



# Harmony

## 2AB will include authorization services with orb2 Protection beyond CSiv2's authentication

### Special points of interest:

- orbLock bundled with orb2 in next Major Release
- Sneak Preview of interceptors in orb2 for C/C++
- Sneak Preview of idl2wsdl compiler and SOAPCORBA genbridge
- Check out the link to all our white-papers
- 2AB has a new office location!

2AB has announced plans to enhance the standard security features of orb2 by bundling orbLock, their CORBA authorization services with orb2 for Java version 5. orbLock (a member of the iLock Security Services product suite) has been offered as a value-added product for a number of years.

2AB has added new security features in each release as an indication of its commitment to provide trusted CORBA products. orb2 for Java version 4 implemented the features of the OMG Common Secure Interoperability version 2 (CSiv2) specification.

In Version 4.2, 2AB added the Identity Manager as an option for managing user information and providing CSiv2 compliant authentication services. The Identity Manager has allowed customers to easily incorporate user ID, password management and authentication services as integral features of their product offerings and business applications.

The CSiv2 standard provides a standard for authentication services but does not include the authorization services necessary to control who can access CORBA operations. The inclusion of orbLock's features in the standard orb2 of-



fering is the next step in ensuring that application-level security is easy for customers to implement in their applications.

The other iLock architected products (jLock: Scalable JAAS, webLock and c/Lock) will remain fully compatible and offer features and APIs tailored for providing robust standards-based application-level security in Web, Java and C/C++ environments. ■

### In this issue:

CORBA authorization Standard feature	1
Developer's Corner Using orbLock	1
Sneak Preview 1 orb2 for C/C++ interceptors	4
Sneak Preview 2 SOAPCORBABridge	8

## Developer's Corner Using orbLock for authorization

CSiv2 is the OMG and JCP standard for secure interoperability. It leverages TLS for server authentication and provides a choice of mechanisms for client authentication. In an earlier newsletter we described how to leverage orb2's Identity Manager and CSiv2 features for authentication (login) services. In this article, we will explore adding authorization checks to protect CORBA op-

erations. We will show how you can control the invocation of a CORBA operation based on the user that is logged in (authenticated). To do this, we will outline how to leverage orbLock's authorization services. We will also provide a look at the Security Center Service and Administrative tools that are part of orbLock today (and will be in orb2 in the next major release).



It is important to understand that orbLock supports both application-unaware and application-aware authorization services. With application-unaware services, there is no need for changes to your source code. You simply configure your servers and clients for authentication and authorization and the rest is handled for you. While this is a very

(See DevCorner on page 2)

(DevCorner Continued from page 1)  
 powerful way to add security to your application, there are times when you may wish to gain access programmatically to information about the authenticated user and/or control certain aspects of authorization. The CORBA Security Service (CSS) APIs are provided in support of application-aware security.

The first step is to define access policy for the CORBA operations (as defined in your IDL). We use the Security Center Administration tool (sc\_admin) to import the IDL for our demo application. We have two IDL files for this demo. They are named people.idl and hrfile.idl. Figures 1, 2 and 3 illustrate how to do import IDL. Note that if there are included idl files in another directory, you can specify that in the Include Path field. You may also provide definitions that you wish to be set when the IDL is parsed.

Next we can define access policy for the operations. As shown in Figure 4, select the operation you wish to protect and right-click. Select "Set Policy". The screen in Figure 5 will allow you to select access policy for who can execute that CORBA operation.

We defined a policy that says "Anybody" can execute the "is\_active" operation on the Employee interface. Although we will allow anyone to determine if an employee is active, we want to limit access to the employee's HR record. To do this, we create a policy that only allows

(Continued on page 3)

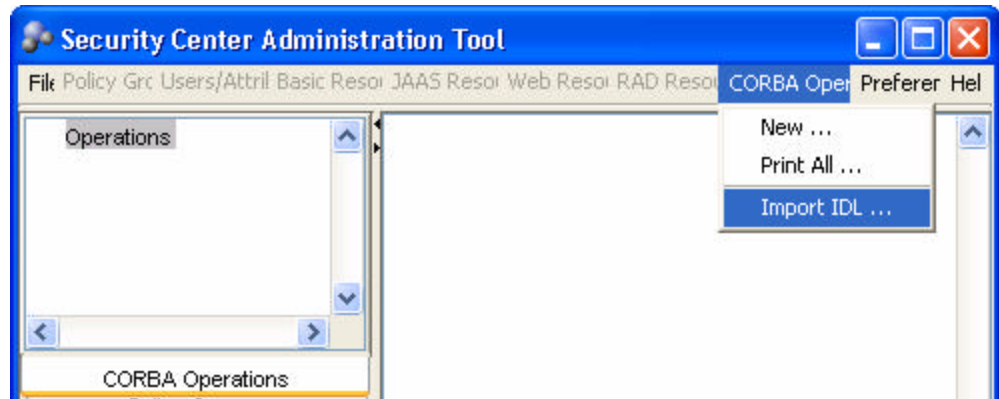


Figure 1: Select Import IDL from the CORBA Operations menu

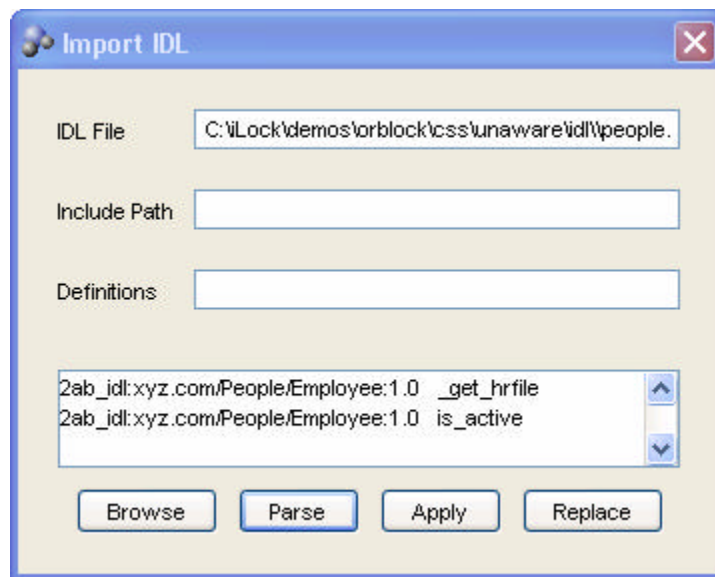


Figure 2: Specify the IDL file and Parse.

Then Apply.

You can use Browse to locate your IDL

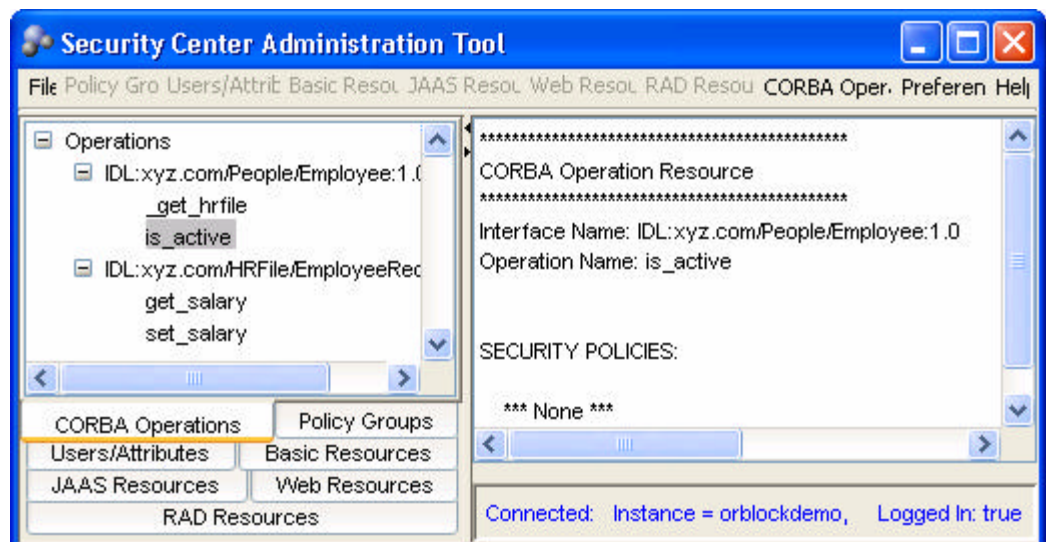


Figure 3: The imported IDL appears in the Security Center window

(Continued from page 2)

members of the group *hr* to invoke the `_get_hrfile` operation (Figure 6). Using this same method, we assign a policy to the `EmployeeRecord` `get_salary` that allows only members of the *hr* group and a policy that only allows people who have the role *hr:executive* to execute the `set_salary` operation.

Figure 7 shows the main screen view of the policy for the `set_salary` operation. You can view each policy in text form by clicking on the policy in this view.

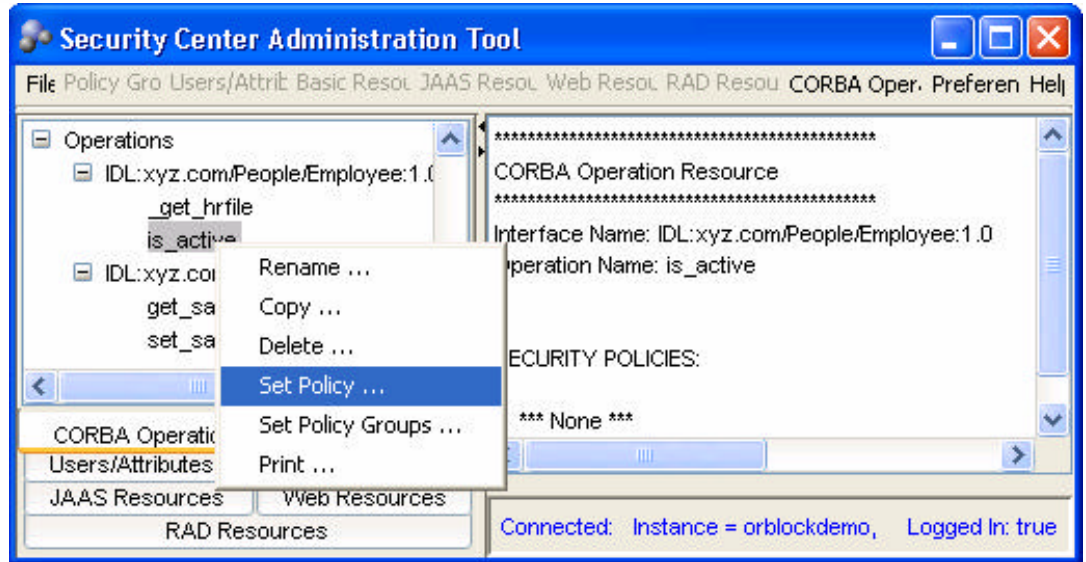


Figure 4: The menu to set policy for an IDL operation

Now that we have defined access policy for our CORBA operations, we are ready to configure our services to require authentication and authorization. We will do this without making any changes to the source code for the clients and servers. This is called application-unaware security.

orbLock provides CORBA interceptors that make the authorization checks. The pre-built interceptors are configured using an orb property. We define this property in the script that runs the demonstration servers and client. We must also specify the security center instance that was configured with the users and access policy. This is done in a property file that is read by the interceptor.

(Continued on page 5)

Figure 5: Setting an access policy that allows anyone to execute the `is_active` operation on `Employee`

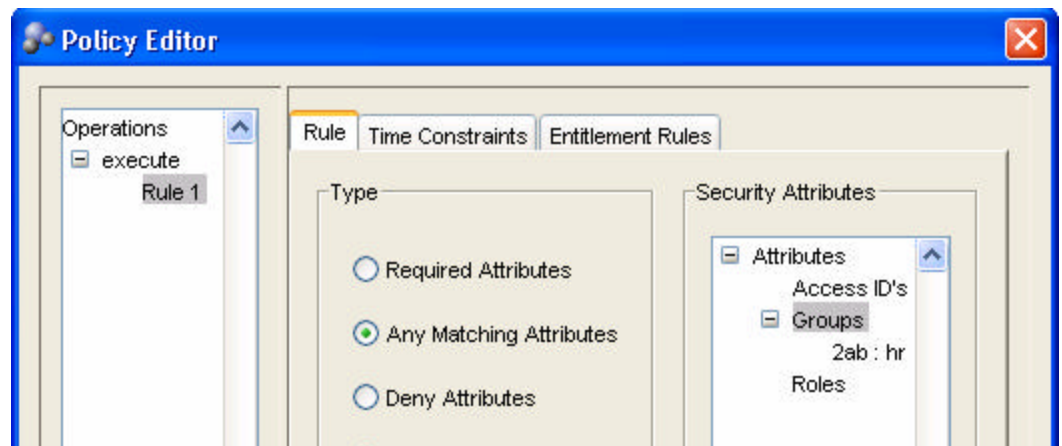
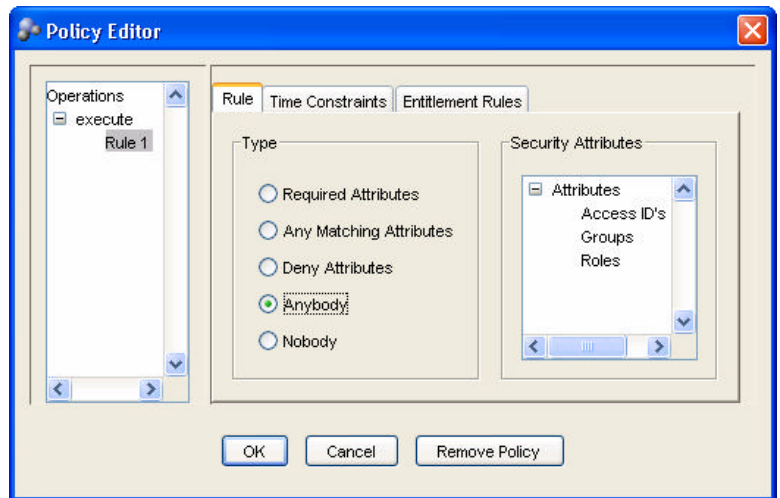


Figure 6: Create a policy that allows only the *hr* group to access the `hrfile` attribute on `Employee`

## Sneak Preview

### Interceptor support in orb2 for C/C++ v4



In this article, we'll take a look at the interceptor support 2AB is providing in orb2 for C/C++ version 4.0. Note that the current version of orb2 for Java already fully supports portable interceptors.

On the client side, there are three interception points supported. They are:

- Send request
- Receive reply
- Receive exception

On the server side, there are also three interception points. They are:

- Receive request
- Send reply
- Send exception

orb2 allows multiple interceptors to be registered and for the user to control the order of execution of these interceptors. The server.c and client.c code shows how to register a set of client and server interceptors.

The client\_intercept.c and server\_intercept.c code shows sample implementations of a client send request and a server receive request interceptor that have been added to a Calc (the standard orb2 demo) client and server.

With the full implementation of interceptors in the C/C++ orb, 2AB will be able to add the type of CORBA authentication and authorization services currently available in orb2 for Java. Changes to support later versions of GIOP and IOR interceptors are also scheduled for this release. ■

```
orb2_interceptor_add_server_request_interceptor (
    &Calc_server_receive_request_int1,
    &Calc_server_send_reply_int1,
    &Calc_server_send_exception_int1 );
...
DAIS_schedule();
```

#### server.c: Registering interceptors

#### client.c: Registering interceptors

```
/* Register the client-side interceptors. */
orb2_interceptor_add_client_request_interceptor (
    &Calc_client_send_request_int1,
    &Calc_client_receive_reply_int1,
    &Calc_client_receive_exception_int1 );
```

#### Client\_intercept.c Client send request interceptor

```
Calc_client_send_request_int1 ( CORBA_unsigned_long _request_id,
                                CORBA_Environment *_ev )
{
    IOP_ServiceContext *context = malloc(sizeof(IOP_ServiceContext));

    printf ("In Calc_client_send_request_int1 for request_id 0x%x\n",
            _request_id );

    context->context_id = 12345;
    context->context_data._maximum = 35;
    context->context_data._length = 35;
    context->context_data._buffer = CORBA_sequence_octet_allocbuf(35);
    context->context_data._release = CORBA_TRUE;

    strcpy( (char *)context->context_data._buffer,
            "All your bases are belong to us!");

    orb2_client_intcpt_add_request_service_context ( _request_id,
                                                    context,
                                                    CORBA_TRUE );
}
```

#### server\_intercept.c Server receive request interceptor

```
void
Calc_server_receive_request_int1 ( CORBA_unsigned_long _request_id,
                                    CORBA_Environment *_ev )
{
    IOP_ServiceContext *context = NULL;

    printf ("In Calc_server_receive_request_int1 for request_id 0x%x\n",
            _request_id );

    context = orb2_server_intcpt_get_request_service_context ( _request_id,
                                                                12345 );

    if ( context == NULL )
    {
        printf ("FAILED to get context with ID = 12345!\n");
    }
    else
    {
        printf ("Got context with ID = %d, len = %d, max = %d, buffer = %s\n",
                context->context_id,
                context->context_data._length,
                context->context_data._maximum,
                context->context_data._buffer );
    }
}
```

(Continued from page 3)

Figure 8 shows the script that is used to invoke the PeopleServer with security enabled. The changes that were made to this script have been bolded. Similar changes have been made to all the scripts.

When we run the scripts with security enabled, a user ID and password is requested (see figure 9). We have previously created two users using `sc_admin`. They are "mjones" and "jdoe." User "mjones" is in the group `hr` and has the role `hrexecutive`. User `jdoe` is in the group `employees` and has the role `infotech`. Because of the security policy we defined above, either user

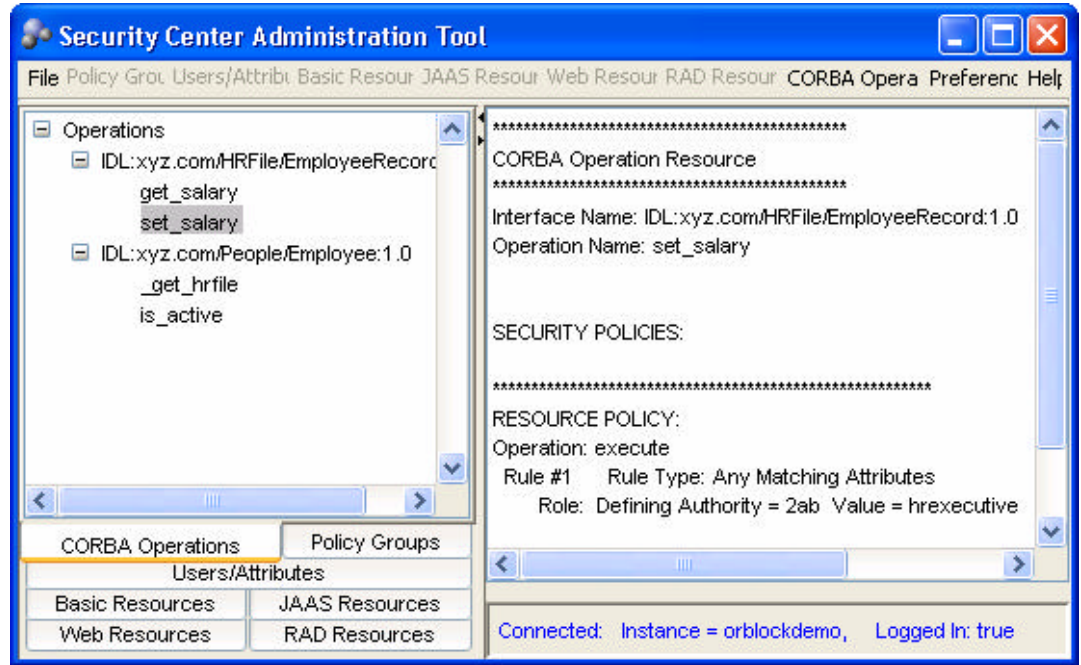


Figure 7: Viewing the policy for a CORBA operation.

```
set PRG_NAME=com.xyz.PeopleServer
set LOC_CLASSPATH=%ILOCK_HOME%\classes;%ILOCK_HOME%\lib\ilockca.jar
set PI=-Dorg.omg.PortableInterceptor.ORBInitializerClass=com.twoab.ilock.ol.pi_portable.Initializer=
peopleserver.properties
set DEFS1=-Dorg.omg.CORBA.ORBClass=com.twoab.orb2.core.ORBImpl
set DEFS2=-Dorg.omg.CORBA.ORBSingletonClass=com.twoab.orb2.core.ORBSingleton
"%JRE_HOME%\bin\java %PI% %DEFS1% %DEFS2% -classpath %ORB2%\lib\orb2.jar;%LOC_CLASSPATH% %
PRG_NAME% -ORBlicense_dir %ORB2%\lib -ORBconfig=%ORB2%\lib\orb.properties -ORBport 9801 %*
```

Figure 8: Bolded items show script modifications to add security to the PeopleServer

will be able to perform functions that require the `is_active` operation on `Employee`, but only `mjones` will be allowed to perform functions that retrieve the `EmployeeRecord` from the `Employee` object and/or access salary information via operations on `EmployeeRecord`. You may wish to review the policy to make sure you understand why that is true.

Figures 10 and 11 show the debug-level trace messages from our execution of the `PeopleServer` using the secure scripts and authenticating as user "jdoe." Figures 12, 13

and 14 show the same interaction when `mjones` is the user. Note that in this demonstration, no code has been changed; only the scripts that start the client and servers.

Although there is no requirement to write/modify any code to secure a CORBA-based service, there do exist programmatic interfaces that provide applications with greater flexibility in dealing with the security environment. These APIs allow you to programatically:

- Retrieve the security attributes (accessids, groups and roles) of the user
- Control authentication

- requirements for objects on a per-POA basis
- Control the object domain for an object
- Control the delegation of security attributes
- Control the authentication information passed to the authenticator
- Provide a custom Authenticator
- Provide a custom AttributeLookup class
- Provide a custom AttributeCache class

The most commonly used of these features is the ability to retrieve security attributes (accessid, groups and/or roles) assigned to the user. The ap-



Figure 9: orblock's Login Dialog Box

plication might need to write log records or gather statistics based on the current user. These attributes may also be used to make application-based access decisions. This is

(Continued on page 7)

Figure 10: client making requests on server with authorization enabled.

UserID authenticated is jdcoe.

```

iLockdemo
C:\iLock\demos\orblock\css\Newsletter>s_client
initializing the ORB
*** CSS Activating *** Version 4.4.2
reading stringified object reference from file
invoking method is_active on People::Employee object reference
[client - send_request for is_active]
[client - Retrieving authentication token. Operation = is_active]
[client - send_request for _get_hrfile]

org.omg.CORBA.NO_PERMISSION:  vmcid: 0x0  minor code: 101  completed: No
    
```

```

iLockdemo - s_peopleserver
[peopleserver - receive_request for is_active]
[peopleserver - Client ID Value: 1168893975925]
[peopleserver - AccessDecision::access_allowed]
[peopleserver - AccessDecision: true for IDL:xyz.com/People/Employee:1.0  is_act
ive]
Employee::is_active invoked
[peopleserver - receive_request_service_contexts for _get_hrfile]
[peopleserver - INFO: CssPolicy not set in IOR. Operation = _get_hrfile]
[peopleserver - receive_request for _get_hrfile]
[peopleserver - Client ID Value: 1168893975925]
[peopleserver - AccessDecision::access_allowed]
[peopleserver - AccessDecision: false for IDL:xyz.com/People/Employee:1.0  _get_
hrfile]
    
```

Figure 11: orblock performing authorization checks on operations requested of peopleserver.

jdcoe is denied \_get\_hrfile on employee.

Figure 12: client making requests on server with authorization enabled.

UserID authenticated is mjones.

```

iLockdemo
C:\iLock\demos\orblock\css\Newsletter>s_client
initializing the ORB
*** CSS Activating *** Version 4.4.2
reading stringified object reference from file
invoking method is_active on People::Employee object reference
[client - send_request for is_active]
[client - Retrieving authentication token. Operation = is_active]
[client - send_request for _get_hrfile]
invoked attribute hrfile on People::Employee object reference
[client - send_request for get_salary]
[client - Retrieving authentication token. Operation = get_salary]
invoked method get_salary on HRFILE::EmployeeRecord object reference. returned:
100000
[client - send_request for set_salary]
invoking method set_salary on HRFILE::EmployeeRecord object reference. set: 2000
00
    
```

```

iLockdemo - s_peopleserver
[peopleserver - receive_request for is_active]
[peopleserver - Client ID Value: 1168895189870]
[peopleserver - AccessDecision::access_allowed]
[peopleserver - AccessDecision: true for IDL:xyz.com/People/Employee:1.0  is_act
ive]
Employee::is_active invoked
[peopleserver - receive_request_service_contexts for _get_hrfile]
[peopleserver - INFO: CssPolicy not set in IOR. Operation = _get_hrfile]
[peopleserver - receive_request for _get_hrfile]
[peopleserver - Client ID Value: 1168895189870]
[peopleserver - AccessDecision::access_allowed]
[peopleserver - AccessDecision: true for IDL:xyz.com/People/Employee:1.0  _get_h
rfile]
Employee::hrfile invoked
    
```

Figure 11: orblock performing authorization checks on operations requested of peopleserver.

mjones is allowed \_get\_hrfile because she is in the group hr.

Figure 11: orblock performing authorization checks on operations requested of hrfileserver.

mjones is allowed get\_salary because she is in group hr, and set\_salary because she has the role hrexecutive.

```

iLockdemo - s_hrfileserver
[hrfileserver - receive_request for get_salary]
[hrfileserver - Client ID Value: 1168895189870]
[hrfileserver - AccessDecision::access_allowed]
[hrfileserver - send_request for get_policy_names]
[hrfileserver - Authentication not required. Operation = get_policy_names]
[hrfileserver - send_request for get_policy]
[hrfileserver - Authentication not required. Operation = get_policy]
[hrfileserver - AccessDecision: true for IDL:xyz.com/HRFile/EmployeeRecord:1.0
get_salary]
EmployeeRecord::get_salary invoked
[hrfileserver - receive_request_service_contexts for set_salary]
[hrfileserver - INFO: CssPolicy not set in IOR. Operation = set_salary]
[hrfileserver - receive_request for set_salary]
[hrfileserver - Client ID Value: 1168895189870]
[hrfileserver - AccessDecision::access_allowed]
[hrfileserver - send_request for get_policy_names]
[hrfileserver - Authentication not required. Operation = get_policy_names]
[hrfileserver - send_request for get_policy]
[hrfileserver - Authentication not required. Operation = get_policy]
[hrfileserver - AccessDecision: true for IDL:xyz.com/HRFile/EmployeeRecord:1.0
set_salary]
EmployeeRecord::set_salary invoked
    
```

(Continued from page 5)

often done using the OMG Resource Access Decision (RAD) Facility (orbLock also provides this service).

The security attributes of the current user can be obtained from the object named **com.twoab.CssSecurity.Current**. Figure 12 provides the code snippet for doing this.

Another useful feature of orbLock is the ability to programmatically control whether or not an object will require that its clients be authenticated. This is controlled by a POA-based policy called the **CssPolicy**. When this policy is applied to a POA, all objects it creates will enforce the policy for authentication. Figure 13 shows how to create a POA policy list.

Notice that the **CssPolicy** has two fields. The **authentication\_required** field is used to specify whether or not clients that access the objects created by this POA must be authenticated. The **domain\_name** field specifies the object domain that objects created by this POA will belong to.

This will give you a sample of how to leverage the APIs associated with orbLock's CORBA Security Service (CSS). To use the other features outlined above, please reference chapter 14 of the orbLock User Guide (version 4.4).

Like all Service-Oriented solutions, orbLock's Security Center allows remote administration and sharing between multiple applications and users. Applications with heterogeneous architectures may

```
com.twoab.CssSecurity.Current the_current = null;
    try {
        org.omg.CORBA.Object obj =
            the_orb.resolve_initial_references("CssCurrent");
        the_current = com.twoab.CssSecurity.CurrentHelper.narrow(obj);
    } catch (Exception e) {
        System.out.println("" + e.toString());
    }

// Now you can obtain your security attributes.

org.omg.Security.SecAttribute [] attrs = null;
    try {
        attrs = the_current.get_attributes();
    } catch (Exception e) {
        System.out.println("" + e.toString());
    }
```

**Figure 12: Obtaining security attributes of the authenticated user**

```
policy_list = new Policy[1];

Any policy_value = orb.create_any();
CssPolicy policy = new CssPolicy();
policy.authentication_required = true; // this requires authentication for every client
policy.domain_name = ""; // this says the object is not in an object domain

CssPolicyHelper.insert(policy_value, policy);

policy_list[0] = orb.create_policy(CSS_POLICY_ID.value, policy_value);
```

**Figure 13: Controlling authentication requirements on a per-POA basis**

also wish to leverage the APIs and features of jLock: Scalable JAAS and/or webLock while sharing the same Security Center repository for user identity and access policy. More information about these products and the features they support can be found on the 2AB website.

If you would like more information about orbLock (or any of the 2AB products), please contact 2AB via e-mail at [info@2ab.com](mailto:info@2ab.com).



**Download Whitepapers!**  
<http://www.2ab.com/whitepapers.htm>

In our newsletters, we offer a preview of different product features, but we just don't have the space to provide a detailed tutorial. In contrast, our documentation is intended as comprehensive reference material and can be overwhelming for a new user.

To provide more tutorial-based information, 2AB provides a variety of whitepapers that will help you understand our products by guiding you through a full demonstration. The newest whitepapers we have developed provide detailed information on using webLock's Filters and/or Security Realms in an Apache Tomcat5 environment. These new whitepapers are:

[Using webLock's Servlet Filters for Application-level Security](#) and

[Using webLock's Security Realm and Authorization Filter for Application-level Security](#)

## Sneak Preview

### idl2wsdl & wsdl2Java: a SOAPCORBABridge in orb2 for Java v5



2AB has implemented and demonstrated an idl2wsdl compiler that complies with the OMG standard. This compiler will be available in orb2 for Java version 5. As a value-add, we also plan to deliver a SOAP-to-CORBA bridge for ease in interoperating when using Apache

Axis. Axis is a core engine for the Web Services SOAP protocol. Our original development project to implement this functionality used TomCat4 and Axis1. When Axis2 was released by Apache, we expected that only minor changes would be necessary to our bridge generator. Unfortu-

nately for vendors like 2AB, Axis2 is a complete re-design and re-write with little thought toward backward compatibility.

We also discovered that the WSDL we generate according to the OMG IDL to WSDL standard is not always acceptable to Axis.

We are currently testing with TomCat5 and Axis2, however, the bridge generator has been significantly impacted by the changes in Axis from version 1 to version 2. A review of the sample code and configuration differences at: [http://ws.apache.org/axis2/1\\_0/migration.html](http://ws.apache.org/axis2/1_0/migration.html) will help you understand the frustration we have experienced during this transition.

While we have the bridge generator functionality working for many IDL data types, we are currently assessing the interest of our customers in an Axis-based solution to CORBA WebServices interoperability. If you have any feedback on this issue for the project team, please contact us at [info@2ab.com](mailto:info@2ab.com). Put SOAPCORBABridge in the subject line for immediate review.

#### Step 1: Compile your IDL to get CORBA stubs and skeletons

```
java -classpath %ORB2_CLASSPATH% com.twoab.orb2.compiler.IdlCompiler -native AbstractBase -all -d .\src Calc.idl
```

#### Step 2: Generate WSDL from the IDL

```
java -classpath %ORB2_CLASSPATH% com.twoab.orb2.compiler.Idl2Wsdll -d .\src Calc.idl
```

#### Step 3: Generate WebService stubs and skeletons from the WSDL

```
call %AXIS2_HOME%\bin\WSDL2Java --output .\ -d xmlbeans -p corba_bridge -s -g -ss -uri Calc.wsdll
```

```
call %AXIS2_HOME%\bin\WSDL2Java --output .\ -d xmlbeans -p corba_bridge -s -sd -uri Calc.wsdll
```

#### Step 4: Generate the CORBA WebService Bridge

```
java -cp ..\classes;\classes;.;%AXISCLASSPATH% com.twoab.bridge.CorbaBridgeGenerator2 -file Calc.bridge -ior server.ior -pkg corba_bridge -cpkg corba -o .\src\corba_bridge
```

#### Step 5: Compile stubs, skeletons for CORBA, WebServices, the implementations and the bridge.

#### Step 6: Create the Calc.aar jar for deployment (includes all classes and resource files)

### Steps necessary for using the SOAPCORBABridge in orb2 for Java v5

## Note new mailing address:

P.O. Box 335  
Helena, AL 35080

Phone: 205-621-7455  
Fax: 205-621-7455  
E-mail: [info@2ab.com](mailto:info@2ab.com)  
Support: [support@2ab.com](mailto:support@2ab.com)

*2AB is a provider of Trusted Solutions for Distributed Business. 2AB offers the middleware, identity and access management tools needed by application developers to integrate systems and protect sensitive information from unauthorized access. 2AB is a business partner of IBM, Intel, HP, Sun, Microsoft and Stratus. The orb2 and iLock product suites are available for Java, C and C++ on AIX, HP-UX (PA-RISC and ia64), OpenVMS (ALPHA and ia64), Linux, Solaris, VOS, and Windows. Founded in 1997, 2AB is privately held. For more information, please see <http://www.2ab.com>.*