AMX Match Accounts to Identities

This tutorial is for IT staff who are experienced in identity management.

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

- Run accountMatch on the demo data.
- Run accountMatch on identity and account data extracted from your environment.

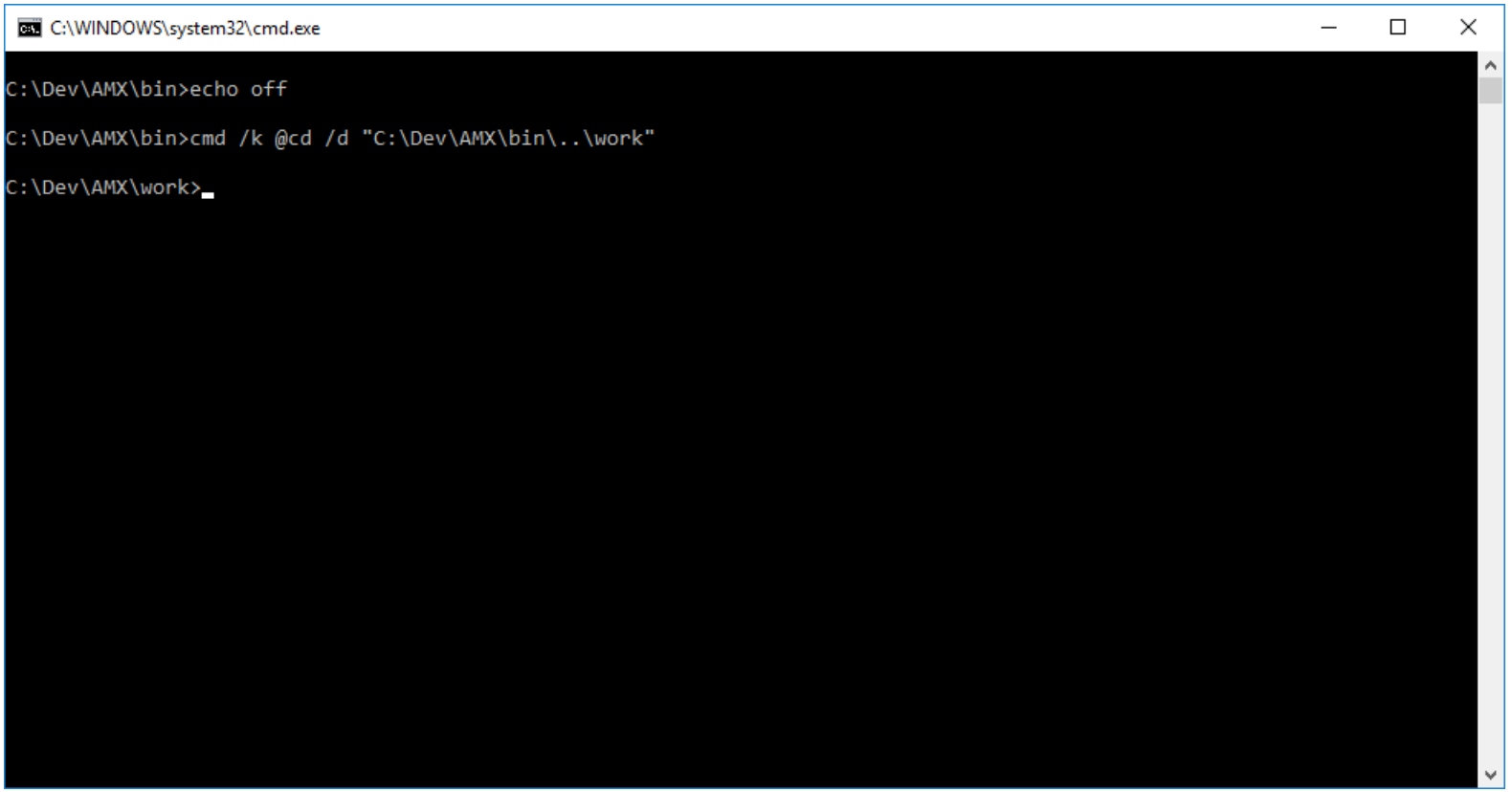
1. Setup

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.

2. Run accountMatch on the Synthetic data

The synthetic data is entirely artificial and is for demonstration purposes.

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



Run accountMatch.exe accountMatch1.properties

```
C:\AMX\Tutorial2>accountMatch AccountMatch1.properties
Begins Mon, 31 Oct 2016 13:08:11 GMT
CSVidentity 1 Identity.csv
Extracted 6 Identities
CSV1 C:\Dev\AMX\Tutorial2\Account.csv
Extracted 12 Accounts
2 matches. = 33% of Identities
```

```
C:\AMX\Tutorial2>
```

This is a simple join of:

MatchJoin = FirstName:firstName,LastName:lastName

Resulting in 2 joins:

```
Join. Exactly one match for Account entry firstName='Ellen' AND lastName='Stevenson' for Identity
FirstName='Ellen' AND LastName='Stevenson' Stevenson->Stevenson, Ellen

Join. Exactly one match for Account entry firstName='Alpin' AND lastName='Thomson' for Identity
FirstName='Alpin' AND LastName='Thomson' Thomson->Thomson, Alpin
```

There are several reasons why there were not more. The match.log provides the details. The first we will demonstrate is a duplicate name:

```
Duplicate Identity FirstName='Andrew' AND LastName='Robertson' for Identity Robertson
Duplicate Identity FirstName='Andrew' AND LastName='Robertson' for Identity Robertson
```

Adding more join attributes reduces duplicates. Adding location for example, run accountMatch.exe accountMatch2.properties on the identity and account files written by the first run and using a join:

```
MatchJoin = MatchJoin = FirstName:firstName,LastName:lastName,City:location
```

Results in 1 additional join. The issue with Robertson is now that for HR the first name is Andrew and for IT it's Andy. These are can be discriminated because in the HR database one of them has a preferred name of Andy. The identity Schema creates preferred names for all the identities by using the concatIfNull attribute flag. This flag updates the attribute only when the value is

null. So for each record the preferred name is used unless it is blank and in this case the first name is used to update preferred name. More details in the AMX Reference Guide:

```
FirstName,
PreferredName,PreferredName;ConcatIfNull:%FirstName%
```

Run accountMatch.exe accountMatch3.properties

```
MatchJoin = PreferredName:givenName,LastName:lastName
```

This results in 1 additional join. Another common issue is the use of "'" in the last name of O'Donovan. The "'" is removed in the account attributes. The identity schema uses the replace attribute modifier to remove it. More details in the identitySync reference guide. The identity schema has an entry:

```
LastName,LastNameNew;replace/''//
```

Run accountMatch.exe accountMatch4.properties

```
MatchJoin = PreferredName:givenName,LastNameNew:lastName,City:location
```

This will find O'Donovan. If the account management process consistently removes all "'" the replace attribute modifier would be added to the schema as:

```
LastName,LastName;replace/''//
```

The last remaining failed identity is:

Cannot find Account entry givenName='Wang' AND lastName='Lee' AND location='London' for Identity Lee

This is a common problem with Chinese identity records. The first and last names are reversed. An additional run of accountMatch accountMatch5.properties with a join of:

```
MatchJoin = PreferredName:lastName,LastName:givenName,City:location
```

Finds the last record. Check the identity6.csv file written by the last run of accountMatch it should be empty. A matchTemplate template:

```
matchTemplate = dsquery -samid %accountName% | dsmod user -empid = %IdentityUnique%
```

Creates a match.csv file containing a batch file to update all 6 of the matched accounts in the Active Directory:

```
dsquery -samid StevensonA | dsmod user -empid = 305
dsquery -samid ThomsonA | dsmod user -empid = 206
dsquery -samid RobertAn | dsmod user -empid = 207
dsquery -samid RobertsA | dsmod user -empid = 306
dsquery -samid OdonavoA | dsmod user -empid = 205
dsquery -samid Wangl | dsmod user -empid = 209
```

The template can be used to update any account attribute, and used to update other account resources. Unix. For example to update Unix a template like for example:

```
matchTemplate = usermod -c  "%firstName%
%lastName%,%location%,%telephoneNumber%,%mobile%,%IdentityUnique%" %accountName%accountMatch
```

3. Other Joins

Lookups

In situations where the account report does not have location to match the identity attribute, it can be derived using the lookup attribute modifier in the account schema. Lookup searches for a value in a lookup table defined in a file. For example:

```
EDN,Edinburgh
LON,London
MAN,Manchester
=,=
```

The account schema derives the location by looking up the ou:

```
ou,ouLocation;lookup=location.csv
```

With a join as:

```
MatchJoin = FirstName:firstName,LastName:lastName,City:ouLocation
```

Run accountMatch.exe accountMatch10.properties. 3 matches will be found, demonstrating that the lookup was successful.

accountName generation

When an organisation consistently creates account names from the first and last names, re-creating the account name from the identity attributes and using it to match accounts. For example account names constructed by adding the first character of the first name to the last name can be reconstructed in the identity schema using:

```
FirstName,FirstNameShort;truncate1
LastName,LastNameShort;replace/'//;truncate10
,AccountName;concat:%LastNameShort%%FirstNameShort%
```

With a join as:

```
MatchJoin = AccountName:accountName
```

Run accountMatch.exe accountMatch11.properties, 2 matches will be found, demonstrating that the lookup was successful. Check the match.log. Notice that the account names of Andy Robertson and Andrew Robertson were both constructed as RebertsonA so they could not be discriminated.

Use of email for first and last names

Sometimes when a person changes their surname after marriage or divorce the new last name is not updated consistently, especially when it's not public. Changes are often just made to email and HR or Payroll. Consequently accountMatch can match the first and last names from email with HR using the attribute modifier "left" which returns the leftmost nth field using the specified delimiter. In this example the email format is first.last@example.com, so the "left" attribute modifier can be used:

```
mail,firstNameEmail;left0@;left0.
mail,lastNameEmail;left0@;left1.
```

Using firstNameEmail and LastNameEmail to match the FirstName and LastName in the Identity record:

```
MatchJoin = PreferredName:firstNameEmail,LastName:lastNameEmail
```

Run accountMatch.exe accountMatch12.properties, 4 matches will be found, demonstrating that the lookup was successful. Check the match.log.

Use of Display Name to extract First and Last Names

In cases where first and last name attributes are not available, these can be often obtained from the display name. In this example the display name format is lastName, firstName initial, there are a number of ways that the names can be extracted, using "nocomma" changes the format to firstName initial lastName so that left and right can be used:

```
displayName,firstNameDisplay;nocomma;left0
displayName,lastNameDisplay;nocomma;right0
```

Using firstNameDisplayl and LastNameDisplay to match the FirstName and LastName in the Identity record:

```
MatchJoin = PreferredName:firstNameDisplay,LastName:lastNameDisplay
```

Run accountMatch.exe accountMatch13.properties, 4 matches will be found, demonstrating that the lookup was successful. Check the match.log.

4. Run accountMatch on Your Organisation's Data

In this tutorial the identities will be from your HR system, and accounts from the Active Directory. These sources usually have a rich set of attributes which can be used to match identities with their accounts.

Extract Identity and Account Data

accountMatch uses a CSV file of identities and a file of accounts. The HR system is the usual source of the identity file. Get a CSV report from the HR system if possible, accountMatch reads CSV files only. If the file is not CSV, for example SAP text file format, or multiple files and formats, identityReport can be used to reformat it into a single consistent CSV file.

The account file can be created with identityReport. Tutorial AD1 describes configuring and running identityReport to create a CSV file of the accounts in the Active Directory. Refer to the AMX identitySync Reference documentation for other account resources. Copy the files to the Tutorial2 subdirectory.

1. Open the identity CSV file and the accountMatch.properties file. Copy the header and paste it into the identityAttributes property.

Update accountMatch properties

2. Create a new file identitySchema.txt and paste the identity file header in to it. The complete format of a schema file is described in the identitySync reference documentation. Add carriage returns after the comma of each attribute so that each attribute is on a new line. For example:

```
EmployeeNumber,
Organisation,
FirstName,
```

3. Pick an attribute for the DisplayName, add ";DisplayName" to the attribute.

4. Select a suitable attribute for the IdentityUnique property which will be used in the matchTemplate.

5. Notice that unlike identitySync it is not necessary to have consistent attribute names in the Metaverse. The attribute names are referenced only in the MatchJoin property.

6. Save the file and add its name to the CSVidentityschema1 property in accountMatch.properties.

7. Repeat the steps 2 to 5 for the account CSV file, updating the accountSchema1 property in CSVschema1.properties.

8. Update the properties CSVidentityResource1 and CSVResource1 in accountMatch.properties with the names of the identity and account CSV files.

9. Update the identityReport and accountReport properties with suitable names of the output files, for example 2IdentitySurvey.csv.

10. Select a MatchJoin for the first cycle. For example:

```
FirstName:firstName,LastName:lastName
```

Run accountMatch

11. Run accountMatch.exe and review the match.log file. This will give you a good indication of what to try to match next and any transforms that may be required

Repeat

12. Copy the accountMatch properties file and paste it with the next name in the sequence. For example copy accountMatch.properties into 2accountMatch.properties.

13. Open the new properties file and update the CSVidentityResource1 and CSVResource1 properties with the identityReport and accountReport names from the first run.

14. Update the identityReport and accountReport names for the output of the unmatched identities and accounts for any subsequent run for example 3IdentitySurvey.csv.

15. Change matchAppend to true which will add new matches to the existing file.

16. Add any attribute transforms to the identity schema, perhaps creating a new attribute to use for the join. For example:

```
mail,firstNameEmail;left0@;left0.
```

17. Change the MatchJoin with another join rule, see the previous section. For example:

```
firstNameEmail:firstName,LastName:lastName
PreferredName:firstName,LastName:lastName
LastName:lastName,location:City
AccountName:accountName
```

18. Run accountMatch.exe and review the match.log file.

19. Repeat steps 11 to 16 until all matchJoin combinations are exhausted.