



Using OMG's Resource Access Decision (RAD) Facility Model for Governance-Based Access Control (GBAC)

August 2005
www.2ab.com

Introduction

Access management is a simple concept. Every business has information that needs to be protected from unauthorized disclosure. To protect information, companies define policies - often mandated by legislation - that govern who can access specific classes of business and/or personal information. For example, if a police investigator seeks to access transaction data related to a suspected terrorist from a private bank, they should have authorization to do so, however, they should not be authorized to access the same information about someone who is not suspected of any illegal activity. That is, there is a policy that specifically governs the release of information collected about individual to other individuals. Or is there? The answer today is: "Probably not." What exists are written policies (or a law) related to disclosure of broad classes of business and personal information. But often individual data is not specifically classified within organizations. Requests for information flow through individuals within an organization. The policy is enforced only because human beings are skilled at generalizations; that is, we expect someone in authority to be able to classify an ad hoc request for a particular piece of information and make a decision.

Access Management software has a simple goal. It allows the human who previously acted as a guardian of sensitive information to be removed from the process without loss of access control. This sounds simple, but most businesses are struggling with the implementation of access management as they integrate and extend their applications. This is because machines cannot classify information or make access decisions unless they are explicitly programmed with algorithms to accomplish this. When you take the responsibility for access decisions away from human beings, it becomes necessary to insert software guards into your applications.

CGI has recently released a whitepaper titled *"Governance-Based Access Control: Enabling improved information sharing that meets compliance requirements,"* introducing a new model for access control. This model, called Governance-Based Access Control (GBAC) is focused on the classification of information assets for the purpose information sharing in an environment where:

- Many organizations may require access to information
- Information may be accessed by, or shared with, external users
- Everyone may be subject to compliance with multiple authorities and jurisdictions

In a previous whitepaper, *"Using jLock's Java Authentication and Authorization Service (JAAS) for Application-Level Security,"* we described a systematic approach to using JAAS to manage the complexity associated with software access management. We explained the JAAS service-oriented architecture (SOA), which maintains a clean separation of concerns between application functionality and access management and discussed the types of JAAS features that a scalable JAAS implementation should support. We also wrote a complementary paper showing how JAAS could be used for GBAC.

In this whitepaper, we explore the Governance-Based Access Control (GBAC) model and use the scenarios presented in the CGI whitepaper as the basis of the sample access policy and the coding examples. The OMG RAD model is defined in IDL. This enables an implementation to be used from a variety of programming environments. The examples here are in Java. We hope the 2AB papers will help you understand the GBAC model and how existing standards and tools can be leveraged to implement it.



What is the Resource Access Decision Facility?

The OMG's Resource Access Decision (RAD) facility 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 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. The framework also 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 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 model addresses many of the issues faced by organizations that are seeking to implement application security.

Application security must address any security-related requirements not provided by the runtime security infrastructure. In the area of access management, any requirement to restrict a) the usage of application features or b) access to business and personal information is part of "application security." Often these restrictions on access to sensitive information are based on legislation. For this reason, the Governance-Based Access Control Model is being progressed as a means to classify information for the purpose of assigning access policy. The CGI whitepaper "Governance-Based Access Control (GBAC)" provides an excellent overview of this model and is the basis of the examples in this whitepaper.

There are overviews of the Resource Access Decision facility available on the OMG Web site (www.omg.org), as well as whitepapers available from 2AB. For that reason, we will assume the reader has some familiarity with the RAD model, and we will focus on how the RAD model can be leveraged to provide the fine-grain access control requirements of "application security" in an environment that uses the Governance-Based Access Control (GBAC) model.

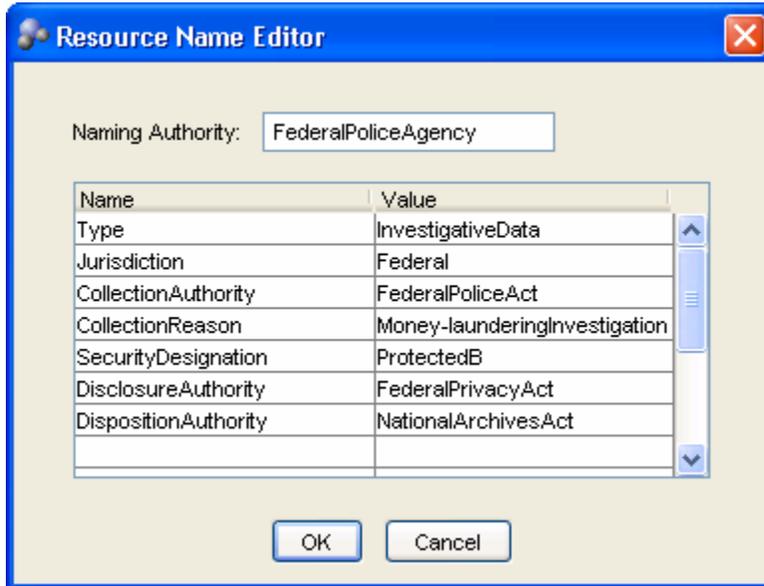
The Resource Access Decision facility and the underlying security infrastructure used for authentication are loosely coupled. For this reason, the application (or product provided to the application) must be able to extract authenticated credentials (security attributes) from the security service and format them as an `OMG Security::AttributeList`. Vendors, such as 2AB, provide such translation tools; the most common being credentials acquired via `login()` or the extraction and transformation of credentials carried by digital certificates.

This paper will focus on how a Java developer uses a commercial implementation of the OMG RAD model in conjunction with the GBAC model. We will show how the RAD model can be used to support the context-sensitive policy requirements of GBAC. We will also explore how user identity and access policies are managed using iLock Security Center administrative tools. We will outline the steps you need to take to use iLock with the GBAC model.



Classification of Information

Each classification of information is mapped to an OMG Resource Name - as defined in the OMG Resource Access Decision (RAD) facility specification. For this reason, they show up as "RAD Resources" or simply "Resources" in the iLock administrative tool. Using this specification allows you to leverage a "NamingAuthority" for each group of resources. For example:



Note: RAD resource names are structural. This structure maps neatly to the GBAC classifications as it supports the ability to assign attributes for information classifications that are intuitive, visible in the names and can be mapped easily to an XML schema. A RAD resource is defined for each GBAC classification as shown in Table 1 of the GBAC Whitepaper.

This iLock user interface supports the OMG RAD concepts, which includes structural naming.

Defining a GBAC Classification as a RAD Resource



Viewing the GBAC Classifications as RAD Resources



Classification of people who may be required to access information

The classification of people in GBAC is based on roles. In the OMG's Resource Access Decision (RAD) facility, a role is a type of security attribute that may be assigned to users.



The dialog box titled "New Defining Authority" has a blue header with a close button (X) on the right. On the left is a help icon (question mark in a speech bubble). The main area contains the label "Defining Authority Name:" followed by a text input field containing "2ab". At the bottom are "OK" and "Cancel" buttons.

The OMG's Resource Access Decision facility uses a structural representation for security attributes. This structure includes a field called a **Defining Authority Name**. You have the option of creating unique names (Defining Authorities) for the different agencies that will access information classified in the GBAC model. This would allow security attributes (Access IDs, Groups and Roles) to be scoped to the organization that defines them. If you choose to do this, you should

be aware that "roles" with the same name created under two different Defining Authorities are distinct security attributes, and the appropriately scoped role must be utilized in the access policy. This scoping of names would, of course, be useful if different agencies used the same role name but with different expectations regarding the access it would enable. For this example, we will keep it simple and use the default Defining Authority of "2ab" for all security attributes.

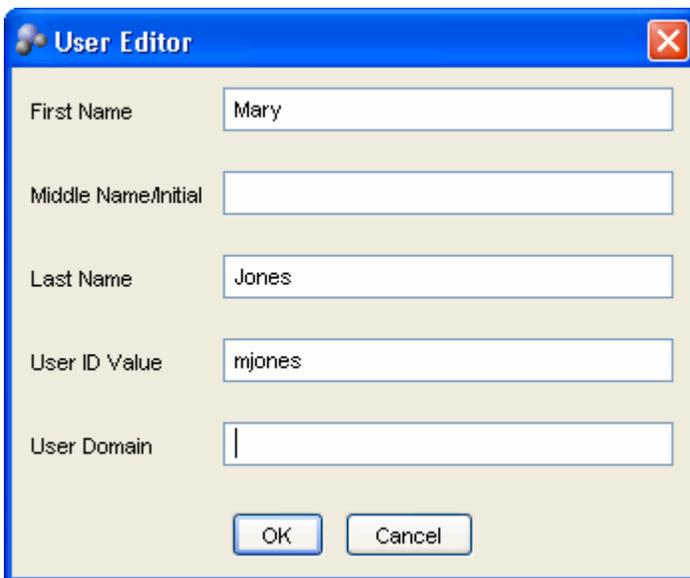
Here we see how GBAC concepts map to the OMG RAD specification concepts.



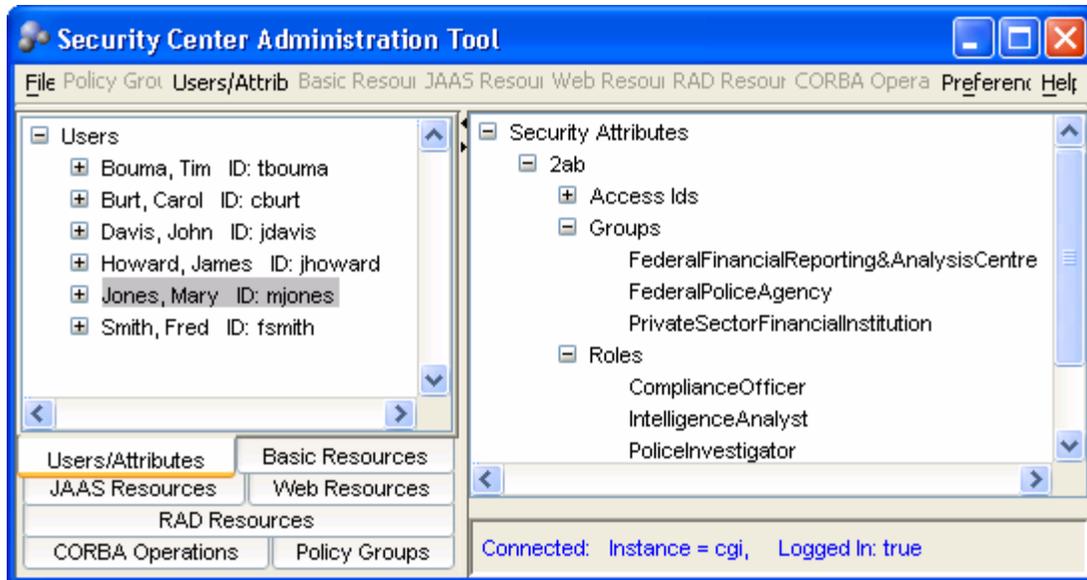
The dialog box titled "New Role" has a blue header with a close button (X) on the right. On the left is a help icon (question mark in a speech bubble). The main area contains the label "Role Name:" followed by a text input field containing "PoliceInvestigator". At the bottom are "OK" and "Cancel" buttons.

After defining authorities are created, roles are defined under them. By selecting and right-clicking "Role" under the 2ab defining authority, the **New Role** dialog box will allow you to type in a Role Name. Groups can be defined in a similar manner. These Security Attributes will be used in the GBAC rules created for each class of information.

Of course, you will also need to define Users. The iLock Security Center Administration tool allows you to create users and assign passwords. Password format requirements can be set using the **Preferences** menu.

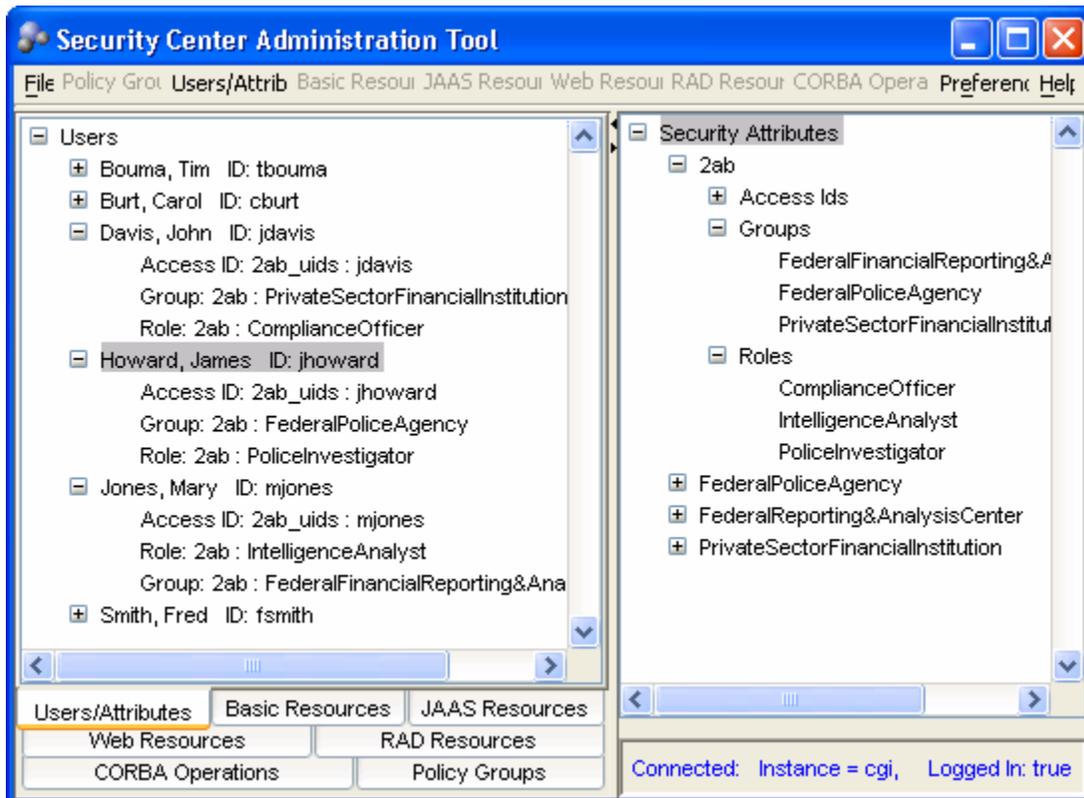


The dialog box titled "User Editor" has a blue header with a close button (X) on the right. It contains several text input fields: "First Name" with "Mary", "Middle Name/Initial" (empty), "Last Name" with "Jones", "User ID Value" with "mjones", and "User Domain" (empty). At the bottom are "OK" and "Cancel" buttons.



The Users/Attributes screen after users, groups and roles have been created.

We assign GBAC roles as follows: John Davis is a Compliance Officer, Mary Jones is an Intelligence Analyst and James Howard is a Police Investigator.



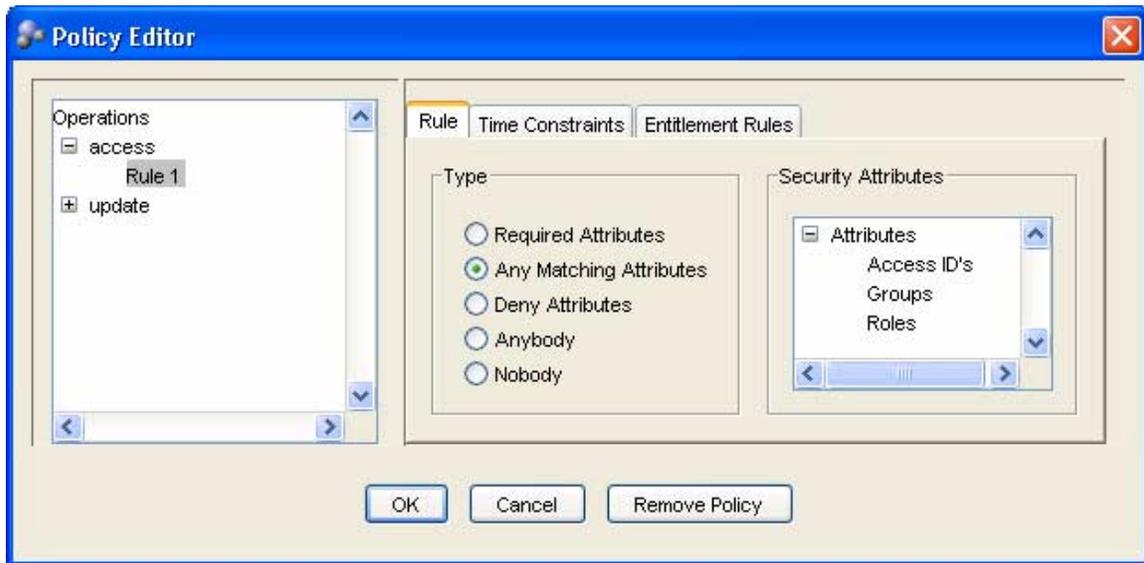
James Howard has the Role PoliceInvestigator and is in the Group FederalPoliceAgency



Definition of GBAC Rules for Information Access

Access Control rules are defined in GBAC based upon the roles of individuals and context information related to the information. The rules we define below are outlined in Table 2 of the GBAC whitepaper.

By returning to the RAD Resource tab, selecting a RAD Resource (these are the GBAC classifications) and right-clicking, you can bring up the **Policy Editor** dialog and create one or more "rules" as shown below.



To create Rule 1, as defined in Table 2 of the GBAC whitepaper, right-click on the Roles under Security Attributes and select the correct role for the Rule.

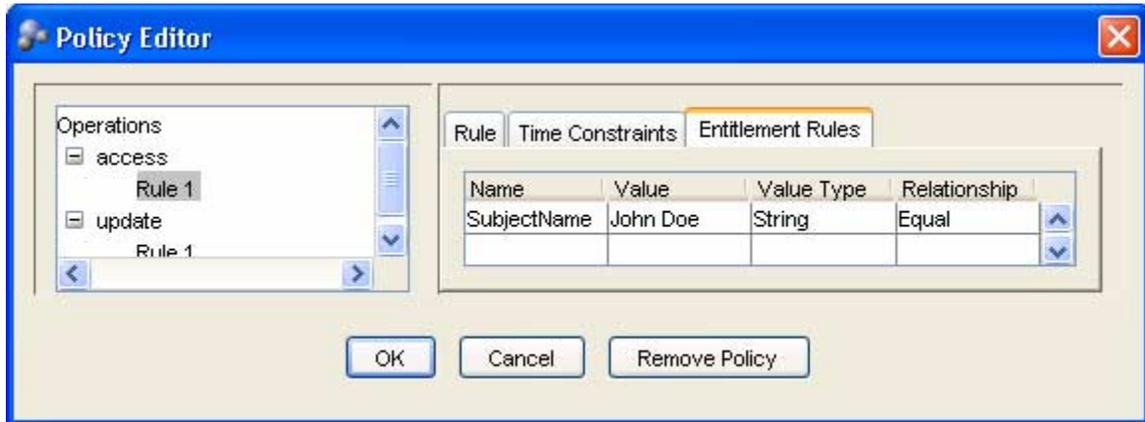
Note that the role is structural in the OMG specification. Drop down boxes allow you to select from previously defined roles.



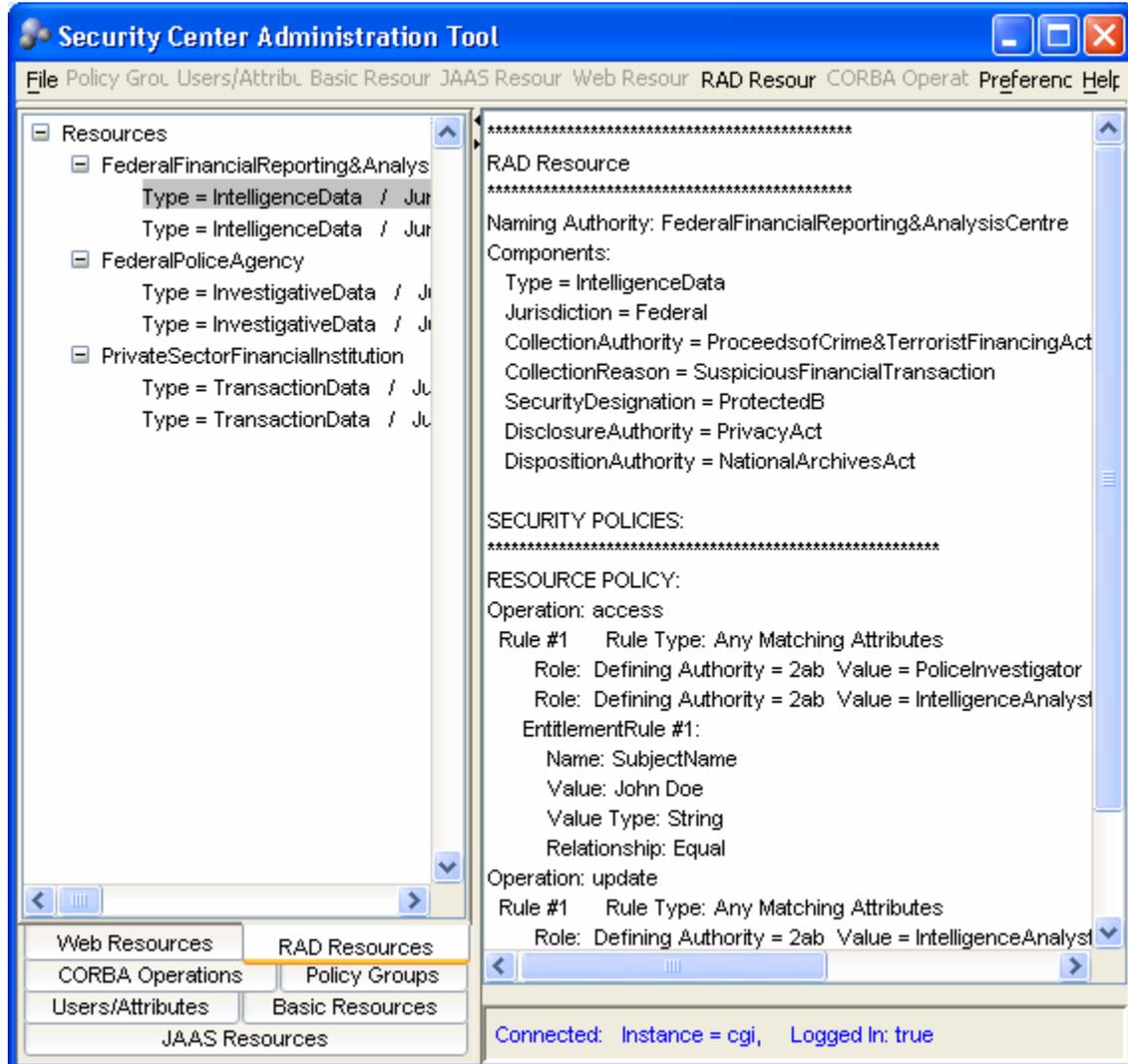
Select the correct Role for the GBAC rule



Next, we select the **Entitlements Rules** tab and enter the "additional criteria" from the rule as shown below.



This completes the creation of Rule 1 from the GBAC white paper. The final rule is shown in the rightmost panel below:



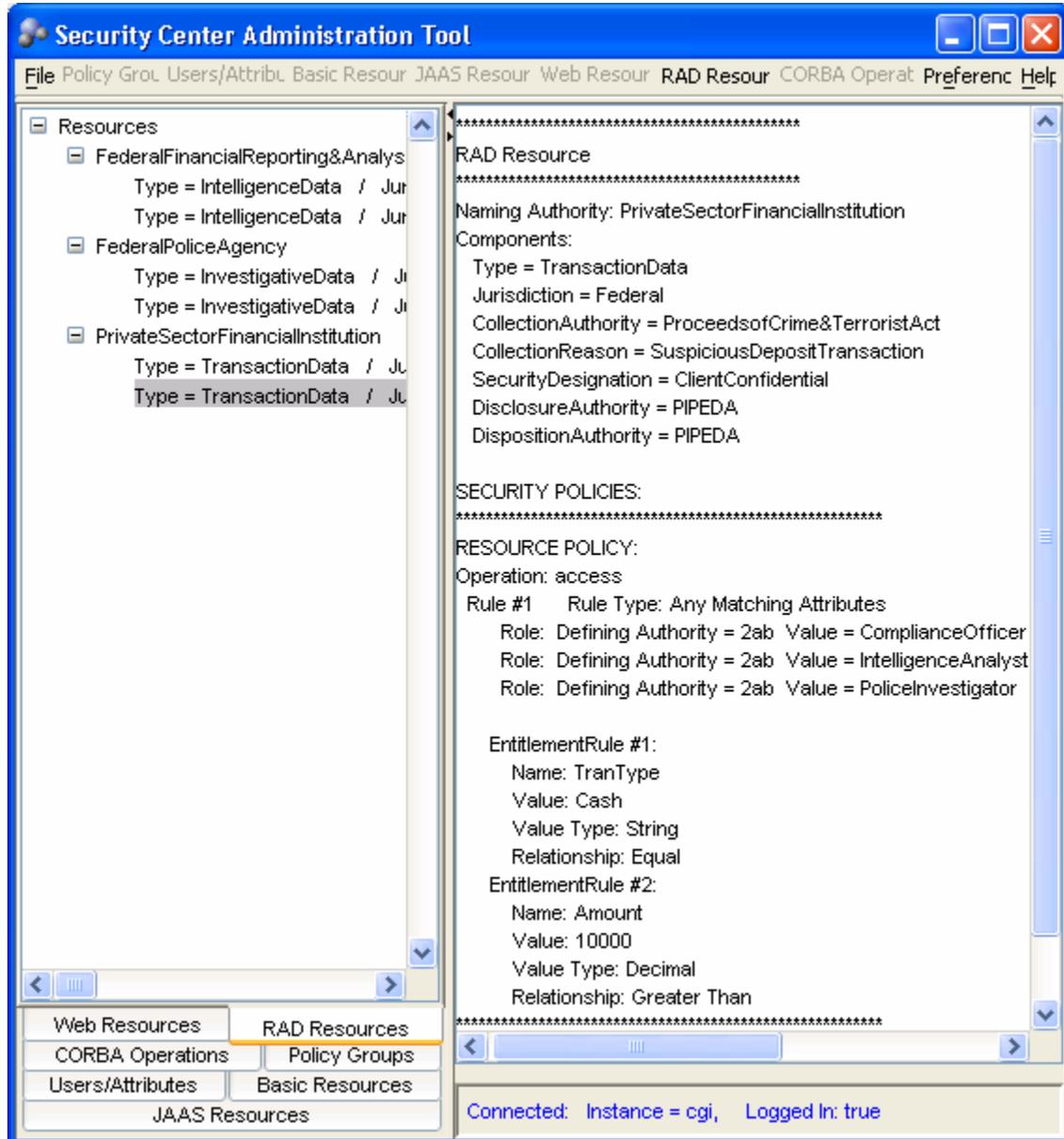


The policy states:

Anyone who has been assigned the role "PoliceInvestigator" or "IntelligenceAnalyst" as defined by "2ab" is allowed access (to the named resource) with the constraint that the "SubjectName" (which is of datatype String) is Equal to "John Doe."

Below is the example of another GBAC rule defined in iLock. The rule is stated as:

Anyone who has been assigned the role "IntelligenceAnalyst" or "PoliceInvestigator" or "Compliance Officer" as defined by "2ab" is allowed access (to the named resource) with the constraint that the "TranType" (which is of datatype String) is Equal to "Cash" and the "Amount" (which is of datatype Decimal) is Greater Than 10000.



Note that these policies are defined so that the local authority is always allowed access



Using the Java iLock Authenticator as part of a GBAC solution

Because the OMG RAD is an authorization service that is loosely coupled with the underlying authentication service, a variety of authentication APIs may be supported. Leveraging an architecture that supports ‘plug-ins’ for authentication ensures that applications can be independent of the underlying authentication mechanism. This has the advantage that new or revised authentication mechanisms can be plugged in without modifying the application code. That is, management of User IDs and Passwords (or other methods of authentication such as digital certificates) are removed from the application’s concern. Since we are providing sample code in Java, we will leverage the dialog-based User ID and Password authenticator that is supplied with the jLock product and leverage the Java iLock Interface (JII) API for authentication.

The first thing you need to do is specify the iLock security center you are using. This is done by initializing a LoginManager (and AccessManager) with properties that set the instance name of the Security Center repository that holds identity and policy information.

```
try {
    Properties props = new Properties();
    props.setProperty("jii.security.center.instance", "cgi");
    lm = new LoginManager(props);
    am = new AccessManager(props);
}
catch (Exception e) {
    System.out.println(e.toString());
    System.exit(1);
}
```

Initializing the LoginManager and the AccessManager

This is the Java code that prints “Hello iLock World” if user authentication succeeds. The method that your application needs to invoke to use a LoginManager that uses a dialog box authenticator is shown in **bold** font.

```
try {
    if (lm.loginFromDialog("") == true) {
        System.out.println("\nHello iLock World!\n");
    }
    else {
        System.out.println("*** Login Failed ***");
    }
} catch (Exception e) {
    System.out.println(e.toString());
    System.exit(1);
}
```

.....

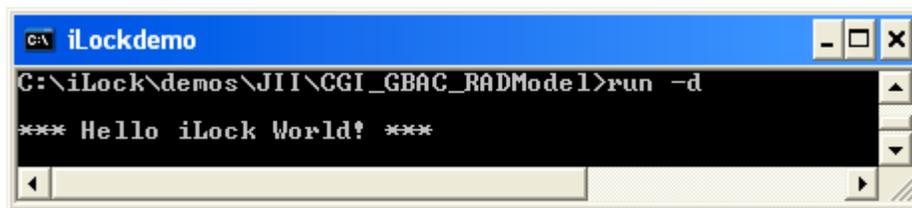
Authentication Code Sample

That is all the code and configuration you need! At the point where the **lm.loginFromDialog("")** is called, the following dialog will appear.



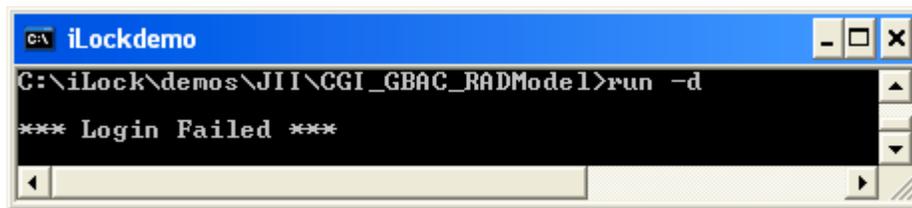
Type in a User ID and Password as shown above and click **OK**. iLock will authenticate the user.

Assuming you typed a valid User ID and Password, the example program results will, as you might expect, look like the following:



Authentication Succeeded

Of course, if you should fail to provide a valid User ID and/or Password, you will see this:



Authentication Failed

The HelloiLock demo program has obviously written no code to manage Users and Passwords or to do the work required to authenticate the user (in this case verify the password). That is the great thing about the RAD architecture; just “plug in” iLock, and it securely manages all that for you! iLock also ensures that the password is never available in clear text. iLock securely stores and transmits password information - even if you are not using an encrypted transport protocol.



Now we are ready to explore authorization in the RAD model. To understand the RAD Authorization model, you must first understand a little more about what happens when you authenticate. When the user (*mjones* in the example above) was authenticated, an array of **SecAttributes** was created. These security attributes are defined by the OMG Security Service and utilized by the Resource Access Decision (RAD) facility in evaluating access policy. The security attributes are structural in form and there are three defined types. The types are **AccessId**, **Group** and **Role**. The “security attributes” or “credentials” are associated with the User at the time of authentication. During the authentication process, the jLock authenticator acquires the credentials from the security center and associates them with the user by creating the appropriate SecAttribute array. A user may always be able to prove their identity, but their credentials (i.e. SecAttributes) may change over time. For this reason, security access policy is defined in terms of the security attributes that are associated with the user at the time identity was authenticated. These are the fundamental building blocks of access policy. In the section above, you can see that the user, Mary Jones, has the following jLock security attributes. Note that these attributes are scoped by a "Defining Authority" that indicates the source of the attribute. In this case, we use 2ab as the Defining authority. It would be possible, however to define security attributes with differing defining authorities. This would be consistent with the GBAC model where they might be defined and used in policy by multiple legal entities.

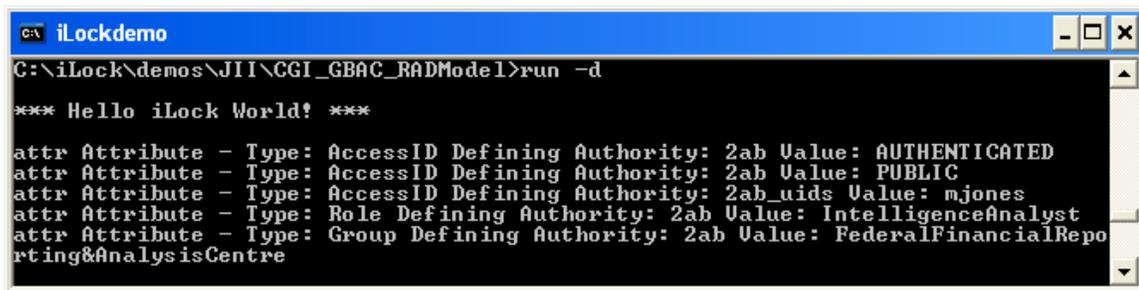
- AccessId: 2ab_uids: *mjones*
- Role: 2ab: *IntelligenceAnalyst*
- Group: 2ab: *FederalFinancialReporting&AnalysisCentre*

If you add the following code to the example, you can see that the LoginManager allows navigation to a SecurityAttribute [] that manages the set of SecurityAttributes.

```
.....
try {
    SecurityAttribute [] attrs = lm.getSecurityAttributes();
    for (int i = 0; i < attrs.length; i++) {
        System.out.println ("attr" + attrs[i].toString());
    }
}
```

Code to display the Security Attributes associated with the authenticated user

Running with this code, you will see the output below following authentication:





Using RAD Authorization for GBAC

The OMG's Resource Access Decision (RAD) facility was designed to enable the support of complex policy models such as CGI's GBAC. Because Resource Names are structural, they map neatly to the concept of Information Classification as outlined in the GBAC whitepaper. We do not explore the capabilities in this demo, but there is also the ability to define resource classifications based on name patterns and assign policy to a group of resources (GBAC classifications) in this manner. Unfortunately, like GBAC, the complexity of the RAD model makes it more difficult to grasp the power of the concepts (and how they can be applied) with some analysis. Information classification is a major hurdle to the adoption of a model such as the OMG RAD for authorization. Assuming this classification has been done, application of RAD to GBAC is straight forward.

```
ResourceComponent [] ic = new ResourceComponent [7];
ic[0] = new ResourceComponent("Type", "IntelligenceData");
ic[1] = new ResourceComponent("Jurisdiction", "Federal");
ic[2] = new ResourceComponent("CollectionAuthority",
"ProceedsofCrime&TerroristFinancingAct");
ic[3] = new ResourceComponent("CollectionReason", "SuspiciousFinancialTransaction");
ic[4] = new ResourceComponent("SecurityDesignation", "ProtectedB");
ic[5] = new ResourceComponent("DisclosureAuthority", "PrivacyAct");
ic[6] = new ResourceComponent("DispositionAuthority", "NationalArchivesAct");
gbac_class = new RadResource("FederalFinancialReporting&AnalysisCentre", ic);

EntitlementData ed = new EntitlementData[1];
ed[0] = new EntitlementData("SubjectName", "John Doe");

if (am.accessAllowed(gbac_class, "access", attrs, ed)) {
    System.out.println(" Granted access to \n" + gbac_class.toString() + "\n ");
} else {
    System.out.println(" Denied access to \n" + gbac_class.toString() + "\n ");
}
```

Code to protect access to the "IntelligenceDataProtectedB" information

It is that simple to add GBAC authorization to your application. We have provided demonstration programs that you can run to see how this works. The demonstration programs request access to all of the GBAC resource classifications named in the GBAC whitepaper. The sample code sets the subject name as John Doe, the transaction type to cash and the amount to \$12,000. Of course, you are free to use the samples as patterns for your own development.

When you run the demonstration program and authenticate using *mjones* as the User ID, you will see the following dialog showing the resources that Mary Jones, an IntelligenceAnalyst is allowed to access.



```
C:\iLockdemo>run -d
*** Hello iLock World! ***

attr Attribute - Type: AccessID Defining Authority: 2ab Value: AUTHENTICATED
attr Attribute - Type: AccessID Defining Authority: 2ab Value: PUBLIC
attr Attribute - Type: AccessID Defining Authority: 2ab_uids Value: mjones
attr Attribute - Type: Role Defining Authority: 2ab Value: IntelligenceAnalyst
attr Attribute - Type: Group Defining Authority: 2ab Value: FederalFinancialReporting&AnalysisCentre

Checking permission to access GBAC classified Information

Granted access to
Naming Authority: FederalFinancialReporting&AnalysisCentre Type = IntelligenceData
Jurisdiction = Federal CollectionAuthority = ProceedsofCrime&TerroristFinancingAct
CollectionReason = SuspiciousFinancialTransaction SecurityDesignation = ProtectedB
DisclosureAuthority = PrivacyAct DispositionAuthority = NationalArchivesAct

Granted access to
Naming Authority: FederalFinancialReporting&AnalysisCentre Type = IntelligenceData
Jurisdiction = Federal CollectionAuthority = ProceedsofCrime&TerroristFinancingAct
CollectionReason = SuspiciousFinancialTransaction SecurityDesignation = TopSecret
DisclosureAuthority = PrivacyAct DispositionAuthority = NationalArchivesAct

Denied access to
Naming Authority: FederalPoliceAgency Type = InvestigativeData Jurisdiction = Federal
CollectionAuthority = FederalPoliceAct CollectionReason = Money-launderingInvestigation
SecurityDesignation = ProtectedC DisclosureAuthority = FederalPrivacyAct
DispositionAuthority = NationalArchivesAct

Granted access to
Naming Authority: FederalPoliceAgency Type = InvestigativeData Jurisdiction = Federal
CollectionAuthority = FederalPoliceAct CollectionReason = Money-launderingInvestigation
SecurityDesignation = ProtectedB DisclosureAuthority = FederalPrivacyAct
DispositionAuthority = NationalArchivesAct

Denied access to
Naming Authority: PrivateSectorFinancialInstitution Type = TransactionData
Jurisdiction = Federal CollectionAuthority = ProceedsofCrime&TerroristAct
CollectionReason = StandardDepositTransaction SecurityDesignation = ClientConfidential
DisclosureAuthority = PIPEDA DispositionAuthority = PIPEDA

Granted access to
Naming Authority: PrivateSectorFinancialInstitution Type = TransactionData
Jurisdiction = Federal CollectionAuthority = ProceedsofCrime&TerroristAct
CollectionReason = SuspiciousDepositTransaction SecurityDesignation = ClientConfidential
DisclosureAuthority = PIPEDA DispositionAuthority = PIPEDA
```

Mary Jones, an IntelligenceAnalyst, requesting access to John Doe's information

However, if you run the demo and authenticate using *jhoward* as the User ID, you will see that James Howard, a PoliceInvestigator, is allowed to access the following classes of information:



```
C:\iLock\demos\JII\CGI_GBAC_RADModel>run -d
*** Hello iLock World! ***

attr Attribute - Type: AccessID Defining Authority: 2ab Value: AUTHENTICATED
attr Attribute - Type: AccessID Defining Authority: 2ab Value: PUBLIC
attr Attribute - Type: AccessID Defining Authority: 2ab_uids Value: jhoward
attr Attribute - Type: Group Defining Authority: 2ab Value: FederalPoliceAgency
attr Attribute - Type: Role Defining Authority: 2ab Value: PoliceInvestigator

Checking permission to access GBAC classified Information

Granted access to
Naming Authority: FederalFinancialReporting&AnalysisCentre Type = IntelligenceD
ata Jurisdiction = Federal CollectionAuthority = ProceedsofCrime&TerroristFina
ncingAct CollectionReason = SuspiciousFinancialTransaction SecurityDesignation
 = ProtectedB DisclosureAuthority = PrivacyAct DispositionAuthority = National
ArchivesAct

Denied access to
Naming Authority: FederalFinancialReporting&AnalysisCentre Type = IntelligenceD
ata Jurisdiction = Federal CollectionAuthority = ProceedsofCrime&TerroristFina
ncingAct CollectionReason = SuspiciousFinancialTransaction SecurityDesignation
 = TopSecret DisclosureAuthority = PrivacyAct DispositionAuthority = NationalA
rchivesAct

Granted access to
Naming Authority: FederalPoliceAgency Type = InvestigativeData Jurisdiction =
Federal CollectionAuthority = FederalPoliceAct CollectionReason = Money-launde
ringInvestigation SecurityDesignation = ProtectedC DisclosureAuthority = Feder
alPrivacyAct DispositionAuthority = NationalArchivesAct

Granted access to
Naming Authority: FederalPoliceAgency Type = InvestigativeData Jurisdiction =
Federal CollectionAuthority = FederalPoliceAct CollectionReason = Money-launde
ringInvestigation SecurityDesignation = ProtectedB DisclosureAuthority = Feder
alPrivacyAct DispositionAuthority = NationalArchivesAct

Denied access to
Naming Authority: PrivateSectorFinancialInstitution Type = TransactionData Jur
isdiction = Federal CollectionAuthority = ProceedsofCrime&TerroristAct Collect
ionReason = StandardDepositTransaction SecurityDesignation = ClientConfidenti
al DisclosureAuthority = PIPEDA DispositionAuthority = PIPEDA

Granted access to
Naming Authority: PrivateSectorFinancialInstitution Type = TransactionData Jur
isdiction = Federal CollectionAuthority = ProceedsofCrime&TerroristAct Collect
ionReason = SuspiciousDepositTransaction SecurityDesignation = ClientConfidenti
al DisclosureAuthority = PIPEDA DispositionAuthority = PIPEDA
C:\iLock\demos\JII\CGI_GBAC_RADModel>
```

James Howard, a Police Investigator, requesting access to John Doe's information

All of the tools for creating and managing the access policy are provided by iLock. The iLock architecture supports dynamic policy updates that take effect immediately, so applications do not need to be restarted when access policies change. iLock also supports remote policy administration.



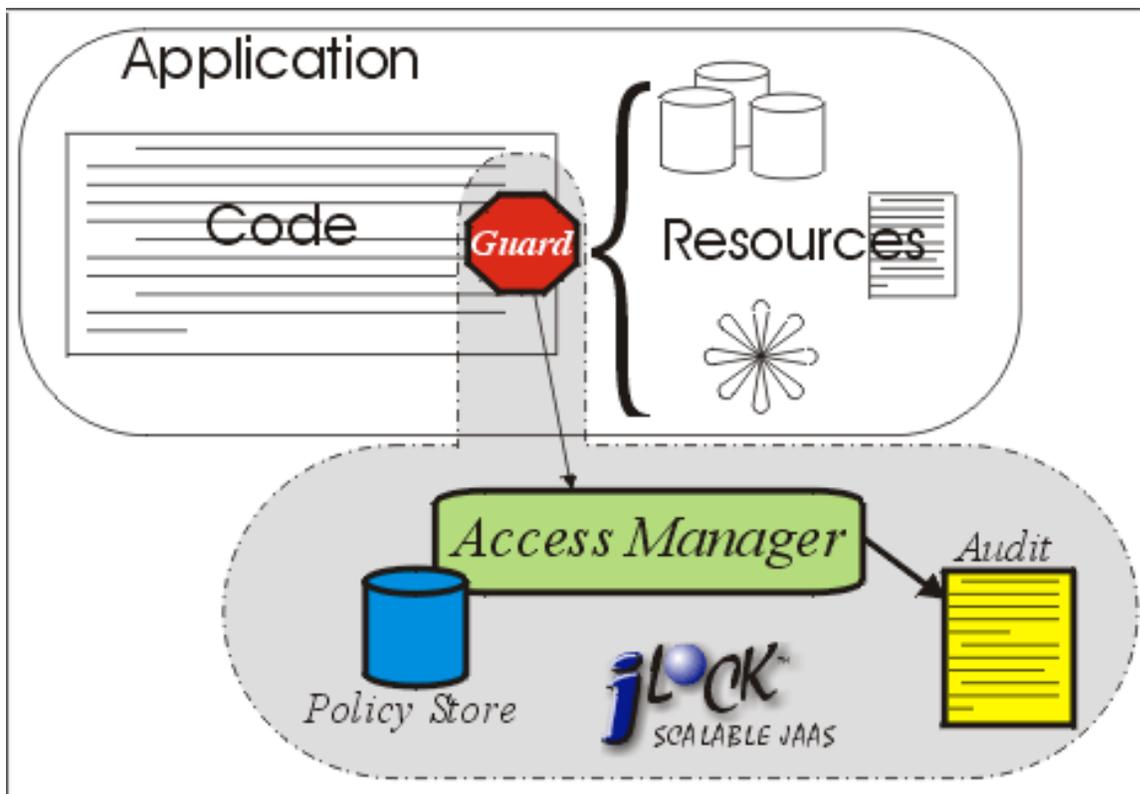
Summary

The trend towards a service-oriented architectural approach to dealing with application-level security is evident in recent analyst reports. For example:

META Group predicted in late 2003: "as businesses begin to put more focus on design for application securability and service oriented architecture, application-specific security mechanisms will migrate to infrastructure."

An OMG RAD-based model - supported in most of the iLock products - provides APIs that enable you to authenticate and easily integrate access control checks within your business applications. iLock supports a pluggable architecture that allows you to select your authentication technology based upon your requirements for authentication (userid password and/or certificates). It also allows you to leverage existing repositories you may have for users, groups and roles. Of course, you have the flexibility to support the sophisticated access policy required by legislation and your corporate policies for governance based access control.

Utilizing the RAD model, your business developers simply insert AccessManager calls (Software Guards) at the points in the software where sensitive resources are exposed. This Guard consults with the iLock Access Manager who evaluates the policy and advises the Guard on allowing access. The OMG RAD architecture enables many different policy models to be leveraged by a Java business application.



RAD supports a service-oriented architecture for authentication and authorization

The SOA model is supported in jLock, c/Lock, orbLock and webLock!



Challenge 2AB!

Are you still not sure if jLock can help with your GBAC requirements? Challenge us to prove it. Send us four or five examples of your access management requirements. We'll configure iLock with policies you can use and send you an evaluation copy of iLock complete with a working demo so you can see how to leverage iLock within your application. We'll even send you the source code for the demo so your development staff can take a look at exactly how little we had to do to insert a guard! Go ahead... challenge us. What have you got to lose – an increasingly difficult access management problem?

2AB, Inc.
1700 Highway 31
Calera, Alabama 35040
877.334.9572 (toll-free)
challenge@2ab.com