AMX Tutorial Dynamic Groups1

This tutorial is for IT staff who are experienced in identity management, it requires insight into how the ActiveDirectory works, and a working knowledge of Windows. This tutorial can be done independently of any other.

Dynamic Groups and Roles are very similar in AMX, Dynamic Groups are simpler and easier to setup. In AMX a Dynamic Group is one which a user is added to based on the value of an attribute. For example user accounts may be added to a group called "London Employees" when the value of their Account Attribute "location" is "London". There may be other Dynamic Groups such as "Edinburgh", "Glasgow", "Leeds" etc with a default if a User's Location is none of these. The default may be blank.

This exercise will demonstrate some of the more advanced features of AMX using the Active Directory, specifically:

- Using identityReport to create a file of accounts from the ActiveDirectory which will be used as a source of identities.
- Updating the source of identities to add an attribute containing a list of groups to manage in the ActiveDirectory.
- Running identitySync to create its transaction file, and reviewing the changes it would make to the ActiveDirectory.
- Using a lookup table to create a list of groups based on a person's attribute such as location
- Reconfiguring and running identitySync to use the lookup table, and reviewing the changes it would make to the ActiveDirectory.

No changes are made to the Active Directory, the tutorial can be run using a non-privileged account.

AMX runs on Windows and must be setup as shown in the AMX Tutorial Setup document. In this tutorial identityReport and identitySync are run from the Command Line using AMXRun which sets the environment variables.

Managed Groups

AMX maintains a list of Managed Groups, and User Accounts are only added and removed from these groups, all other groups that a person is a member of are ignored. In this case the Managed Groups are generated for the Active Directory from the Identity Source. For more details about defining Managed Groups see the Reference document.

Role Hysteresis

Hysteresis is intended to be used when individuals change roles. identitySync immediately adds a person to a group, but can delay the removal allowing the person to perform either roles for a configurable length of time. Hysteresis allows flexibility in the timing of the change of role. In this exercise no delay has been configured. For more details about Role Hysteresis see the Reference document.

1. Create an Identity Source

identityReport will read the ActiveDirectory and produce a file of accounts and their attributes. This involves updating the properties for identityReport, the ActiveDirectory schema and password and then running identityReport to create a file which will be used as an Identity Source.

Update identityReport Properties

Edit the identityReport Properties file ActiveDirectory2.properties, it is in the <installDirectory>\Tutorial1 directory

The parameters that need to be updated may have been already changed in Tutorial AD1, they are:

```
// Systems
ActiveDirectoryResource1 = dc1.corp.example.com
ActiveDirectoryAccountContainer1= OU=accounts,DC=corp,DC=example,DC=com
ActiveDirectoryUser1 =
ActiveDirectoryFilterAttribute1 = distinguishedName
ActiveDirectoryFilterValue1 = OU=EDN:ou=lon
```

- ActiveDirectoryResource1 can be the resolvable DNS name of the target Domain Controller
- ActiveDirectoryAccountContainer1is domain specific. For better performance during the tutorial specify the container where the accounts of interest are located. Use the MMC Console Users and Computers to identify the correct container. In production the account container must have all the accounts otherwise the check for account uniqueness will fail.
- ActiveDirectoryUser1 can be left blank in situations where a domain account is used on the PC running AMX. The account does not need to be an administrator to read the Active Directory accounts. If an alternative account is needed, create an ActiveDirectoryPasswd1.txt file adding the password in the first line, it will be encrypted when identityReport.exe first runs.

  ```
  ActiveDirectoryPasswd1 = ActiveDirectoryPasswd1.txt
  ```

- ActiveDirectoryFilterAttribute1 is the Active Directory attribute that is used to select accounts that are to be extracted, ActiveDirectoryFilterValue1 is the value that the filter attribute must contain to be included. Any Metaverse Attribute that is defined in the ActiveDirectorySchema.txt file can be used as a filter.

In this example the distinguishedName attribute is used and the ActiveDirectoryFilterValue1 defines values that the attribute must contain to be included in the identityReport report. The value is not case sensitive, and the ":"character allows multiple filter values to be entered. For example:

```
OU=EDN:ou=lon
```

The combination of AccountContainer and filtering define which accounts will be managed by identitySync. For security purposes you may want to use test accounts. In this case an alternative ActiveDirectoryFilterAttribute could be description, with an ActiveDirectoryFilterValue1 of "test". The ActiveDirectoryFilterAttribute must be defined in the Active Directory Schema file ActiveDirectorySchema.txt, see the next section for details of how to add attributes to the schema.

Update ActiveDirectory Schema File

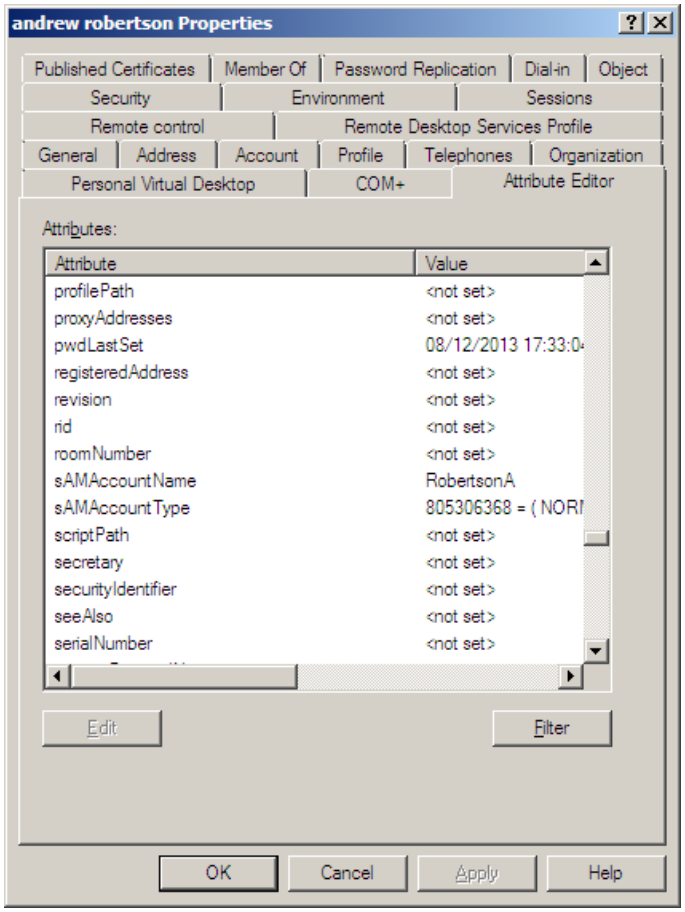
Update the file ActiveDirectorySchema1.txt, uncomment

```
memberOf,memberOf;delta
```

This file does not need any other modifications unless a FilterAttribute is used that is not defined in the schema file. The format of the schema file is:

```
Staging Attribute,Metaverse Attribute;Attribute Flags and Modifiers
```

Staging is the attribute name in the ActiveDirectory, this can be checked in the MMC Console Users and Computers, using the attribute editor.



Note that the sAMAccountName is the staging name for the user logon name.

The Metaverse Attribute names are used by AMX to build its internal database.

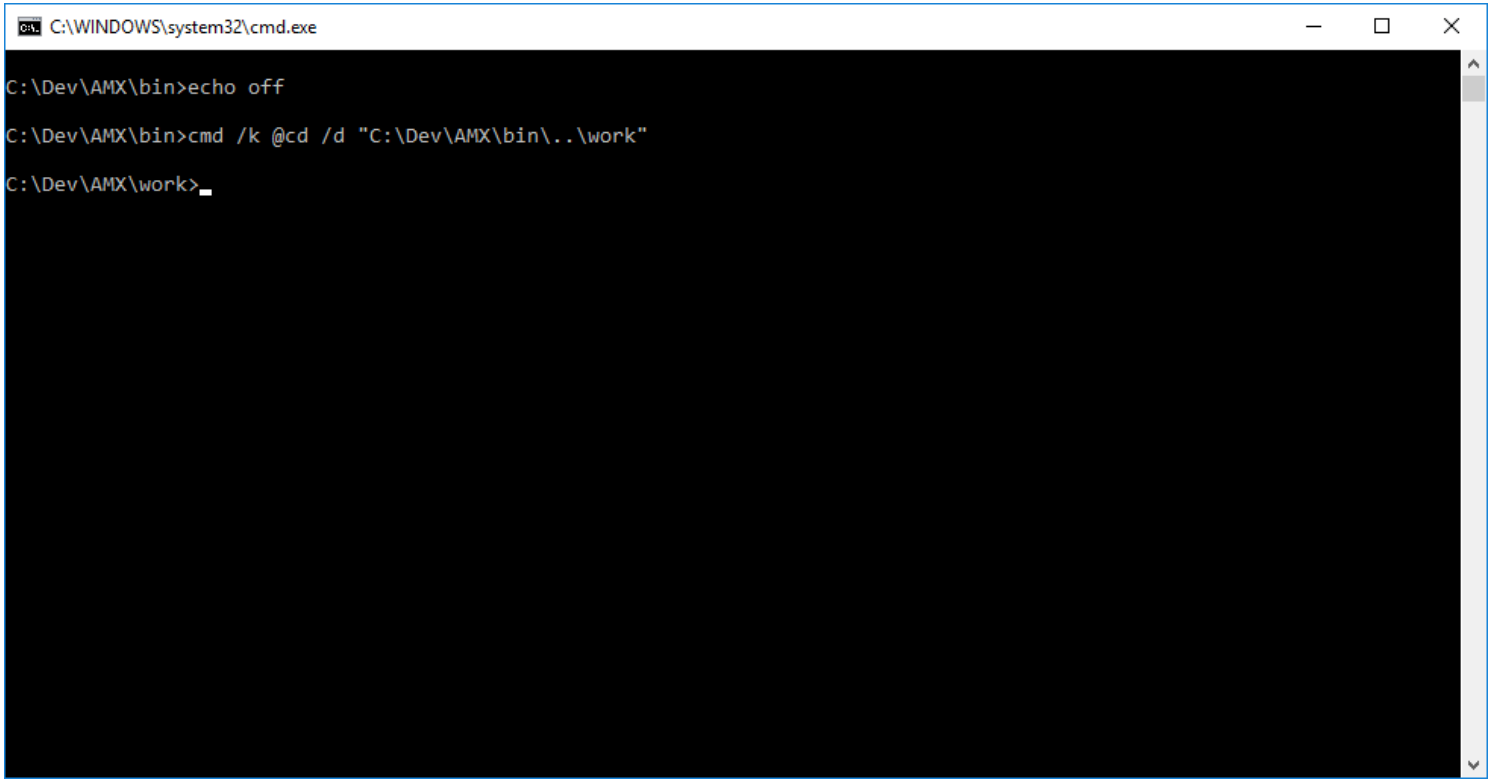
To add additional attributes, they must be defined in the ActiveDirectorySchema, and the Metaverse Attribute names must match the list of report header attributes defined in the accountAttributes property in the ActiveDirectory2.properties file.

Update ActiveDirectory Passwd

When ActiveDirectoryUser1 is defined, enter the password in the first line of the file ActiveDirectoryPasswd1.txt, when a domain account is used, the password is not required. The password will be encrypted when identityReport first runs and the clear text password will be removed.

Run identityReport

Right click on AMX Run in the Start Programs menu or AMXRun.bat in the installation bin directory, and Run as Administrator. For example:



This will open a Command Prompt.

Change directory to Tutorial1 and run identityReport.exe

```
C:\AMX\Tutorial1>identityReport.exe ActiveDirectory1.properties
Begins Mon, 31 Oct 2016 12:24:30 GMT
ActiveDirectoryIdentity1 corpID
Extracted 102 Identities
ActiveDirectoryIdentity Finished Mon, 31 Oct 2016 12:24:32 GMT
Total of 102 Identities

Finished Mon, 31 Oct 2016 12:24:32 GMT

C:\AMX\Tutorial1>
```

Check that the expected number of accounts were returned in the report IdentityReportAD1.csv. In situations where the number of accounts is incorrect, open the debug file and check for errors.

Failed to connect to the Active Directory

The console may show a message such as

```
Error: ActiveDirectory Extract A referral was returned from the server.
 for LDAP:// dc1.example.com/DC=corpz,DC=example,DC=com
Error: Exception of type 'AMX.ActiveDirectory+CustomException' was thrown.
```

This is caused when the domain controller does not recognize the domain.

Bad credentials

```
Error: ActiveDirectory Extract The username or password is incorrect.
 for LDAP://192.168.121.61/DC=corp,DC=example,DC=com
Error: Exception of type 'AMX.ActiveDirectory+CustomException' was thrown.
```

ActiveDirectory Extract Filter Attribute company not in Metaverse

A filter attribute defined in the ActiveDirectory2.property file ActiveDirectoryFilterAttribute1 is not defined in the right hand side of the ActiveDirectorySchema. See Update ActiveDirectory Schema

Filter failed to match any accounts

Search for "Skipped" in the debug.txt file

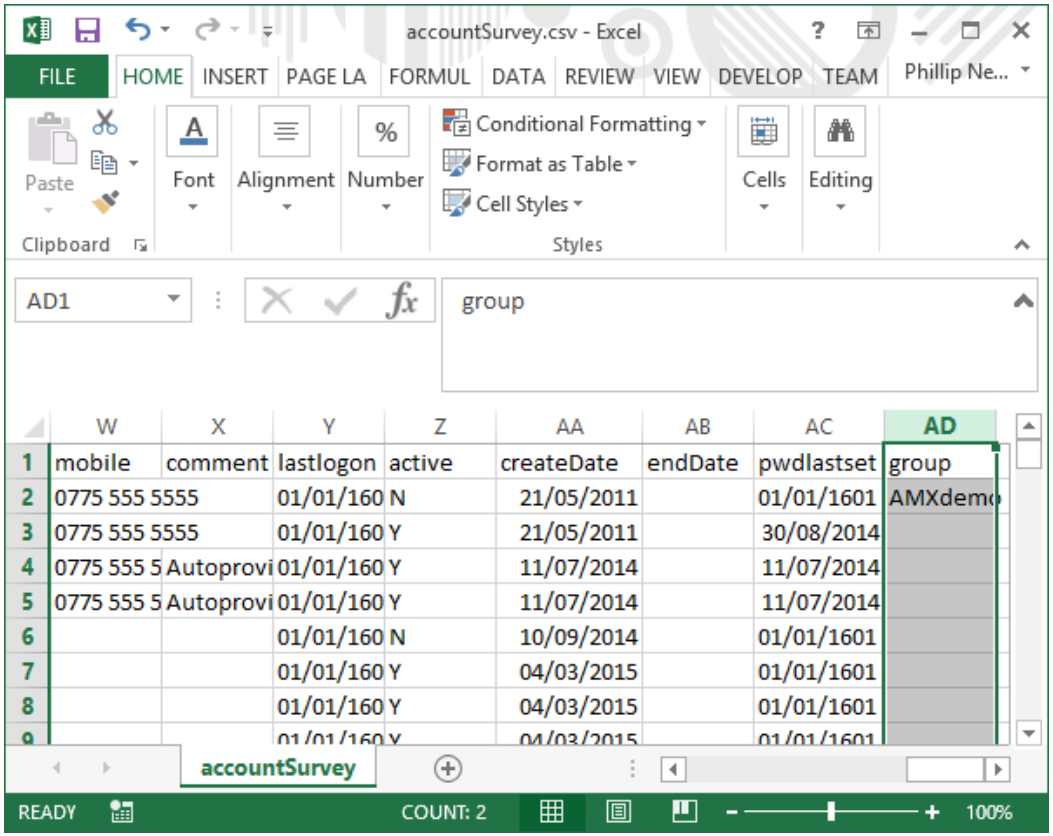
```
ActiveDirectory ExtractAttribute filter distinguishedName value = cn=alpin
thomson,ou=accounts,dc=corp,dc=example,dc=com filter ou=accountsz

ActiveDirectory ExtractAttribute Skipped ..... Thomson, Alpin
```

In this example the filter is miss-spelled ou=accountsz and is not found in the filterAttribute distinguishedName, so the record is skipped.

2. Defining Groups in the Identity Source

identityReport creates a file called IdentityReportAD1.csv and this will be used as a source of identities. It does not contain any groups, we will create a new attribute and add a dummy group to a single account, and since the ActiveDirectory will not be updated the dummy group does not need to be a real group in the ActiveDirectory.

Open IdentityReportAD1.csv in Excel or equivalent and add a header called "group" and for the first Identity add a dummy group called AMXdemo.



Replace the file as a CSV and exit Excel.

Update identitySync Properties

We will use identitySync to show the changes that would be made to the ActiveDirectory. Its properties need to be updated to suit your environment. Edit the ActiveDirectory2.properties file, it is in the <installDirectory>\Tutorial1 directory, identitySync needs some additional parameters that are not used by identityReport. The NETBIOS domain name:

```
ActiveDirectoryName1 = corp
```

The ActiveDirectoryLoadMode controls the behaviour of the identitySync Load process (Create, Update, Disable, Delete) CRUDD. U is update only. Note that identitySync never deletes anything, a delete is a move to a recycle bin in the ActiveDirectory.

```
ActiveDirectoryLoadMode1 = U
```

Update the CSV Schema

The new attribute in the IdCSVSchemaAD1.csv file needs to be added to the CSV Schema so that it can be read. Open the CSVSchema.txt file with a text editor and remove or comment out:

```
,memberOf
```

Add:

```
group,memberOf
```

This will make identitySync synchronise the memberOf Active Directory Attribute.

Update the Active Directory Schema

Open the ActiveDirectorySchema.txt file with a text editor and remove the comment for:

```
,memberOf
```

If this is not done the transaction file will show no changes.

Run identitySync

Run identitySync and check the results. The group AMXDemo does not exist and will not be created in this tutorial. No changes will be made to the Active Directory. For example:

```
C:\AMX\Tutorial1 >identitySync.exe ActiveDirectory2.properties
Error: must be run as an admin
Begins Tue, 10 Sep 2014 17:30:28 GMT  analyze
```

```
CSVIdentity 1 C:\AMX\Tutorial1\IdentityReportAD1.csv
Last updated 10/09/2014 17:22:05
Extracted 102 Identities
CSVIdentity Finished Tue, 10 Sep 2014 17:30:28 GMT
Total of 102 Identities

ActiveDirectory 1 dc.corp.example.com
Extracted 102 Accounts
Account joins    102
Account creates  0
Account updates  1
Account disables 0
Account deletes  0
ActiveDirectory Finished Tue, 10 Sep 2014 17:30:28 GMT
Analysis Ends Tue, 10 Sep 2014 17:30:28 GMT
Ends Tue, 10 Sep 2014 17:30:28 GMT

C:\AMX\Tutorial1>
```

The Transaction file ActionFile.txt in the Tutorial1 directory will contain something similar to:

```
10/09/2014|corp;CN=Alison
Craig,OU=LON,OU=accounts,DC=corp,DC=example,DC=com|Update|memberDel=AMXdemo|memberAdd=AMXdemo
Ends|28YMbL2OXo34K2SeJ1uVnw==
```

The 4th field is the undo value and the 5th field is the new value. When identitySync is run in the "do" mode this will use the new value for memberAdd to add the account to the group AMXdemo. identitySync is reversible so the memberDel is the transaction that would be done in "undo" mode.

There is only one group added or removed because identitySync only manages groups that are defined in the identity source. Existing groups in the Active Directory that are not managed by identitySync are not changed.

The add and delete behaviour of the transaction is caused by the delta attribute flag. If this is removed the Active Directory account will have all its group membership changed to those defined in the identity source. In this case AMXdemo group would be the only group that the account would be a member of. Clearly the undo feature is very appropriate for a situation like this. Groups are separated with the character or characters defined in the property ActionFileDelimList which in this case is defined as ":".

3. Defining groups in an Identity Source using a lookup table

The same process can be used to create group membership by virtue of an Identity Attribute. Choosing the attribute depends on what is available in identity source. Since this is a tutorial, it is not necessary to use an attribute like Location. An attribute that will work in every situation is accountName.

A lookup table will be used to populate the memberOf attribute based on the value of a person's attribute. The CSV Schema will be changed to use the lookup table and identitySync re-run.

Create a Lookup Table.

Open DynamicGroups.txt in the Tutorial1 directory. It assumes location is available and contains:

```
London,LondonRW:EdinburghRO:LeedsRO:GlasgowRO
Edinburgh,LondonRO:EdinburghRW:LeedsRO:GlasgowRO
Leeds,LondonRO:EdinburghRO:LeedsRW:GlasgowRO
Glasgow,LondonRO:EdinburghRO:LeedsRO:GlasgowRW
*,LondonRW:EdinburghRO:LeedsRO:GlasgowRO
```

This is intended to dynamically add users to groups based on their location, giving them RW (Read and Write) to their local group and RO (Read Only) to the others. The default is defined by the line starting with a "*" and matches the London user's groups. Groups defined in this lookup table will be the only groups that will be managed. Modify the DynamicGroups.txt file to suit attributes in your Active Directory. For example if the attribute location is not populated use accountName. Replace everything in DynamicGroups.txt with:

```
CraigA,CraigsList
*,
```

This will add the account CraigA to the group CraigsList and none others. No other accounts will be affected.

Update CSV Schema

Edit IdCSVSchemaAD1.txt. Use the “lookup” Modify Attribute to Transform the “memberOf” Attribute.

Remove or comment out “,memberOf”.

When using location add

```
location,memberOf;lookup=DynamicGroups.txt
```

When using accountName add

```
accountName,memberOf;lookup=DynamicGroups.txt
```

Run identitySync

Run identitySync again and check the Transaction file ActionFile.txt.

```
..\identitySync.exe ActiveDirectory2.properties
```

Next Steps

To actually make a change in the Active Directory create the target group and run identitySync in the “do” mode.

```
..\identitySync.exe ActiveDirectory2.properties do
```

To reverse the change

```
>..\identitySync.exe ActiveDirectory2.properties undo
```

For further information concerning the use of Dynamic Groups in AMX see the reference documentation. Facilities allow lookups of combinations of Attributes, for example Attributes isaManager and location could be used to create a group containing only the managers at each location of the organisation.

Multiple Dynamic Groups can be managed by using the concatenate Attribute Modifier. For example add accounts to groups based on their location, and then add them to more groups based on their job title.

AMX can use Role Templates rather than lookup tables to manage Dynamic Groups in the ActiveDirectory. The Role Templates can be created using roleAnalayzer. When Role Templates are updated the changes propagate to the group membership of persons having the role.

There are tutorials for both creation and the use of templates on the web site.