Cppm nested groups

This article will explain how to search within nested groups to determine whether a user is actually a member of a higher level group. This information can then be used to make policy decisions within ClearPass Policy Manager (CPPM).

For example, in my lab, I have three groups in the following configuration:

Group1

Group2

Group3

My user is member of Group3, but I want CPPM to check my membership against Group1. In the real world, you may run into something like:

All Students

ReaganHS

Freshman

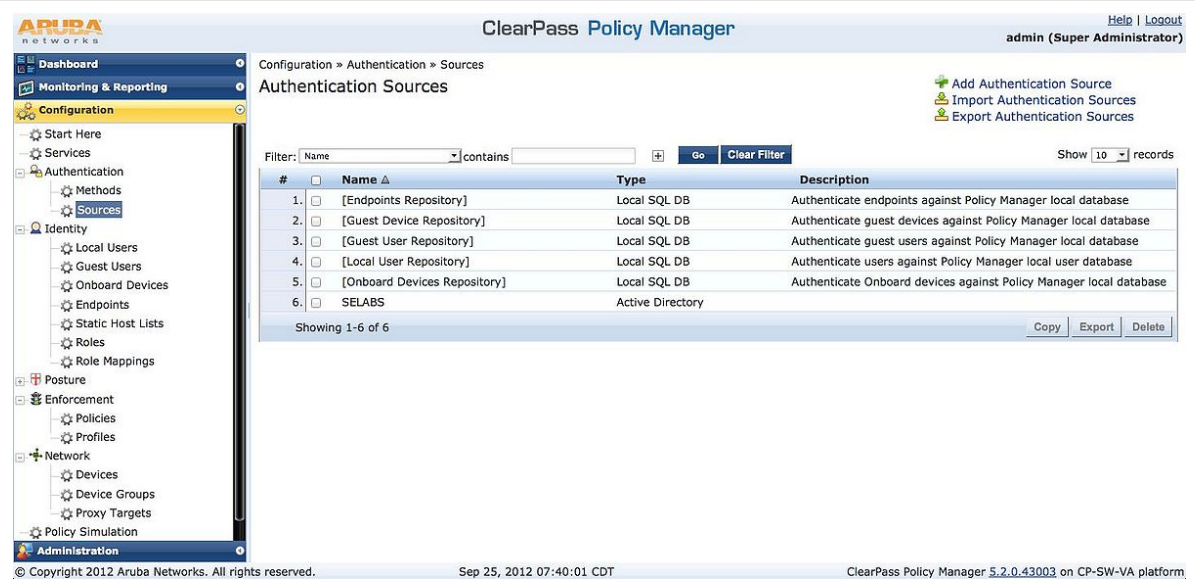
The administrator may want to know if a student is a member of ReaganHS or All Students. Out of the box, CPPM will report that the user is “Freshman”, but not the higher level groups.

The default logic of CPPM is to return only the “memberOf” for the user record. Below, we will configure CPPM to use memberOf to again query AD and return the group membership of the group itself. We will then query AD again for the group membership of the second level group. Along the way, we store each set of group memberships in a different variable so that we can use them for policy decisions. In the example above, CPPM will know that Student1 is a member of Freshman and will query AD again to get the group membership of Freshman, which will return ReaganHS. CPPM will then query AD again to get the group membership of ReaganHS, which will return All Students. CPPM will store Freshman, ReaganHS and All Students as different variables.

To start, you must already have joined CPPM to your domain and added a domain controller to CPPM as an authentication source. The process for doing these steps is documented in other places and outside the scope of this document.

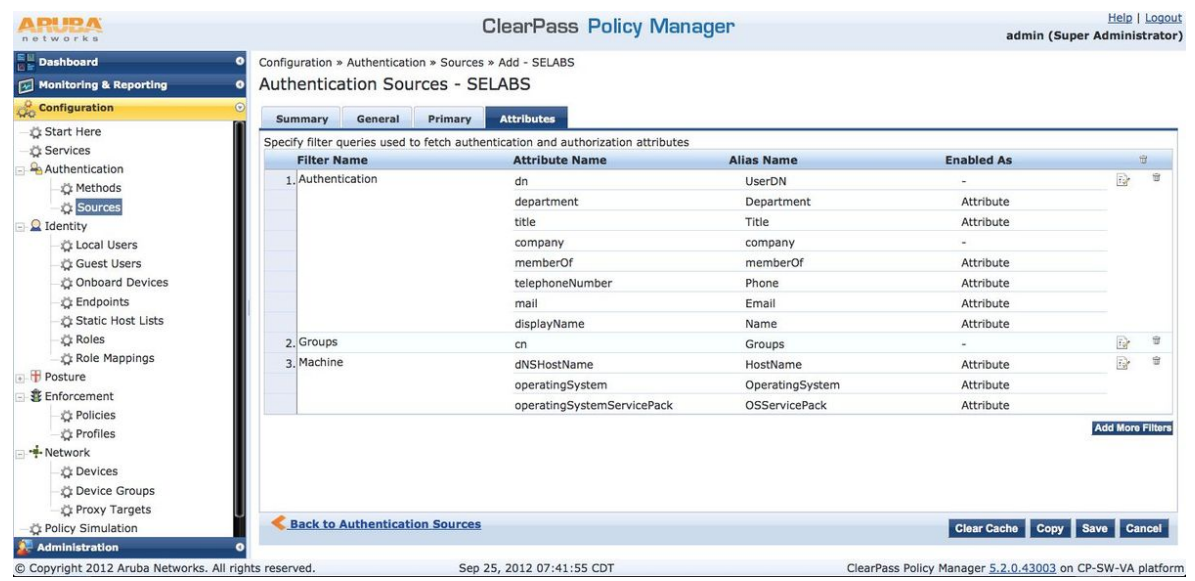
Once you have successfully joined Active Directory (AD) and added a domain controller as an authentication source, you are ready to look for nested groups.

To start, click on the name of your AD server (SELABS in the example below)

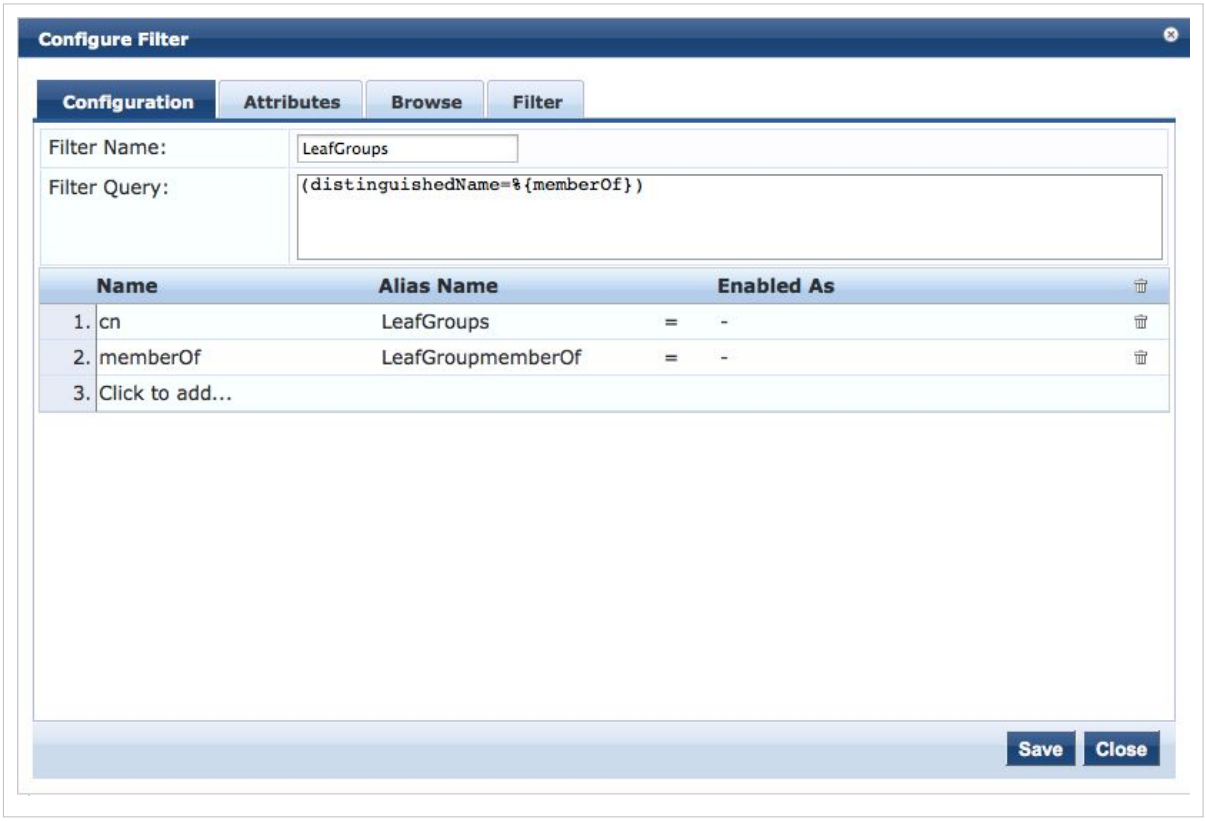


Sources.jpg

Click on the “Attributes” tab. Your source should already have several filters configured (Authentication, Groups, and Machine) as shown below.



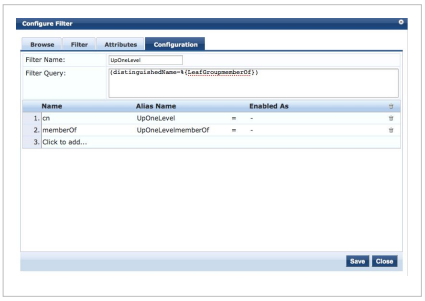
First, edit the filter named “Groups” by clicking on it. Change the Filter Name to “LeafGroups” and the Alias Name to “LeafGroups”. Add another Attribute by clicking on the “Click to add….” link below the existing Attribute. The name must be “memberOf” (it IS case sensitive) and the Alias Name should be “LeafGroupmemberOf”. The screen should now look like the example below.



The Alias Name LeafGroupmemberOf will be referenced later, so remember it.

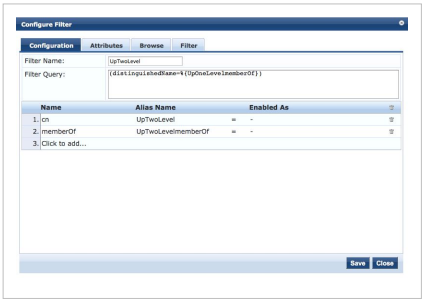
Make sure you click “Save”.

Now, add another filter by clicking the “Add More Filters” button on the bottom right corner of the window. Click the “Configuration” tab on the next window and enter “UpOneLevel” as the Filter Name. In the Filter Query box, enter (without the quotes) “(distinguishedName=%{LeafGroupmemberOf})”. This tells the filter to search for the variable called LeafGroupmemberOf, which was set in the initial query of the user record. Click the “Click to add…” link and enter “cn” as the name (again, it is case sensitive) and “UpOneLevel” as the Alias Name. Click the “Click to add…” link again and enter “memberOf” as the name and “UpOneLevelmemberOf” as the Alias Name. The filter should look like the screen shot below.



Click “Save” to save your progress.

Add another filter using the same process above, but make it look like the screen shot below.

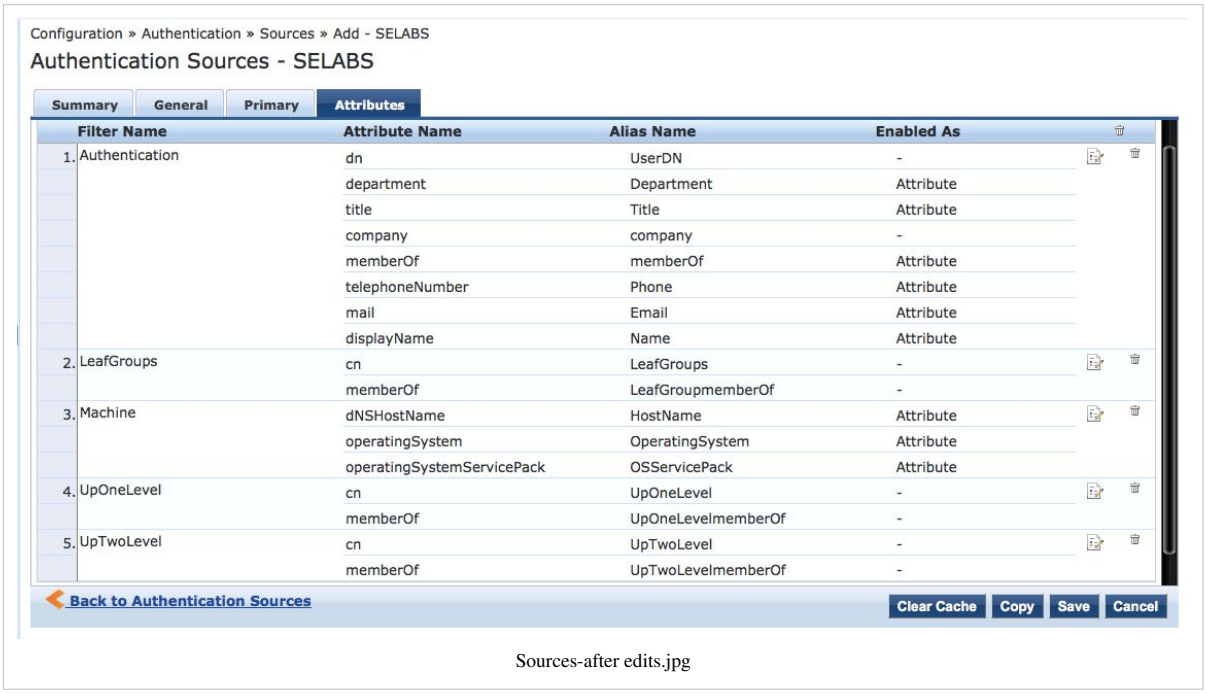


Notice that the Filter Query is now looking for “UpOneLevelmemberOf”. The Filter Name and Alias Names have changed as well.

Click “Save”.

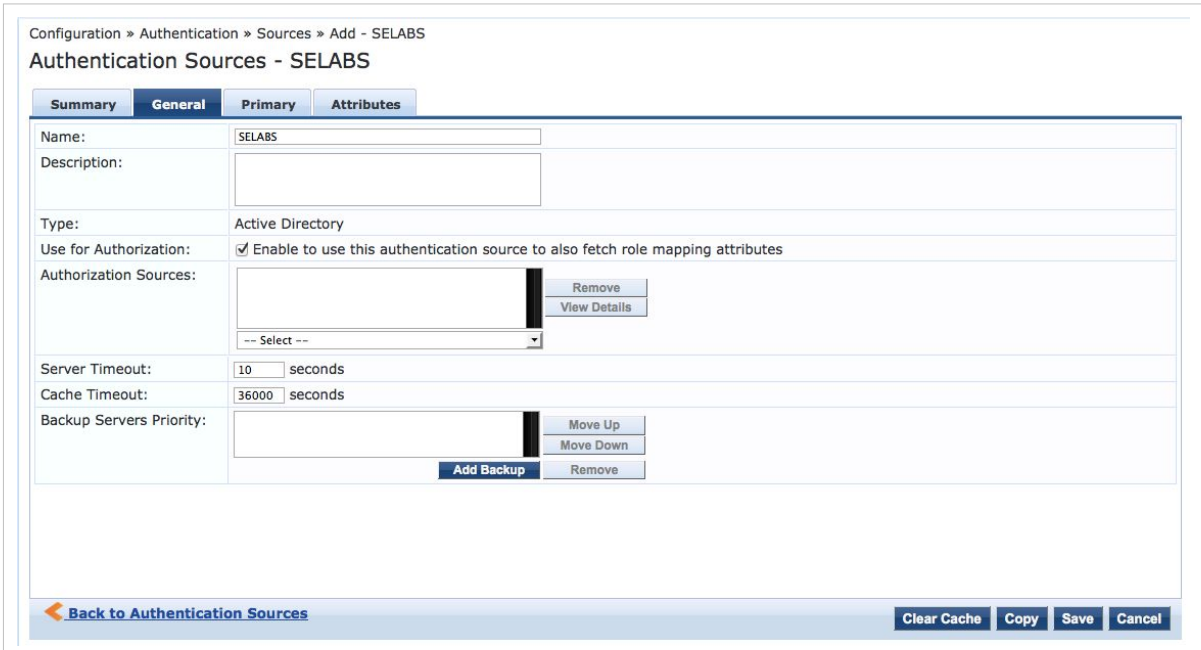
If you have more than three levels of nested groups, keep adding levels. For my demo, I only have three.

The final attributes screen should look similar to the one below.



Sources-after edits.jpg

You must configure your authentication source for authorization before CPPM will report authorization attributes. This is done by clicking on “Authentication”, “Sources”, then the “General” tab. Make sure you have checked the “Use for Authorization:” box as shown below.

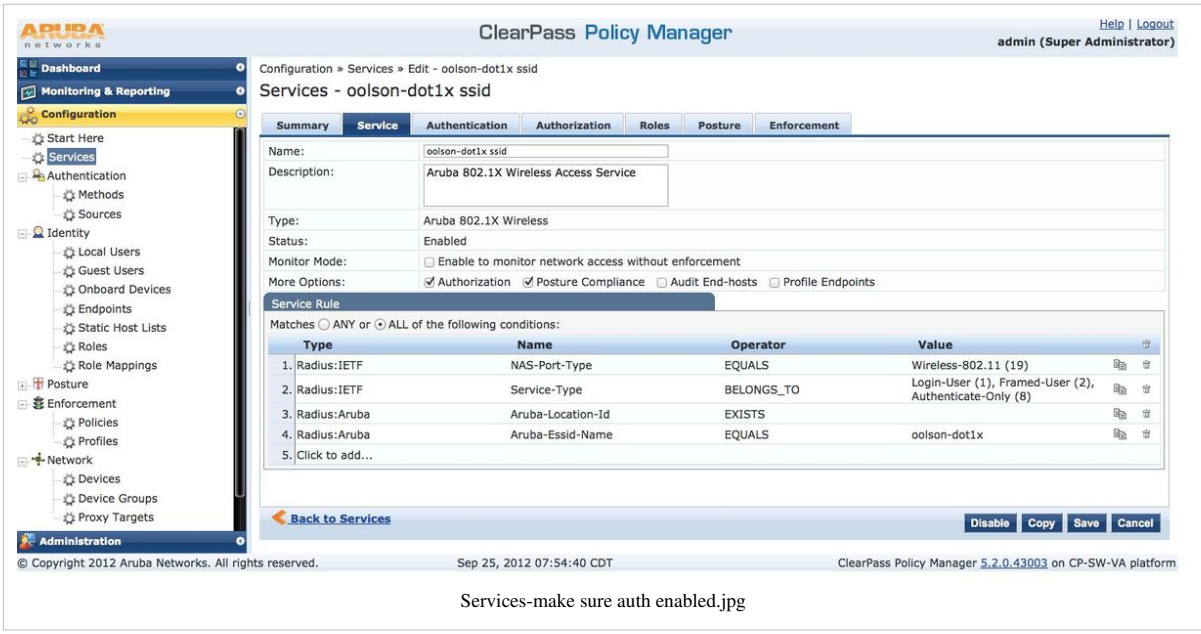


Sources with authorization.jpg

CPPM also requires that you add Role Mapping to your Service before the attributes will be recorded.

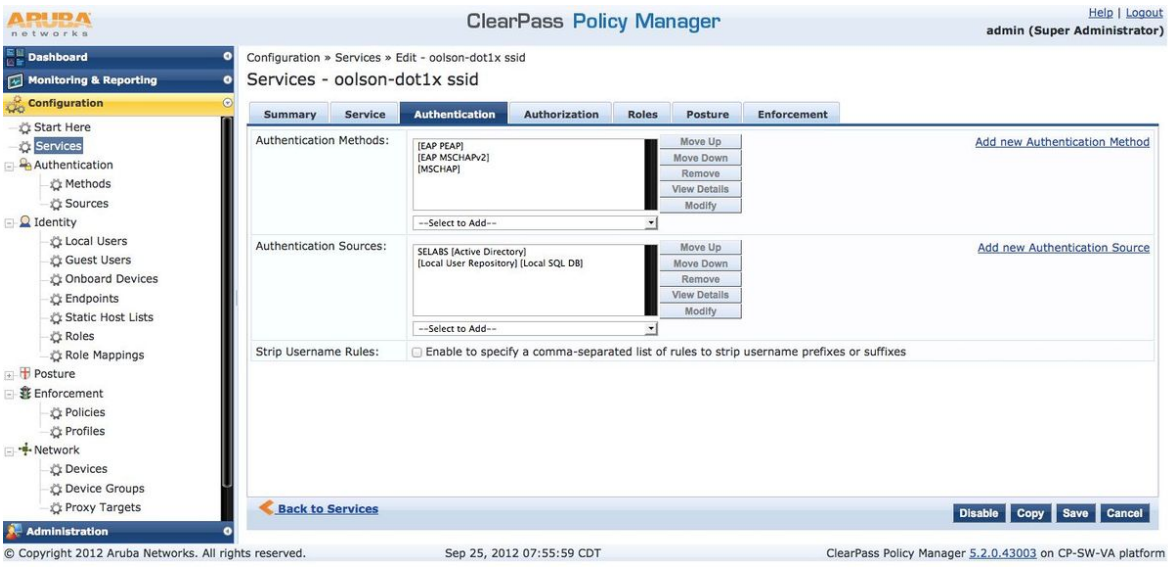
To add Role Mapping, click on Services and edit the Service that uses the Authentication Source you modified above. In my lab, it is called “oolson-dot1 ssid”.

Click the Service tab and ensure Authorization is enabled in the More Options box.



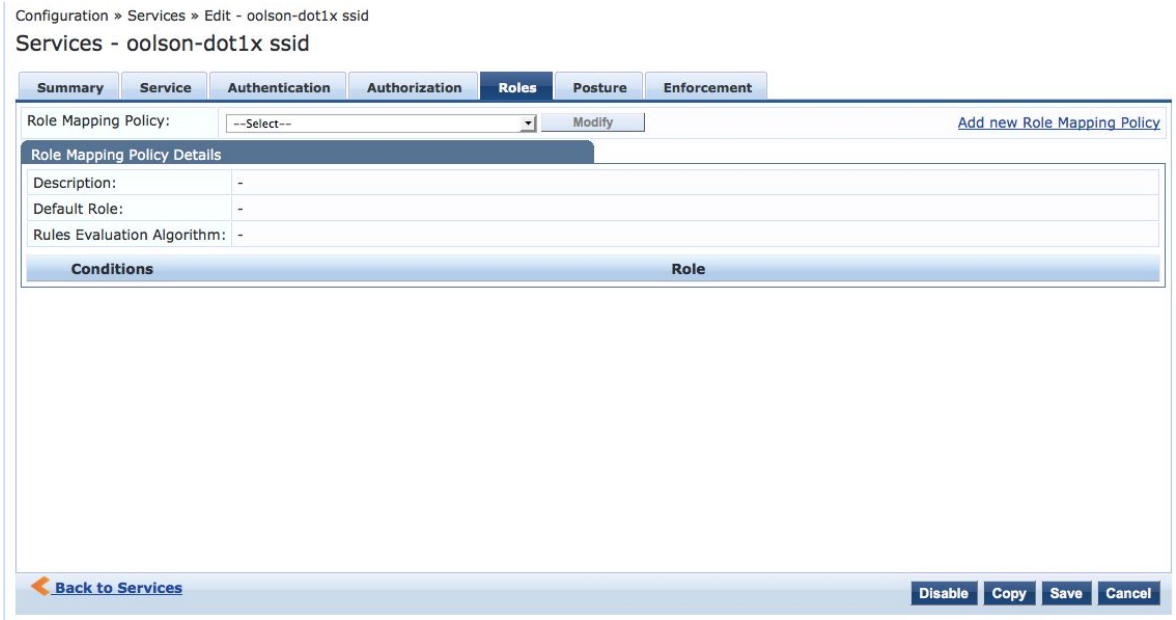
Services-make sure auth enabled.jpg

Click the Authentication tab and ensure your modified AD server is listed first in the Authentication Sources box.



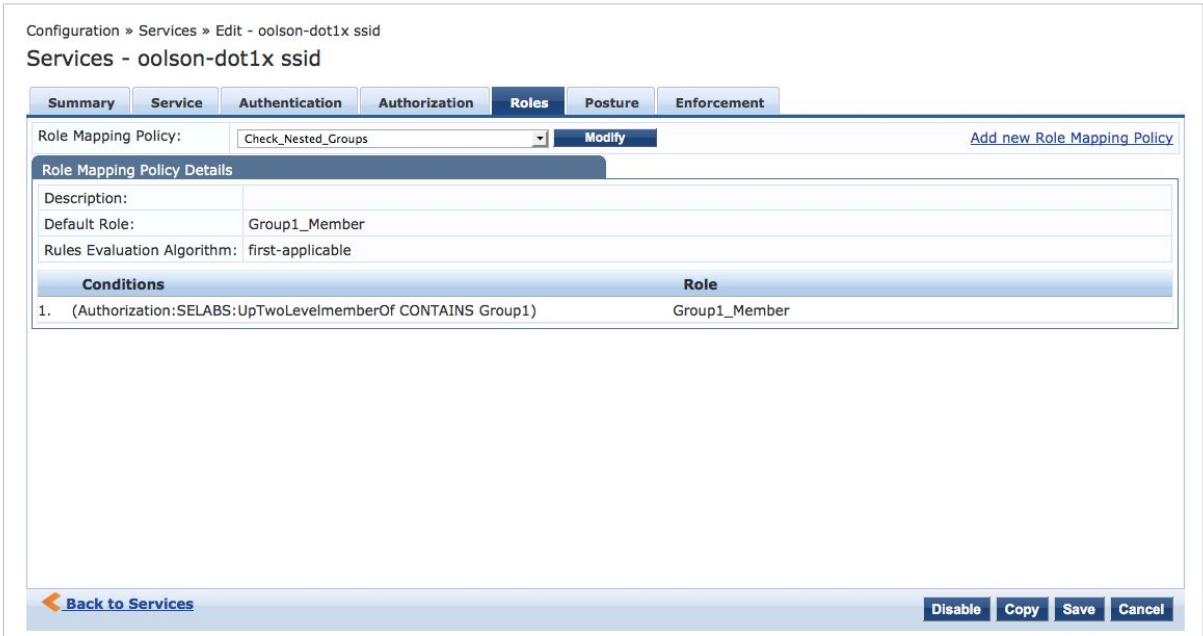
Services-make sure authentication is correct.jpg

To add a Role Mapping, click the Roles tab. Click the “Add new Role Mapping Policy” link.



Services-add new role mapping.jpg

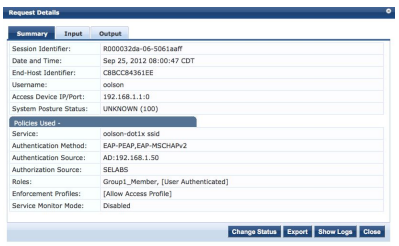
Add a Role Mapping Policy called “Check_Nested_Groups”. I added a new TIPs role called “Group1_Member”, but you can add what ever TIPs roles that make sense to you. The role checks “Authorization:SELABS:UpTwoLevelmemberOf” for “Group1” and assigns Group1_Member TIPs role if it matches.



Services-after add new role mapping.jpg

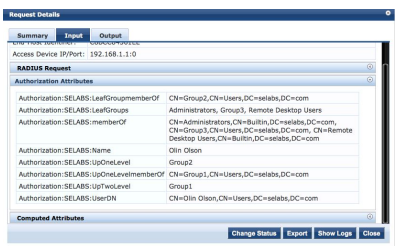
To verify its working, create an AD account and several groups. Make Group3 a member of Group2 and Group2 a member of Group1. Then, make your AD account a member of Group3.

Once you have the configuration above (both CPPM and AD) and you can successfully authenticate against AD via CPPM, you will be able to see the following logs in the CPPM Access Tracker.



AccessTracker-summary.jpg

Notice that the Roles: in the output above include “Group1_Member”. Group1_Member was assigned since my account is a member of Group3, which is a member of Group2, which is a member of Group1.



AccessTracker-input.jpg

In the output above, you can see that there are several Authorization attributes, including Authorization:SELABS:UpTwoLevel = “Group1”.