AMX Account Management

This tutorial is for IT staff who are experienced in identity management, it requires insight into how the Active Directory works, and a working knowledge of Windows. It should be done after the first tutorial.

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

- Creating account names with predictable formats
- Disabling accounts using an end date
- Deleting accounts by moving them to a Re-cycle bin
- Creating Metaverse values using immediate evaluation of attributes
- Creating Metaverse values by combining them with the concatenate function.
- Updating First Names with Preferred Names when present, using the ConcatIfNull attribute flag.

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. This tutorial assumes step 1 of Tutorial AD 1 has been performed successfully and an identityReport identityReportAD1.csv has been created from the active directory and that the resources names and passwords have been updated in the properties file ActiveDirectory2.properties step 2.

1. Account Create

This excercise will create a new account.

1. Edit the Properties file ActiveDirectory2.properties, it is in the <installDirectory>\Tutorial1 directory, change the LoadMode to Create and Update. The account password will be formed from the password template where C is an

uppercase consonant, c lowercase, V is an uppercase vowel, v lowercase and n is a numeric. Modify the template if the Active Directory password policy requires a password longer than 8 characters.

```
ActiveDirectoryLoadMode = CRU
ActiveDirectoryPasswordTemplate = CvcnCvcn
```

2. Open identityReportAD1.csv and copy and paste a new record to making all the attribute values unique. Leaving just the accountName blank. Check that the active flag is Y.

3. Run identitySync.exe in the analyse mode. Notice 1 new account was found. Check ActionFile.txt has a Create

4. Run identitySync.exe in the do or redo mode to create the account.

```
C:\AMX\Tutorial1>identitySync.exe ActiveDirectory2.properties do
Warning: Not run as administrator
Begins Fri, 29 Apr 2016 14:59:17 GMT  do
CSVIdentity 1 C:\AMX\Tutorial1\IdentityReportAD1.csv
Last updated 29/04/2016 14:54:42
Extracted 103 Identities
Total of 103 Identities

ActiveDirectory 1 192.168.121.61
Extracted 102 Accounts
Account joins    102
Account creates  1
Account updates  0
Account disables 0
Account deletes  0
ActiveDirectory Finished Fri, 29 Apr 2016 14:55:28 GMT

ActiveDirectory 1 corp
ActiveDirectory Load corp Create do cn=ailsa sutherland,ou=lon,ou=accounts,dc=corp,dc=example,dc=com
ActiveDirectory Finished Fri, 29 Apr 2016 14:59:18 GMT

C:\AMX\Tutorial1>
```

5. Notice that:
   - The analyse phase again found 1 new user, which it then created in the do phase.

6. Use the ActiveDirectory Users & Computers MMC to find the new user. Note that a refresh of the MMC display will be required to show the new user. In production situations an email will be sent to the new user's manager with details of the new account. See tutorial AD3 for details.

7. Run identitySync.exe in the undo mode to remove the account. This is the only situation where a delete occurs in AMX.

8. Remove the new user from identityReportAD1.csv

Failed to create User

Check the "ActionFile.txt" if does not contain a create action, check that ActiveDirectoryLoadMode has been changed to CRU (Create and Update)

Error: ActiveDirectory Load Create Access is denied.

The account defined in the Active Directory properties file does not have administrative rights in the domain.

Error: ActiveDirectory Load Set Password failed Exception has been thrown by the target of an invocation.
Check debug.txt

```
System.Reflection.TargetInvocationException: Exception has been thrown by the target of an invocation. --->
System.Runtime.InteropServices.COMException (0x800706BA): The RPC server is unavailable. (Exception from HRESULT:
0x800706BA)
```

This is usually caused by the Firewall blocking. Check firewall logs (you may have to turn logging on for dropped packets). Logs are in C:\WINDOWS\system32\LogFiles\Firewall\pfirewall.log

```
Error: ActiveDirectory Load  Create The directory service cannot perform the requested operation on the RDN attribute of an
object.
```

The account already exists in the Active Directory, the uniqueness check only works for accounts that were loaded from the ActiveDirectoryAccountContainer.

2. Account Disable

identitySync uses the "active" metaverse attribute to enable or disable an account, based on its value or an EndDate. This exercise will show the use of both methods to enable and disable accounts. Use of the active flag:

1. Update the Active Directory properties file, change ActiveDirectoryLoadMode to CRUD, (Create, Update and Disable).
   ```
   ActiveDirectoryLoadMode = CRUD
   ```

2. Update identityReportAD1.csv, change the active attribute of an account in to "N".

3. Run identitySync.exe in the analyse mode. Check the ActionFile.txt contains an update such as:
   ```
   09/12/2013|corp;Smithb|Update|active=Y|active=N
   ```

4. Run identitySync.exe in the redo mode to update the Active Directory. Check that the account has been disabled in the Active Directory. A refresh of the MMC display will be necessary.

5. Run identitySync.exe in the undo mode to re-enable the account. Check that the account has been enabled in the Active Directory. A refresh will again be necessary. Notice that other bits in the userAccount Control attribute are unchanged, for example “smart card is required for interactive login” remains set during the disable and re-enable process.

6. Update identityReportAD1.csv, change the active attribute of the account back to “Y”.

Use of the EndDate. The end date is the last day of work, by default the account will be active until midnight. See the reference document for details of how to terminate an account earlier in the day.

1. Update the IdCSVschemaAD1.txt file, comment out the “active” staging attribute and use endDate to indicate active.
   ```
   //active,;active
   endDate,active;active;endDate
   ```

2. Update IdentityReportAD2.csv, add an Endate earlier than today in any record that has the active attribute “N”. Modify another active record adding an Endate earlier than today to disable it.

3. Run identitySync.exe in the analyse mode. Check the ActionFile.txt contains a disable update for the record selected above.

3. Update identityReportAD1.csv, remove or change the endDate attribute of the account to make the account active again.

3. Account Delete

AMX and identitySync never deletes anything in any of its managed systems, a delete by identitySync in the Active Directory moves the account into the deletedUsers container that is defined in the ActiveDirectoryDeletedContainer attribute in the ActiveDirectory2.properties file.

1. Update the ActiveDirectory2.properties file, change ActiveDirectoryLoadMode to CRUDD, (Create, Update, Disable and Delete).

2. Update the ActiveDirectory2.properties file, change the ActiveDirectoryDeletedContainer to the name of the OU where your users are placed when they are no longer in use. Create the ActiveDirectoryDeletedContainer OU in the Active Directory if it does not exist.

   ```
   ActiveDirectoryDeletedContainer1 = OU=Deleted,OU=AMX,DC=corp,DC=example,DC=com
   ```

3. Cut an active identity record from identityReportAD1.csv. Save it in the clipboard.

4. Run identitySync.exe in the analyse mode. Check that identitySync reported that the account was still active and that the account was not moved to the deletedContainer. This is a safety feature of AMX it will not move accounts to the deletedContainer while they are active and may still be used.

5. Disable the account in the Active Directory and re-run identitySync in the do mode. Check that the account has been moved to the ActiveDirectoryDeletedContainer using the MMC. A refresh will be required.

6. Run identitySync.exe in the undo mode to restore the account. Check that the account is restored to its original OU and remains disabled

7. Paste the identity back into identityReportAD1.csv and remove or change the endDate of the account to its original value to enable the account.

8. Run identitySync.exe in the analyse mode and check that the account will be enabled

9. Run identitySync.exe in the redo mode to enable the account.

4. Account Name Formatting

The format used for creating unique new account names is defined in the attributes ActiveDirectoryAccountNameTemplate and ActiveDirectoryAccountNameFormat in the ActiveDirectory2.properties file.

The template defines the attributes that will be used to create the account number, these are surrounded by “%” characters. Typically these are First Name, Last Name, Initials or employeeID. Anything not surrounded by “%”s are constants and included in the account name, for example a prefix. An attribute called “*” is the discriminator, it is a number that is used to create a unique account name. There can be only one discriminator in the template.

The format defines the preferred number and the maximum number of characters for each attribute, constant and the discriminator. These are in the form of n.m separated by “,”s. There must be a format descriptor for each attribute, constant and discriminator defined in the template.

Unique Account Name from attributes

This exercise will show unique account names being constructed using accountNameTemplates and Formats. When a unique account name cannot be constructed the process aborts. For example:

```
ActiveDirectoryAccountNameTemplate = %lastName%%firstName%
ActiveDirectoryAccountNameFormat = 7.7,1.2
```

This will create account names such as:

SmallA, SmallAl

The format for the firstName allows a second character from the firstName to be used to make the account unique.

1. Update identityReport1.csv, adding two users with similar first and last names

firstName	preferredName	initials	lastName	displayName
Aileen		L	Small	Small, Aileen
Alison		L	Small	Small, Alison

Add values for all the attributes, especially:

- distinguishedName
- name,name
- CN,CN
- displayName,displayName
- firstName

- lastName

2. Edit ActiveDirectory2.properties, setting ActiveDirectoryAccountNameTemplate and ActiveDirectoryAccountNameFormat as above

3. Run identitySync.exe in the analyse mode. Notice 2 new accounts were found.

4. Check the Action.txt file for the account names that would be created.

```
08/12/2013|corp;SmallA|Create|CN=Aileen Small
08/12/2013|corp;SmallAl|Create|CN=Alison Small
```

Unique Account Name from attributes with discriminator

AMX can add a numeric to make the account names unique. This shows numeric discriminators in use, for example:

```
ActiveDirectoryAccountNameTemplate1 = %lastName%%firstName%%*%
ActiveDirectoryAccountNameFormat1 = 7.7,1.1,0.2
```

This will create account names such as:

```
SmallA, SmallA01
```

The discriminator when present is used to make the account name unique using the maximum number of characters.

1. Update Edit ActiveDirectory2.properties, setting ActiveDirectoryAccountNameTemplate and ActiveDirectoryAccountNameFormat as above

2. Run identitySync.exe in the analyse mode. Check the Action.txt file for the account names that would be created.

Failed to create 2 accounts

```
Error: AccountNameGen failed to create unique account name from %lastName%%firstName% 7.7,1.1
Error: ActionFile CheckAccounts Create account has blank account name for Small, al
```

Discriminator not defined or not long enough, or as in this case discriminator has not been defined.

Object already exists

```
ActiveDirectory Load corp Create cn=ailsa sutherland,ou=lon,ou=accounts,dc=corp,dc=example,dc=com
Error: ActiveDirectory Load corp Create The object already exists.
```

distinguishedName not unique, add the attribute modifier ”unique” to distinguishedName in CSVSchema.txt. The process will still fail, but it will be in the Transform phase. Create a unique distinguishedName.

No such object on the server

```
ActiveDirectory Load corp Create cn=andy small,ou=ednZZ,ou=accounts,dc=corp,dc=example,dc=com
Error: ActiveDirectory Load corp Create Exception There is no such object on the server.
```

When the distinguished name does not match the OU structure. Or

```
ActiveDirectory Load corp Create cn=andy small,ou=edn,ou=accounts,dc=corp,dc=example,dc=com
Error: ActiveDirectory Load corp Create Exception There is no such object on the server.
```

More details in the debug.txt file, the distinguishedName of a manager does not exist.

Account Name from employee ID

This exercise shows accountNameTemplates creating accounts from other strings and attributes, for example:

```
ActiveDirectoryAccountNameTemplate1  = E%employeeID%
ActiveDirectoryAccountNameFormat1 = 1.1,6.6
```

Will create account names such as:

E75151, E66762

1. Edit ActiveDirectory2.properties, setting ActiveDirectoryAccountNameTemplate and ActiveDirectoryAccountNameFormat as above

2. Run identitySync.exe in the analyse mode. Check the Action.txt file for the account names that would be created.

5. Creating Metaverse Values

AMX allows Metaverse values to be created from existing attributes as the Identity records are extracted. This is intended for attributes in an authoritative source of identities, where they can be coed or modified to create values that are formatted properly for the target resource, which in this tutorial is the Active Directory.

Immediate Evaluation

Attribute modifiers that are evaluated in place from left to right are:

1. comma will convert a string to a comma format. See also nocomma. For example:
   ```
   Name,fullname;comma
   ```
   Will convert an attribute value such as Alan J Brown to Brown, Alan J

2. escape, add the "\" to a value containing ","s for instance a CN of the form lastName, firstName. Escape has no effect where there is no ",". See also noescape.

3. left, extract substring number starting 0 on left after splitting the string with the specified delimiter. See also right. For example:
   ```
   Name,lastName;left0,
   ```

Evaluating Name containing a value “Small, Alan”

Uses “,” to split the value and returns the first leftmost part.

Sets lastName to “Small”.

```
Name,firstName;left1,
```

Evaluating Name containing a value “Small, Alan”

Sets firstName to “Alan” trimming any leading or trailing spaces.

```
Name,firstName;left2,
```

Evaluating Name containing a value “Small, Alan”

Sets firstName to “”.

Evaluating Name containing a value with no delimiter returns the whole value.

```
distinguishedName,CN;left0,;left1=
```

Evaluating distinguishedName containing a value "cn=alastair brown,ou=lon,ou=accounts,dc=corp,dc=example,dc=com",

First uses “,” to split the value and returns the first part cn=alastair brown

Then uses “=” to split the returned value and returns the second part “alastair brown”.

4. lookup, converts the value of the attribute by looking it up in a table constructed by reading a CSV file. For example:

```
departmentCode,department;lookup=department.csv
```

The department.csv file containing:

```
1001,Sales Dept
1002,Marketing Dept
*,General
```

This will convert an attribute value containing 1001 to Sales Dept
Other possible entries in the the CSV file are =,= and *, these are mutually exclusive:
=,= copies the lookup value to the resulting output when there is no match
* is the default value when there is no match.

If the lookup key contains a comma enclose it in “s, such as:
“London, England”, UK

5. noescape, remove the “\” escape character from a value containing “\,”. See also escape.

6. nocomma, the reverse of comma. For example:

```
Name,fullname;nocomma
```

Evaluating a Name containing a value “Brown, Alan J” returns “Alan J Brown”

When the value of the attribute does not include a “,” no conversion is done.

Evaluating a Name containing a value “Alan J Brown” returns “Alan J Brown”

Names with “De”, “Mc”, “Mac”, “Van” and “Von” are converted properly.

Evaluating a Name containing a value “Robin van der Waal” returns “van der Waal, Robin”

7. replace, use Regular Expressions (Regex) to replace strings. See http://msdn.microsoft.com/en-us/library/az24scfc(v=vs.110).aspx for a quick reference guide

```
description,employeeID;replace/Staff-|staff-//
```

   Will remove “Staff-“ or “staff-“ from the string such as Staff-75151 becomes 75151

```
description,employeeID;replace/(?i)[a-z-]//
```

   Will remove any alphabetic character case insensitively and “-“ from the string such as Staff-75151 becomes 75151

```
description,employeeID; replace/[a-zA-Z]+-*(\d*)/$1/
```

   As a demonstration of the power of Regex, this will remove any upper or lower case alphabetic character, zero or more “-“ and match the remaining numbers which are the result. A string such as Staff-75151 or Staff75151 becomes 75151

8. right, extract substring number starting 0 on right with a specified delimiter. See also left

```
distinguishedName,ou;right2,;right0=
```

   Will return the value “Edinburgh” from a string containing
   “CN=Philip Nesfield,OU=Users,OU=Edinburgh,DC=example,DC=com”

9. title, capitalise the first character of each word

```
description,description;title
```

   Will return the value “Sales & Marketing” from a string containing “Sales & marketing”

10. toLower, convert the value to lower case.

11.toUpper, convert the value to upper case.

12. trim, remove leading and trailing spaces from a value

Attribute modifiers are evaluated from left to right. For example:

```
payrolllocation,city;right0/;trim;lookup=city.csv
```

operating on a value “EMEA UK / Edinburgh”
with city.csv containing Edinburgh,Scotland
returns Scotland

Post Evaluation

After all the attributes are extracted post evaluation modifiers are processed

1. concatIfNull, only updates the metaverse value if it is blank. For example:

```
givenName,
preferredName,firstName;ConcatIfNull:%givenName%
```

   Will always store the preferredName in the firstName attribute, but when it is blank the givenName will be stored

2. concat, concatenate metaverse attributes and store them into a new one.

   Metaverse attributes are delimited with ‘%’, everything else is text. For example:

```
,fullname;concat:%lastname%, %firstname%
,comment;concat:ActiveDirectory schema version 3.1 - 10/12/2010
```

   Concats are processed in the order that they appear in the Schema. So, both of these produce the same result:

```
,CN;concat:CN=%name%
,distinguishedName;concat:%CN%,ou=%ou%,dc=example,dc=com
```

or:

```
,distinguishedName;concat:CN=%name%,ou=%ou%,dc=example,dc=com
```

6. CN from the first and last names

This exercise shows how to build the CN from the first and last names identity attributes

1. Update IdCSVschemaAD1.txt. Replace the CN with an evaluated value using a concat to create a CN by concatenating the first and last names in the format used in the ActiveDirectory, for example:

   ```
   ,CN;concat:%firstName% %lastName%;unique
   ```

   or

   ```
   ,CN;concat:%lastName%\, %firstName%;unique
   ```

2. Run identitySync.exe in the analyse mode, check Action.txt to see the updates that are required, that is the CN evaluated in the identity source and found in the ActiveDirectory are different. This indicates inconsistencies in the Active Directory. For instance where last names have been changed but the CN has not. identitySync could be used to correct this.

3. Change IdCSVschemaAD1.txt to its original values.

Notice that the CN uses the “unique” attribute flag, this checks for uniqueness but does not enforce it. In cases where HR does not give individuals unique full names, a discriminator must be used. The discriminator must be based on a unique attribute, for example employeeID.

1. Add the employeeID and a descriminator value to a lookup file, for example descriminator.csv. The default “*,” is no descriminator. For example:

   ```
   2142,1
   ```

```
          *,
```

2. Update the Identity schema IdCSVschemaAD1.txt. For example:
   ```
         employeeID,descriminator;lookup=descriminator.csv
         ,CN;concat:%firstName% %lastName%%descriminator%;unique
   ```

   This would add a "1" to CN of employee 2142

7. distinguished name from Identity attributes

This excercise constructs the distinguishedName from Identity attributes such as firstName, LastName and location or organisation. In a production situation when managing the ActiveDirectory or LDAP the distinguishedName can be constructed in the Identity Metaverse, or a CN used with the AccountContainer property during the load phase.

In a production situation it is assumed that there is some logic to the allocation of accounts to OUs that identitySync can use to place newly created accounts directly into the intended OUs. For example accounts are allocated to OUs based on the account owners location or department:

1. Update IdCSVschemaAD1.txt. Replace the distinguishedName with an evaluated value, substituting the Canonical Name CN with the same evaluation as the previous example and the OU. The leftmost OU for accounts is in identityReportAD1.csv as created by identityReport1.exe, see Tutorial AD1 for how this was obtained.
   ```
   ,distinguishedName;concat:CN=%firstName% %lastName%,ou=%ou%,ou=accounts,dc=corp,dc=example,dc=com
   ```

2. If your organisation uses an OU structure based on a business or location, evaluate the OU. In step 2 of Tutorial1 multiple OUs were extracted, for example:
   ```
         ActiveDirectoryFilterValue1 = OU=EDN:ou=lon
   ```
   In situations like this the location can be used to derive the OU, for example the location ends in a city name that can be mapped to an OU. Update IdCSVschemaAD1.txt

```
location,ou;right0 ;lookup=city.csv
```

The right evaluator will extract the word at the end of a location which can then be looked up in a table of cities and OUs in a file city.csv, for example city.csv contains:

```
London,LON
Lon.,LON
Edinburgh,EDN
Edin.,EDN
*,LON
```

3. In situations where OUs are based on a structure of business and location, evaluate OU1 based on location and OU2 based on business. Combine them in the evaluation of the distinguishedName

4. Run identitySync.exe in the analyse mode, check Action.txt to see what updates that are required. The updates will show how close the evaluation of the distinguishedName represents reality in the ActiveDirectory. Updates indicate an anomaly between an account's OU and an individual's address – the individual may have told HR that they have moved location without informing IT. identitySync could be used to correct this.

5. If these anomalies are not intended to be corrected before identitySync.exe is run in production in the do mode, prevent the updating of distinguishedName by adding nosync in the ActiveDirectorySchema1.txt and use the CN when creating a new account: For example in the Identity schema:

```
distinguishedName,distinguishedName;nosync
,CN=%firstName% %lastName%;unique
```

6. Run identitySync.exe in the analyse mode, check Action.txt to verify that no distinguished name updates are present.

8. Preferred Name

In some organisations the identity source maintained by HR uses given names and the accounts use preferred names. For example a person in the HR database may have a given name Elizabeth and a preferred name Liz. Their account in the Active Directory uses Liz. Often records have blank preferred names. identitySync is able to accommodate this using the concatIfNull attribute modifier, see IdCSVschemaAD1.txt.

This exercise shows the use of a preferredName when present, being used to replace a firstName

1. Review IdCSVschemaAD1.txt note:

   ```
   firstName,givenName
   ```

   Adding the attribute modifier concatIfNull:

   ```
   preferredName,firstName;ConcatIfNull:%givenName%
   ```

2. Edit identityReportAD1.csv modify a record, for example change:

accountName	name	CN	firstName	preferredName	lastName	displayName
BrownL	Liz Brown	Liz Brown	Liz		Brown	Brown, Liz

To:

accountName	name	CN	firstName	preferredName	lastName	displayName
BrownL	Liz Brown	Liz Brown	Elisabeth	Liz	Brown	Brown, Liz

3. Run identitySync.exe in the analyse mode and verify that there are no changes in the Action File, indicating that the preferred name was used for the firstName.