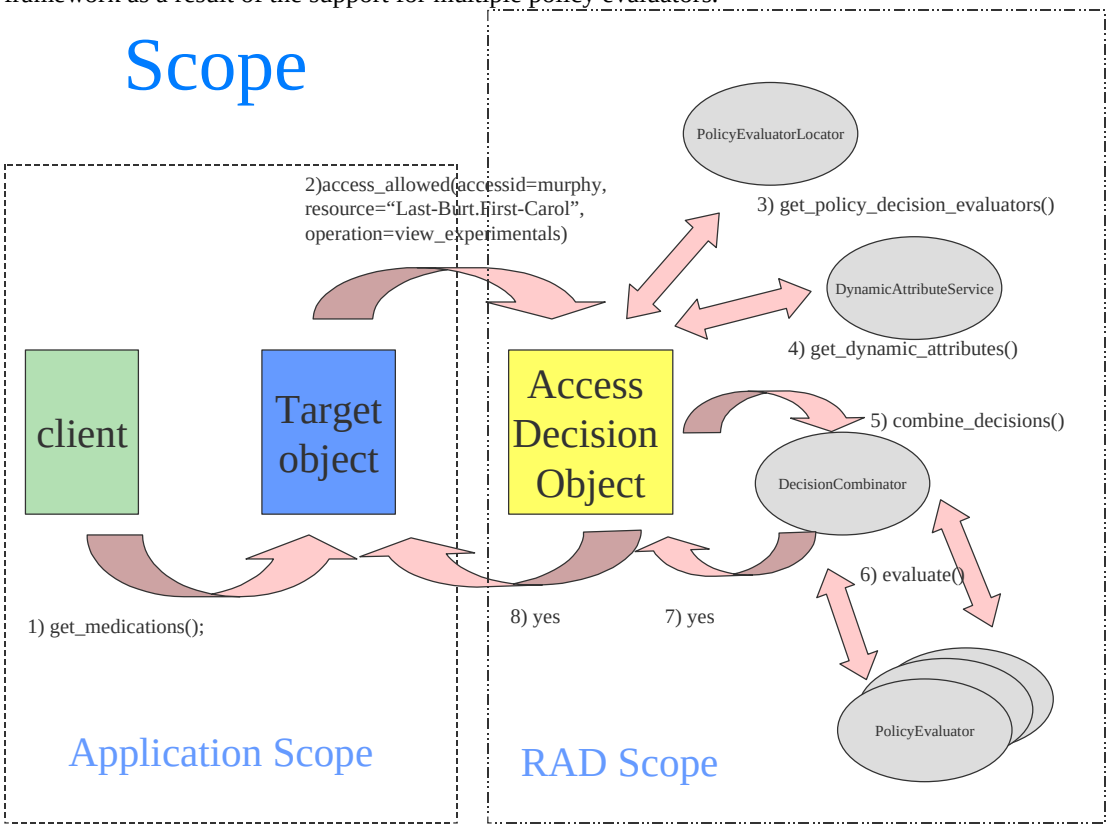

Figure 2 – RAD Scope

To perform application-level access control, an application (ADO client) extracts credentials from the security infrastructure, requests an authorization decision from a RAD facility and enforces that decision. A simple interface between the application and the authorization service is used. There are two operations available to the ADO client: access_allowed() and multiple_access_allowed(). An application programmer only needs to make a single invocation on the authorization service in order to obtain a decision (or set of decisions).

A RAD-compliant access control environment divides responsibilities as follows:

- Components "name" their secured resources and the operations they perform on them
- The underlying security service provides the component access to authenticated credentials
- Users administer security policy for the named resources (via a policy engine accessible by RAD)
- Components call access_allowed() providing the ResourceName, Operation and SecAttributes
- RAD makes the access decision!

A resource name can be associated with any valuable asset of the application. The RAD does not attempt to interpret semantics of the resource name or the operations that can be performed on them.

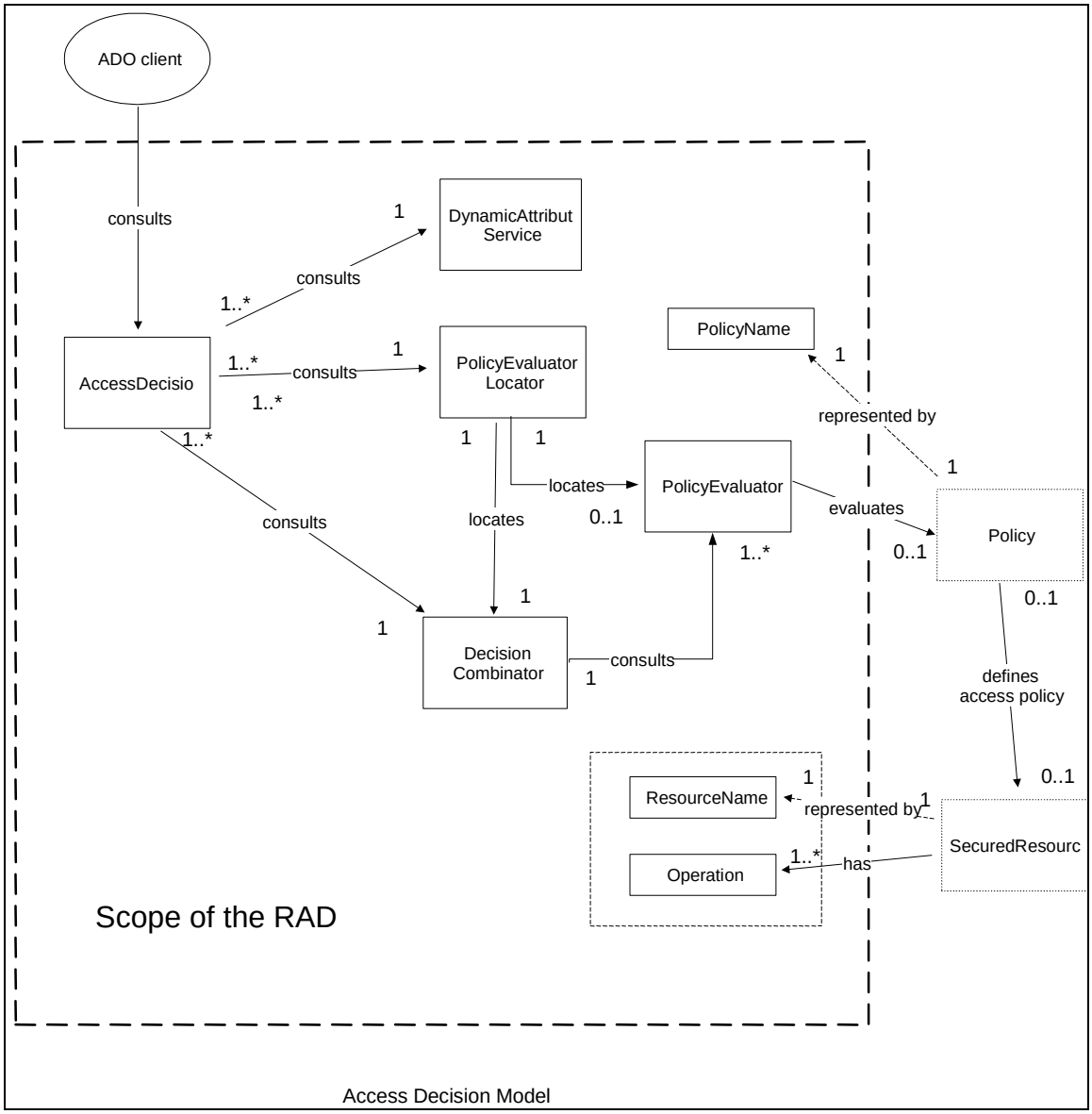

Figure 3 – Access Decision Model

Figures 3 and 4 provide an overview of the responsibility of the internal RAD objects. The AccessDecisionObject (ADO) receives requests for authorization decisions from RAD clients. From an application perspective, the ADO is the only exposed interface, however, internal to the RAD, the ADO consults other objects (each of which may be replaced using the administrative interfaces) to make access decisions. The ADO first consults a DynamicAttributeService that determines whether or not it is appropriate to modify the security attributes of the principal given the context of the access request. The DynamicAttributeService may add or remove security attributes that are then used for the access decision. The ADO also consults a PolicyEvaluatorLocator that locates the PolicyEvaluator(s) and the DecisionCombinator that must be consulted to make the access decision for the secured resource. The role of the DecisionCombinator is to combine results of the evaluations made by PolicyEvaluator(s) into a definite yes/no decision. The combinator calls the evaluators so that the most efficient method of making decisions can be utilized. It is expected that combinators that implement simple "AND" or "OR" policies will be provided with products, however, the user may replace default combinators with sophisticated implementations that use precedence logic.

Of course, there is also an administrative aspect to the RAD that provides the "plug-in" features mentioned earlier (see Figure 3). Some of the advanced RAD features that are available through the administrative interfaces are:

- Provides the ability for secured resources to be grouped for the purpose of defining access control rules (Patterns)
- Supports dynamic security attribution to allow access policy to include the notion of allowing decisions based on relationships or transient roles (DynamicAttributeSevice)
- An interface that allows multiple access control policy decisions governing access to the same resource to be reconciled (DecisionCombinator)
- Ability to Plug in Policy engines! (PolicyEvaluator)
  - Custom wrappers for existing vendor products and user legacy solutions
  - Custom built for the way resources are named in specific technologies (CORBA® IDL, XML Schema)

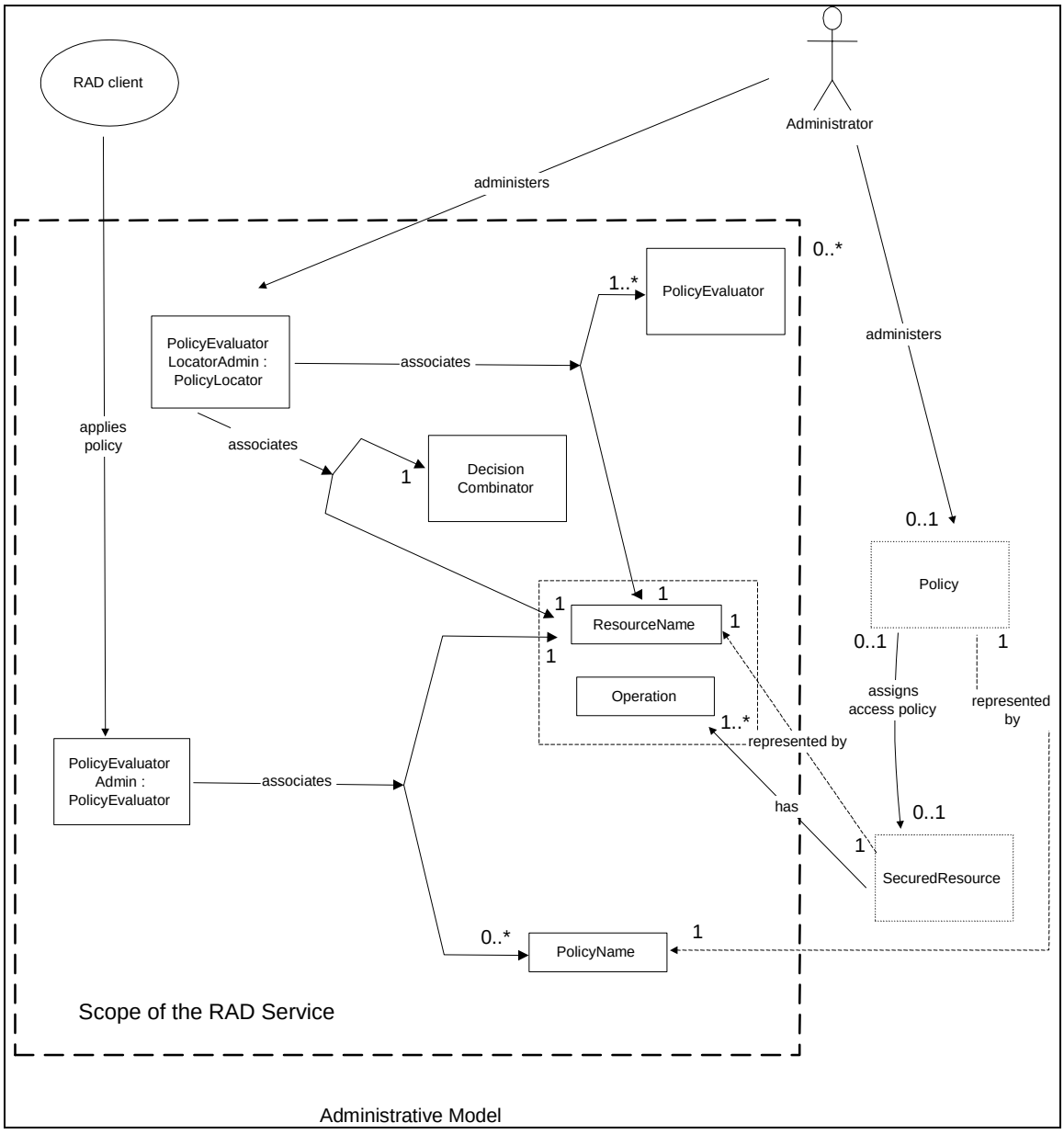

Figure 4 - Administrative Model

Summary

The specification for the RAD facility is available from the OMG[1]. This specification, originally targeted to meet the requirements of the healthcare domain for privacy and confidentiality, is receiving wide acceptance in other domains such as defense, telecommunications and finance. Products, such as 2AB's iLock Security Services suite, leverage this specification for fine-grain access control in multi-platform environments.

[1] Object Management Group Resource Access Decision. Document formal/01-04-01. January 2001.

How do I get further information?

This White Paper has intentionally only scratched the surface of RAD. Further information can be obtained by phone, fax or e-mail

Phone: USA +1 877 334-9572
Fax: USA +1 205 621-7455
E-mail: info@2ab.com.com

Comments on this White Paper or on the iLock Security Service suite of products are welcome. Please use the above contacts.

General information about iLock Security Services or any other 2AB products and professional services can be found at:

Web: http://www.2ab.com/

General information about the Object Management Group and CORBA can be found at:

Web: http://www.omg.org/