AMX identitySync and identityReport Reference

Table of Contents

Taxonomy

An Identity source is an instance of an adapter in AMX that corresponds to an authoritative source of identities.

A managed resource is an instance of an adapter in AMX that corresponds to a system with user accounts. AMX manages the accounts.

Accounts are objects or records in a resource that users can use to logon and authenticate themselves. Accounts give access to the resource.

Adapters are AMX classes of resource types, for example ActiveDirectory, LDAP, Excel etc.

Instances of adapters are configured using properties files to correspond to identity sources or resources for example Production ActiveDirectory, Integration Test ActiveDirectory etc.

Components of AMX

This document describes identitySync, identityServer and identityReport. Other components are accountMatch, roleAnalyze, and SODanalyze are described elsewhere.

identitySync, a digital identity lifecycle management system that creates, updates, and de-activates accounts on managed resources based on the account owner's status within the organisation as maintained by HR or other authoritative sources. Accounts are created when a person joins the organisation, updated when they change roles and de-activated when they leave.

Other programs are intended to be used during the configuration and release to production phase. They are all consistent with one another, for example the properties file can have the identityReport specific properties added to it and then it can be used by both identitySync and identityServer.

identityServer implements a remote adapter for identitySync and identityReport that is intended to operate as a proxy through a firewall where the well-known ports used by LDAP, SQL, Windows etc are blocked for security reasons. identityServer uses a single nominated port. It uses the same adapters, with the same properties and schemas as identitySync. It has additional properties of its own to define the network and port.

accountMatch: recursive matches accounts to identities and creates a report. Separately documented. It uses the CSVIdentity and CSV adapters, with the same properties and schemas as identitySync. It has some additional unique properties of its own. When matching accounts from other adapters use identityReport to create a CSV file.

identityReport: creates a report containing the identity and account attributes from all the resources. It uses the same adapters, with the same properties and schemas as identitySync. It has 5 additional unique properties of its own all prefixed Report. identityReport creates a report which can be used during development to review the attributes, their values and the results of attribute transforms. For example the alphabetic case of the attributes. Reports can be sorted by the join attribute used in identitySync to review the matching of identities and their accounts.

identitySync

identitySync is run as a scheduled task or from a command prompt:

```
identitySync.exe propertiesFile [info|do|redo|undo]  [transaction file]
```

If the propertiesFile is qualified with a directory, identitySync sets its current directory to that directory so that other properties do not need to be qualified. For instance a properties file Production1\identitySync.properties can contain references to the unqualified SchemaFile CorporateSchema.txt

The modes:

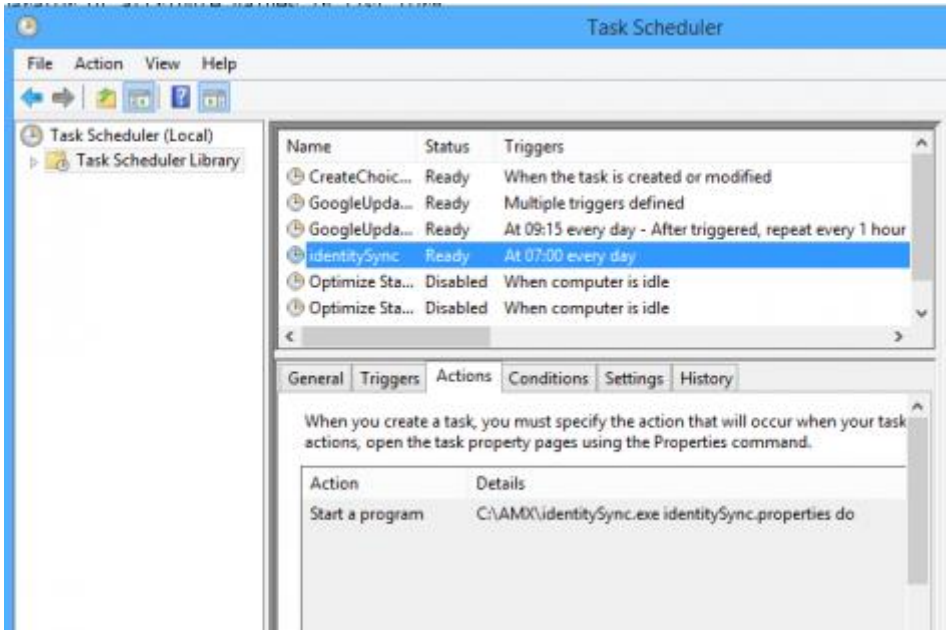
1. Default or Info creates the transaction file and exits
2. Do creates a new transaction file Active.txt and processes it
3. Redo processes an existing transaction file, doing each transaction with a date of today or earlier.
4. Undo processes an existing transaction file, undoing each transaction with a date of today or earlier.

The default transaction file is Action.txt. Transaction files are archived in a subdirectory Archive where they can be used for undo and redo operations.

The do mode is a complete refresh so after a change is made to the identity source and subsequently the managed resource, the identitySync process can be repeated multiple times without having any further effect on the managed resource. This is a requirement for a large replicated directories such as LDAP because it prevents un-necessary replication.

identitySync is fully reversible using the undo mode and the archive of transaction files makes it reversible to any level.

identitySync is intended to be run as a scheduled task synchronising accounts with their authoritative source of identities and reporting its activity by email.



When identitySync is updating the local system or a domain that it is a member of it should be run as an administrator otherwise an access denied error message will be printed during the update phase. A warning is logged in the info and debug files when identitySync starts and checks its environment state.

identityServer

identityServer can be run from the command line

```
identityServer.exe
```

The properties file is always identityServer.properties. identityServer implements the end point of the Remote adapter on a server using a nominated port for use in situations where a firewall blocks ports used by other adapters such as ActiveDirectory. identityServer Extracts and Loads adapters locally.

identityReport

identityReport can be run from the command line

```
identityReport.exe propertiesFile
```

identityReport creates a report of identities and / or accounts from all the adapters configured in the properties file. The report is sorted by the attribute defined in a schema having a join flag. See Attribute Flags.

The propertiesFile operates the same way as identitySync if it is qualified with a directory.

AMXUser

A Windows application for use by IT staff monitoring identitySync. identitySync usually sends email with a summary of its activity, for deeper insight AMXuser can search and display historical information. Examples of its use is to find the lifecycle of a particular account, its creation, updates, and disabling.

AMXUser is documented separately.

accountMatch

AccountMatch recursively matches accounts to persons or identities, it uses a file of identities and a file of Accounts. Typically the file of identities is created by HR. AMX tools identityReport is also available to create them. The match rule uses combinations of attributes common in both files, for example first & last names, employee numbers, phone numbers and locations. AccountMatch is intended to be used during the release to production process to identify the owners of all the active accounts.

The matched accounts can be tattooed by updating an attribute in each matched account in the managed resource, employeeID for example, or used by identitySync as a lookup table.

accountMatch is documented separately.

roleAnalyzer

roleAnalyzer is a Role Mining process that can create roles for use in an Identity Management system. roleAnalyze uses Identity Attributes to establish roles, then the responsibilities of every person with the role are summarised, and if the responsibilities are consistent, the role is established and defined. The role definition describes the responsibilities a person with this role should have. roleAnalyse can create a CSV file or a batch file to create the roles (template users) with the responsibilities (groups) in the ActiveDirectory which can then be used by an Identity Management system such as identitySync.

roleAnalyzer is documented separately.

SODanalyzer

SODanalyzer uses a spreadsheet of incompatible roles and responsibilities to review reports created by identityReport.

SODanalyzer is documented separately.

Properties

Properties are defined in the properties file with a format:

```
Property = value
```

Properties that are used by identityReport concerning the format of the report are ignored by identitySync:

1. Report, the name of the file that identityReport will write.

2. ReportAttributes a comma separated list of column headings in the Report file. The column headings must be defined consistently with the metaverse attribute names in the adapter schemas. Only attributes defined in ReportAttributes will be included in the report. If the property is blank all the metaverse attributes are output.

   If the extract has multiple different types of adapters with different schemas, this property should be used to define a consistent set of attributes present in all the schemas.

3. ReportAppend add the extract to the existing report.

4. ReportDelimiter the delimiter used in report, default is “,”.

5. ReportDelimList, the deliminator in Attribute values that are lists. The default is ActionFileDelimList.

Properties that are used uniquely by identityServer are ignored by identitySync:

1. Interface, the ip address of the network that identityServer listens on for systems with multiple network interfaces. The default is the DNS lookup of hostname.

2. Port, the port used.

3. Remote, the remote adapter type, must be defined in the identityServer properties file.

Other properties are common, though identityReport uses a subset of the properties, properties used exclusively by identitySync are marked with a <I>. Properties are:

1. <I> ActionFileDelim1 is used to separate fields in the Action File. For readability this is set to |, but in production where readability is not required it can be changed, for example to <field>. See Transaction File for details.

2. <I> ActionFileDelim2 separator of attributes, for example <attr>

3. <I> ActionFileDelimList separator of attributes that have multiple values, for example <list>. The list deliminator is used in roles and group management and must be consistent in:

    - Adapter property RoleGroups used by ActiveDirectory, Unix and Windows.
    - Lookup tables for groups associated with roles. For example:

```
Dba,dbadmin<list>admin<list>mail
SQL,oracleadm<list>admin<list>mail
etc
```

    - Identity records defining groups.

4. <I> CreateAccountMessageTemplate a file containing a message template. The template has Metaverse attributes surrounded by %s, for example:

```
Message from IT
A new account %account% has been created on %resource% for %firstName% %lastName%
Password is %passwd%
```

   A special attribute %table% adds a table of all the attributes and their values.
   The template can be written in simple HTML with the first characters of the template <html. The "%" characters used by the attributes cannot be used elsewhere in the HTML template.

5. <I> CreateAccountMessageSubject is the email subject for messages containing new account information. The property can contain Metaverse attributes surrounded by %s, for example %resource% Account for %displayName%

6. <I> DebugSubject is the email subject for messages containing debug.txt

7. <I> DebugRecipients, the recipients of debug.txt in MailMode=0. Recipients separated by a ";". When blank the message is not sent.

8. <I> DebugUser, the value of the adapter attribute flagged displayName in a record in both identity and resource Metaverses. All the attribute names and values in the Metaverse at the end of the Extract and Transform will be displayed at the end of the debug.txt file when identitySync is run with LoggingLevel >= 2. See the displayName attribute flag. An alternative is to run identityReport using the same properties and schema files which will report all the accounts and identities in the Metaverse.

9. <I> DefaultRecipients, the email address that all account create email is sent to when MailMode = 1, or when no suitable email address is found for a manager in MailMode=0. Recipients separated by a ";".

10. <I> FromAddress, the email address that messages are sent from. For example amx@example.com or noreply@yourdomain.com. Used when MailMode is < 2.

11. <I> InfoRecipients the recipients of info.txt in MailMode=0. Recipients separated by a ";". When blank the message is not sent.

12. <I> InfoSubject is the email subject for messages containing info.txt
13. LoggingLevel, 1 – 5, 1 = summary, 5 maximum, default 3. The logging methodology is:

    1 debug log matches info. Minimal logging.

    2 configurations, debug user.

    3 details of configuration.

    4 identity and resource record names.

    5 details of identity and resource attributes and the match process.

14. LoggingType, 1 – 3, default 1:

    1 write to info and debug log files only.

    2 write to info and debug, with info to console.

    3 write to info and debug, with info and debug to console.

15. <I> MailMode:

    0 identitySync will send email to relevant recipients, for example a person's manager on create. MailMode 0 should only be used in production.

    1 send email to default Recipients, intended for testing.

    2 inhibit email. Default.

When the property LoggingLevel is >= 2 the email message for a user's account creation is reproduced in the debug file with the password obscured with asterisks.

16. <I> ManualSubject is the email subject for messages containing manual updates as defined by the Adapter Property ManualDo and ManualUndo.
17. <I> ManualRecipients the recipients of manual updates in MailMode=0 and when the Adapter property ManualDo and/or ManualUndo are defined. When blank manual updates are not sent.
18. <I> MaximumCreates Default adapter maximum, see adapter property of the same name. Default 0.
19. <I> MaximumDeletes as above
20. <i> MaximumDisables ditto
21. <I> MaximumUpdates, ditto
22. ProxyUser, for WSDL and Web API adapters JSON and SalesForce, the name of the account to authenticate with a web proxy. If the proxyUser is blank no web proxy is in use.
23. ProxyPasswdFile the password for the proxy user. If the proxyUser is non blank and the password is blank, the current user credentials are used.
24. <I> SMTPaccount, optional property, the username if connections to the SMTP mail server requires authentication.
25. <I> SMTPpasswd, optional property, the password if connections to the SMTP mail server requires authentication.
26. <I> SMTPport, default 25.
27. <I> SMTPserver the hostname of the SMTP mail server used to send mail for account creates, manual update files, info.txt and debug.txt. The SMTP server is used as an anonymous relay without authentication.
28. <I> SMTPssl optional property, default is true. False if the SMTP mail server does not support SSL connections and its digital certificate is invalid.

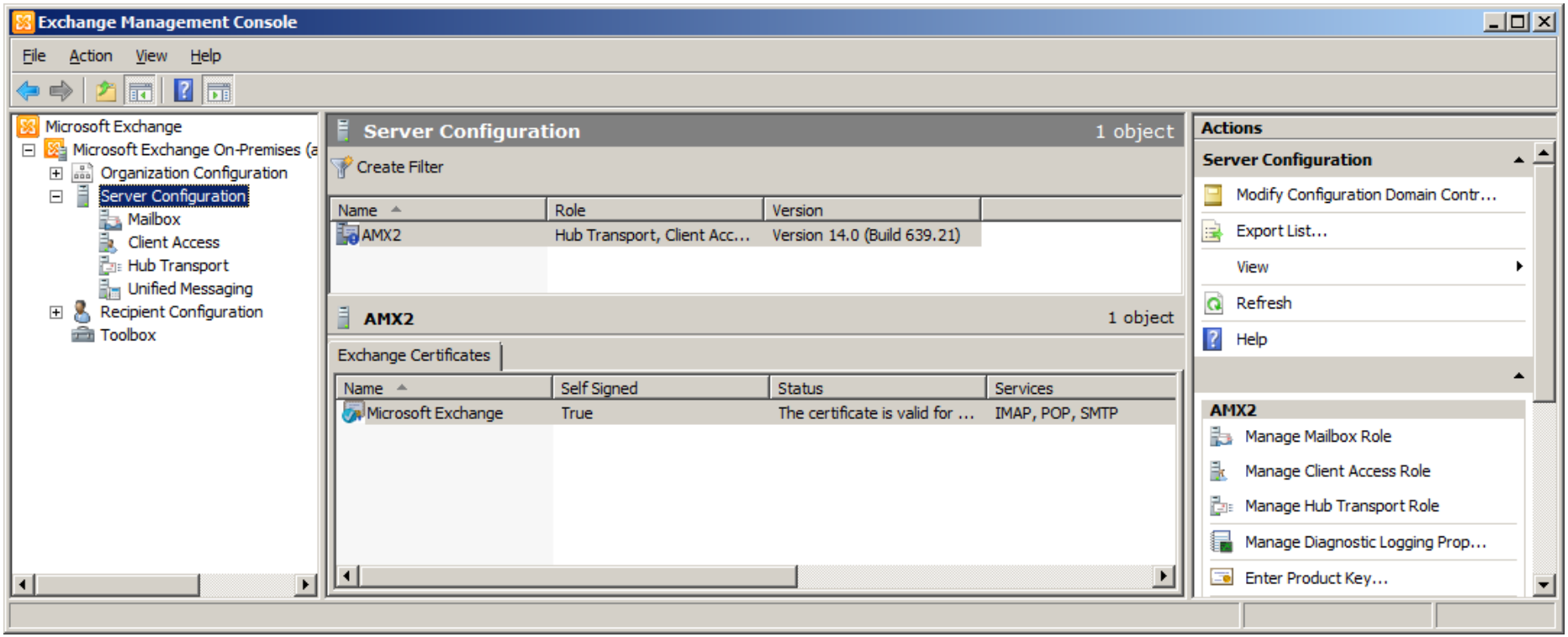
eMail

identitySync email uses .NET Mail which is SMTP with Explicit SSL which opens a connection in the clear and then sends StartTLS to begin encryption. It does not open the connection using encryption, see Microsoft documentation for full details.

The eMail sends the info and debug log when configured, so any mail error message is only available on the console. It is not in the logs.

Configuration for Exchange:

- SMTPaccount cannot be administrator, Exchange application event log shows “provided valid credentials, but is not authorized to use the server”.
- SMTPssl must be true, when false Exchange application event log shows “The SMTP server requires a secure connection or the client was not authenticated”.
- Check the certificate that SMTP is using in the Exchange Management Console / Server Configuration



Note that the certificate is self-signed and that it cannot be exported from the EMC. Use Certificate Management to do this, and then install it on the system running AMX.

Configuration for gmail. https://support.google.com/a/answer/176600?hl=en . The “lesssecure” security setting must be true, or an email will be received with the subject “Google Account: sign-in attempt blocked”. To understand and activate “lesssecure” visit https://support.google.com/accounts/answer/6010255?hl=en

- SMTPserver = smtp.gmail.com
- SMTPssl = true
- SMTPport = 587
- SMTPaccount = ****@gmail.com
- SMTPpasswd = GmailPasswd.txt

Adapter Properties

Adapter properties are prefixed by the adapter name and suffixed by the instance number starting at 1. For example ActiveDirectoryLoadMode1, LDAPManualDo1.

Adapter properties are designed for use by many instances, so properties of adapter instances > 1 inherit previous values. For example if ActiveDirectorySchema1 is defined its value will be used when ActiveDirectorySchema2 is not defined.

The property “Resource” must be defined for every instance for adapters. If the value of “Resource” is blank the instance is skipped.

There are also some adapter specific properties.

1. EmailChanges, the email address of one or more admins who will make manual changes to the resource. When blank the message is not sent and is only sent when the MailMode property is < 2. When MailMode is 1 it is sent to the default recipients. The message is the transaction file in a more readable form:

    Adapter instance, Load Mode, accountName
    accountName => attribute1 = new attribute value
    accountName => attribute2 = new attribute value
    etc etc

2. FilterAttribute, the metaverse attribute that is used to filter valid accounts in a resource. A blank FilterAttribute disables the filter. If FilterAttribute is defined, FilterValue must be defined too.

3. FilterValue, the metaverse value of the FilterAttribute will be tested against this value.

    If FilterValue is an integer, only records greater than or equal to this value will be included in the extract.

    If FilterValue is in the format <integerLower>-<integerUpper>, records greater than or equal to < integerLower> and less than or equal to <intergerUpper> will be included.

    Otherwise the FilterValue is treated as a string containing a list of values for an attribute defined in FilterAttribute, separated by FilterValueDelim that indicate records matching the value will be included. The match is case insensitive. For example:

        FilterAttribute = distinguishedName

        FilterValue = ou=EDN:ou=lon

        Will include records with ou=EDN or OU=lon in their distinguished names

    The values may be integers, they are treated as strings because of the ":" character, for example:

        FilterAttribute = GUID
        FilterValue = 1:2:3:4:5
        Is equivalent to FilterValue = 1-5 but computationally much less efficient.

If the first character of the FilterValue is a ! the records will be excluded rather than included from the extract. For example:

FilterAttribute = distinguishedName
FilterValue = !ou=GLA:ou=NEW
Will exclude records with ou=GLA or ou=NEW in their distinguished names

FilterAttribute = GUID
FilterValue = !1-249
Will exclude records with GUIDS in the range 1 to 249. This is equivalent to FilterValue = 250

4. FilterValueDelim, the delimiter used to split FilterValues. Default ":"

5. LoadMode, string containing the mode for loading resources in the form of one or more characters [M][CRUDD]. M (Manual) the load is inhibited for this instance and intended to be changed manually either using the manual updates or the EmailChanges message. CR (Create), U (Update), D (Disable), DD (Delete).

    a. CRUDD = Create, Update, Disable and Delete

    b. CRDD = Create and Delete only

    c. DD = Delete only

    d. CRUD = Create, Update and Disable only

    e. U = Update attributes only

    f. D = Disable only

No managed resource adapter ever deletes an account, they are moved aside so that a delete undo can always re-establish the account as it was before the change.

Delete will only update an account if it is currently disabled.

6. Lock, true, false or undefined. When true the passwd file is locked to the properties and schema files for the adapter, and neither can be changed without resetting the password in the password file. If the passwd file is not locked it could be used in a new properties file to perform an unauthorised extract or load. Another adapter instance with a different schema cannot use the locked password file, it will need its own password file. See the Security Tutorial document for further details.

7. ManualDo, the file name of the manual updates that can be used for this resource. The file may be emailed to the Generic property ManualRecipients if non-blank and MailMode < 2.

8. ManualUndo, the file name of the manual undo corresponding to ManualDo. This file may also be emailed to Manual Recipients.

9. MaxMatches, the maximum number of accounts in the managed resource that a single identity will match. Default = 1.

10. MaximumCreates, adapter specific maximum, defaults to global MaximumCreates property. Load phase of the adapter is skipped when the transaction file contains more this number of creates for the adapter.

11. MaximumDeletes, as above

12. MaximumDisables, ditto

13. MaximumUpdates, ditto

14. Name, the name that will be copied to the resource metaverse attribute when this is defined in the schema. If this is undefined resource name will be used. Must have a null staging attribute defined in the schema, for example:

    , resource

15. Passwd, the file name of a file containing the password used to authenticate the user defined in the User property with the server. The contents of the file are encrypted after the first time they are used and the clear text password deleted. The encryption uses DES. To set or change the password, enter the password on the first line.

    Use the Lock property to prevent the encrypted password being used in other unauthorised property files.

16. PasswordTemplate, the format of passwords generated for new accounts. Format is a series of character types which the generator uses to create random values:

    C = Upper case Consonant

    c = Lowercase Consonant

    V= Uppercase Vowel

    v= Lowercase vowel

    n= Numeric

    The set of consonants and numeric excludes characters that can be misread such as 1,l,0,O.

    For example CvcnCvcn creates a password like Pek3Bat7

17. RandomString, the character set used by the %random% attribute in accountNameTemplates. Default is:

        ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

18. Resource, the file name or hostname of the resource. File names are used for CSV, Excel, LDIF and Textfile.

19. Schema, file name of the resource's schema file.

20. User, the administrative account name used with Passwd to authenticate with the Server. User can be:

    a. Blank, in which case in a Windows Domain environment the account that is executing the application will be used on the target server.

    b. Defined in the User property, in which case the Passwd property contains the name of a file containing the password. In Windows environments the password can contain the domain component, for example corp\engela.

Adapters

All adapters work remotely, there is no installation of services on target instances. In cases where a firewall prevents access identityServer can be used with a single port. The sequence that the adapters are evaluated is:

1. Identity Adapters
2. Resource Adapters

Within each category the sequence of adapters is as found in the properties file. Following that it is in the numerical order of each adapter instance

The sequence is significant because resource adapters like the Active Directory will update blank account names in the identity mataverse so that these account names can be used in other managed resource adapters such as Windows Local account names or HomeShare directory names. See schema for details.

ActiveDirectory

The Active Directory can be used as an identity source as well as a managed resource. As a Managed Resource it can manage accounts or contacts. The Active Directory can also be used as the definition of Role templates. The adapter names are:

- “ActiveDirectoryIdentity” for an identity source.
- “ActiveDirectory” for a managed resource. Accounts that do not have an associated identity record are moved to the “DeletedContainer” as defined in the properties. See below.
- “ActiveDirectoryRole” for the definition of role templates. See below.

Accounts that have no matching identities are considered deleted, they are moved to the Deleted OU as defined in the properties. Accounts are never deleted unless a create is being undone. This preserves the reversibility of identitySync.

1. AccountContainer, the DN of the account container or the root of the account containers. For example ou=accounts,dc=corp,dc=example,dc=com, or if defaulted or blank will extract all the accounts from all the containers and have to use the FilterAttribute and FilterValue to select the relevant

accounts. Using a specific AccountContainer will improve performance, however if there are accounts that are not part of the AccountContainer these will not be extracted and will not be tested when creating a unique account name.

If the AccountContainer is or includes CN=Users, some well-known builtin accounts are automatically filtered out. These are:

- Administrator
- krbgt
- SystemMailbox
- FederatedEmail
- DiscoverySearchMailbox

The any other system accounts must be added to the property FilterValue. Synchronising user accounts in the CN=Users container is not recommended, consider moving managed accounts to an OU.

2. DeletedContainer, the container that deleted accounts will be moved to for the Active Directory resource type. For example:

   OU=Deleted,OU=accounts,DC=corp,DC=example,DC=com

3. ExtractMode, Single, Add, Merge. Add and Merge is only available for sources of identity “ActiveDirectoryIdentity” resources and the default is “Add”.

   Merge is performed after other identity sources are added, when an existing identity record cannot be found the merge is a no-op. The Metaverse and the Active Directory must have a common attribute for the merge to be successful and this must be defined in the Active Directory Schema with the “join” attribute flag. Merge is useful in the following example situations:

   a. An HR system may provide identity attributes but not an existing person’s account name. A person’s Active Directory account name may be required to identify the matching account in HomeShares, Databases, Unix systems or in the Windows Local SAM.

When IdentitySync has an Active Directory defined as an Managed Resource it will merge some attributes into the Identity Metaverse by default and defining an ActiveDirectory Identity adapter is not necessary. These attributes are accountName, DN and mail.

For managed resources the default ExtractMode is “Single”, where each instance of the resource is analysed individually. “Add” mode is intended for situations where different OUs are managed by different parts of the organisation and consequently have different formats. In “Add” mode, the accounts of this instance are appended to the previous instance. To use “Add” mode, set the ExtractMode property for first instance to “Add” and use the default mechanism of properties so that ExtractMode is also “Add” for the subsequent instances. Use FilterAttribute and FilterValue to select the OU for each extract.

4. ImageSize, the width in pixels of the thumbnailPhoto image. Default 104. Aspect ratio is maintained for the height.
5. Name must be the NETBIOS name of the domain
6. NewAccountContainer, the DN of the container for new accounts when the CN and not the distinguishedName of the account is defined. See ActiveDirectory schema for more information. Defaults to the AccountContainer.
7. RoleAttribute, the attribute names in the Active Directory that define the roles. For example roles are defined for Location and Title for adapter 1:

```
ActiveDirectoryRoleAttribute1-1 = Location
ActiveDirectoryRoleAttribute1-2 = Title
```

8. RoleContainer, the container in the ActiveDirectory that contains the role model for each role. When blank role processing is disabled. For instance:

```
OU=roles,OU=AMX,DC=corp,DC=example,DC=com
```

9. RoleDeleteDelay, the default is 7. This defines the hysteresis for group removal for this ActiveDirectory instance in days. For example a DeleteDelay of 2 will add groups to users having the role, and mark for removal any groups that the user has that are not part of the new role 2 days later. Only managed groups are removed, that is groups that are in any of the Active Directory role models or Identity based Roles.
10. RoleMode = strict or loose, the default is loose. See Roles for details.
11. SearchFilter, the default is (&(objectCategory=Person)(objectClass=user)), to extract and manage Contacts rather than users set SearchFilter to (objectClass=Contact).

ActiveDirectory Containers

The extract builds a table of all the account names in AccountContainer, so that a unique account name can be created for new accounts. The AccountContainer must be the parent container of the DeletedContainer and the NewAccountContainer. For example:

AccountContainer = DC=corp,dc=example,dc=com
DeletedContainer = OU=deleted, DC=corp,dc=example,dc=com
NewAccountContainer = OU=accounts, DC=corp,dc=example,dc=com

If a structure like below is used:

AccountContainer = OU=accounts,DC=corp,dc=example,dc=com
DeletedContainer = OU=deleted,OU=accounts, DC=corp,dc=example,dc=com
NewAccountContainer = OU=accounts, DC=corp,dc=example,dc=com

The domain may have accounts in CN=Users which will not be extracted and not added to the table of account names. This is suitable for situations where the managed account names have a completely different format, as defined in the property AccountNameFormat, or identitySync is configured never to create accounts.

ActiveDirectory Manager Attribute

The ActiveDirectory Manager attribute is the distinguishedName of the person's manager. In order for identitySync to update a person's manager, the identity metaverse must also have the distinguishedName of the person's manager. This is not usually available from an identity source, where a person's manager may be defined by the manager's employee number or full name.

identitySync handles this in the following way.

The Identity attribute containing the identity of a person's manager is flagged with ManagerID.

The Identity attribute in the manager's record that contains ManagerID is flagged with ManagerJoin.

For example a person's identity record has the manager attribute and it contains the employeeID of the person's manager. The Identity schema would contain:

```
Manager,managerEmployeeID;ManagerID
employeeID,employeeID;ManagerJoin
```

If the person's Identity record manager attribute is the manager's full name, the Identity schema would contain:

```
Manager,managerFullName;ManagerID
FullName,FullName;ManagerJoin
```

In situations where identities are extracted from multiple sources, ManagerID and ManagerJoin can be defined differently in each schema. For example the first schema can use the first example above and the second schema can use the second example.

ManagerID and ManagerJoin allows the identity record of a person's manager to be found by identitySync. Given the person's manager's record, identitySync updates the person's Identity attributes flagged with ManagerName, and IsaManager, and also to send email to a person's manager when their account is created. If the Identity Metaverse does not contain the value of the manager's distinguishedName or email it is copied from the ActiveDirectory. The Identity schema would then contain:

```
,Manager;ManagerName
,ManagerRole;IsaManager
```

The ActiveDirectory schema would contain:

```
Manager,Manager
```

identitySync will compare the Manager attributes in the Metaverse, and update the ActiveDirectory Manager attribute where there is a difference.

ActiveDirectory Groups

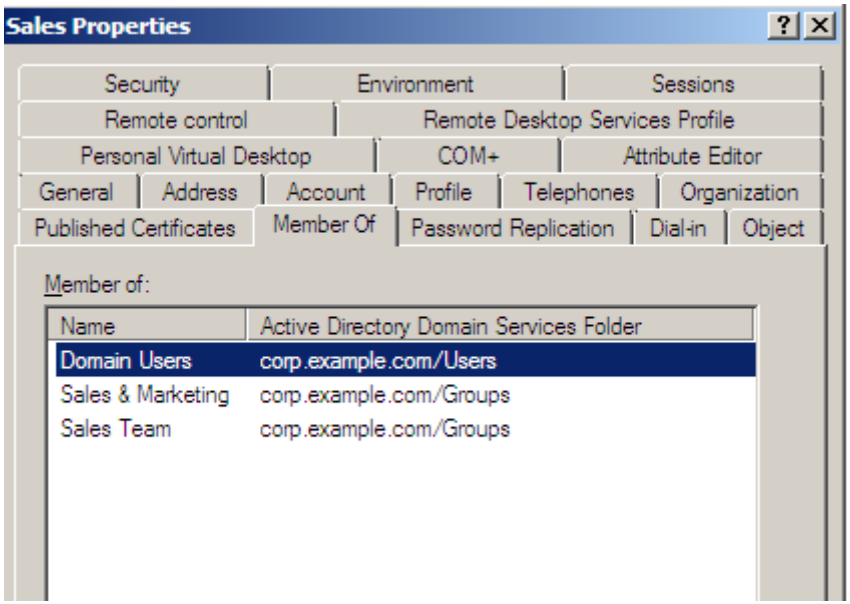
identitySync only manages groups that are "Managed" by virtue of being defined in the identity source, or as being part of a role template.

Groups defined in an identity source are added, and may be removed if the "RoleMode" is "strict" at a later date defined by the property "RoleDeleteDelay". Groups that a person is a memberOf that are not managed, are neither added nor deleted.

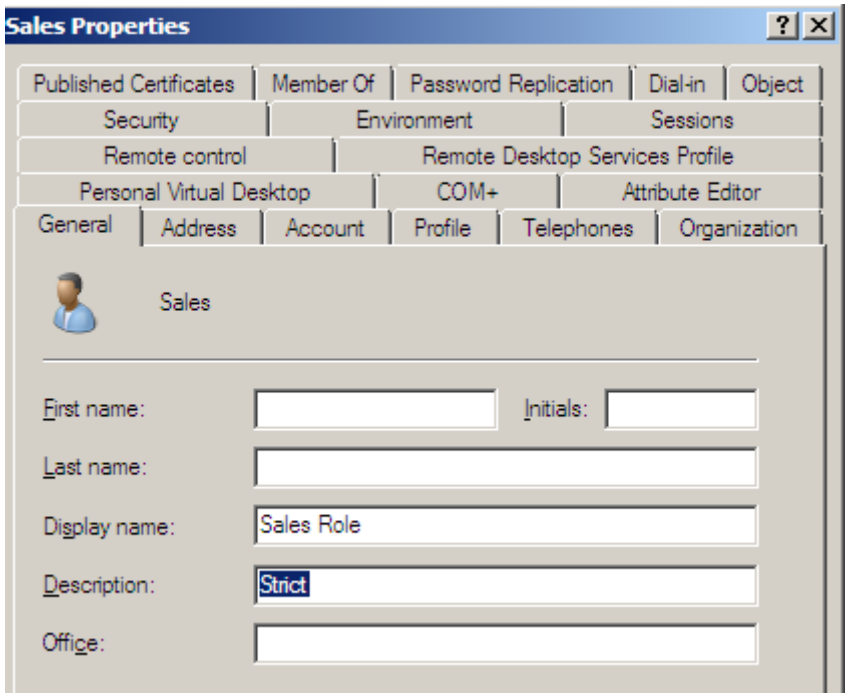
ActiveDirectory Role Templates

The ActiveDirectory implements Roles by using either Role Templates or Identity based Roles. Never use both. Roles are managed when the Schema Attribute memberOf is defined. Role templates are used when the ActiveDirectoryRoleResource is defined in the Active Directory properties file.

A person's identity attribute can be mapped to a role template in the ActiveDirectory to add and remove them from groups associated uniquely with the role. A role template is a user account with the name of the role that is a member of the groups associated with the role.



The role template also defines the mode that is used during a change of role. Modes are loose and strict, and strict allows hysteresis (see Roles). The role delay is defines in the RoleDeleteDelay property in the identitySync.Properties file. If a group is defined in both a strict and loose role template it is managed in the loose mode, that is it is added but never removed.



1. The role attribute is the attribute name in the Metaverse that defines the role template. Multiple role attributes can defined in the identitySync.Properties file. For example the attributes can be:

    a. A job title = title

    b. A location = location

    c. A department = department

    The properties file for adapter 1 would be:

    ```
    ActiveDirectoryRoleAttribute1-1 = title

    ActiveDirectoryRoleAttribute1-2 = location
    ```

```
ActiveDirectoryRoleAttribute1-3 = department
```

2. The location of the OU which will be contains the role templates is defined in identitySync.Properties file.

```
ActiveDirectoryRoleContainer1 = OU=roles,OU=AMX,DC=corp,DC=example,DC=com
```

The role container must contain Template Users with names matching the values of the Role Attributes. For example if the title attribute has values, sales, marketing, finance etc, there must be Template Users with the same names. A non-fatal error will occur if a value has no matching template.

The set of groups that are managed by identitySync are only those defined in the Template Users. Only Managed groups are synchronised by identitySync.

CSV

An adapter that reads a comma separated file as a source of identities or as an extract of users from a managed resource. The resource adapter does not provide manual updates.

The adapter names are:

- “CSVIdentity” for a source of identities
- “CSV” for an managed resource
- “CSVRoles”

The CSV format is fields separated by a delimiter, using a “ to surround the field when it contains the delimiter. For example:

```
distinguishedName,givenName,firstName etc
"cn=alban wilson,ou=lon,ou=accounts,dc=corp,dc=example,dc=com",alban,wilson, etc
```

The CSV file may contain a header defining the staging attribute name of the column, or no header in which case the staging attribute name is the column number starting from 0.

Additional properties

1. Delim, the separator, default ",".

2. ExtractMode, add, merge. The default mode is to add records to the Metaverse. ExtractMode Merge for the CSVIdentity adapter updates attributes to the identity Metaverse for an existing record. The Metaverse and the CSV file must have a common attribute for the merge to be successful and this must be defined in the CSVIdentity Schema with the "join" attribute flag. Merge is useful in the following example situations:

    a. An organisation's telephone directory is managed separately and there is a requirement to add it to the ActiveDirectory. Extract the Identities and then merge the telephone directory into the Identity Metaverse, and then synchronise the resultant identities with the accounts in the ActiveDirectory.

    b. Where persons have the same first and last names, a manually created unique email address or CN is required and when this cannot be done in the authoritative source of identities a merge can be used. For example, a Metaverse attribute "Discriminator" can be defined in the Identity Schema, the value is by default blank and updated for specific identity records by a merge file. The merge file could contain a header with the attribute names, followed by data:

        ```
        employeeID,discriminator
        75151,(IT)
        8182,(HR)
        etc
        ```

        The schema for the merge adapter instance:

        ```
        employeeID,employeeID;join
        discriminator,discriminator
        ,CN;concat:CN=%firstName% %lastName%%discriminator%
        ```

        This will create a CN with a suffix of (IT) for identity record 75151 overwriting a non-unique CN. Uniqueness can be checked by adding the attribute flag "unique".

    c. The organisation uses a lookup table to match accounts in some managed resources. The lookup table can be merged with the identities and used to synchronise these resources. This is a short term solution, eventually the managed resource should be updated or tattooed so that there is a permanent marker identifying the owner of every account in the managed resource. Lookup tables are difficult to maintain.

3. Header, false there is no header. The staging attributes in the schema specify the column number starting from 0.

Header true is the default and the staging attribute names are found in the header.

Database

Databases can be used as identity sources as well as managed resources. The adapter names are:

- "DatabaseIdentity" for an identity source.
- "Database" for a managed resource. Accounts are disabled by locking them, and re-enabled by unlocking them. The adapter does not implement delete, and only ever deletes (drops) accounts when undoing a create.

Additional properties:

1. AccountContainer, the database and table containing the accounts. This table will be used for both extract and load for the managed resources. The adapter assembles a query for an extract using the schema and adapter properties.

   ```
   Select <Staging Attributes> from <AccountContainer | SelectFrom>
   ```

   This allows multiple tables and joins, for example account Container for MSSQL user accounts and logins:

   ```
   DatabaseAccountContainer1 = sys.server_principals spr left join sys.server_role_members srm on
   spr.principal_id = srm.member_principal_id join sys.server_principals spr2 on srm.role_principal_id
   = spr2.principal_id where spr.type in ('U','S')
   ```

   Oracle if auditing has been enabled – see AMX installation guide:

   ```
   DatabaseAccountContainer1 = sys.dba_users u left join (select username,max(timestamp) as lastlogon
   from sys.dba_audit_session a group by username) a on u.username = a.username left join
   sys.dba_role_privs on u.username = grantee
   ```

2. ManualDo, the file name of the manual updates that can be used for this resource. This contains a copy of the SQL commands sent to the database in do/redo mode.

3. SelectFrom, used by the DatabaseIdentity adapter only, the database and table containing the identities. This table is used for extract of identity resources. Defaults to AccountContainer.
4. FilterAttribute: a metaverse attribute with integer values such as uid
5. FilterValue: an integer with the format of FilterValue is <begin>[-<end>] where <begin> is the first integer in range, and optionally <end> is the last. For example FilterValue = 1000 will extract accounts with a uid 1000 and greater, FilterValue = 1000-59999 will extract accounts with uids between 1000 and 59999.
6. Resource: the URL of the database in the format <DNS name or ip address of the server>:<port>/<SID>. Default port 1521, SID = orcl

   For example:

   ```
   DatabaseResource1 = db6.corp.example.com:1521/orcl
   ```

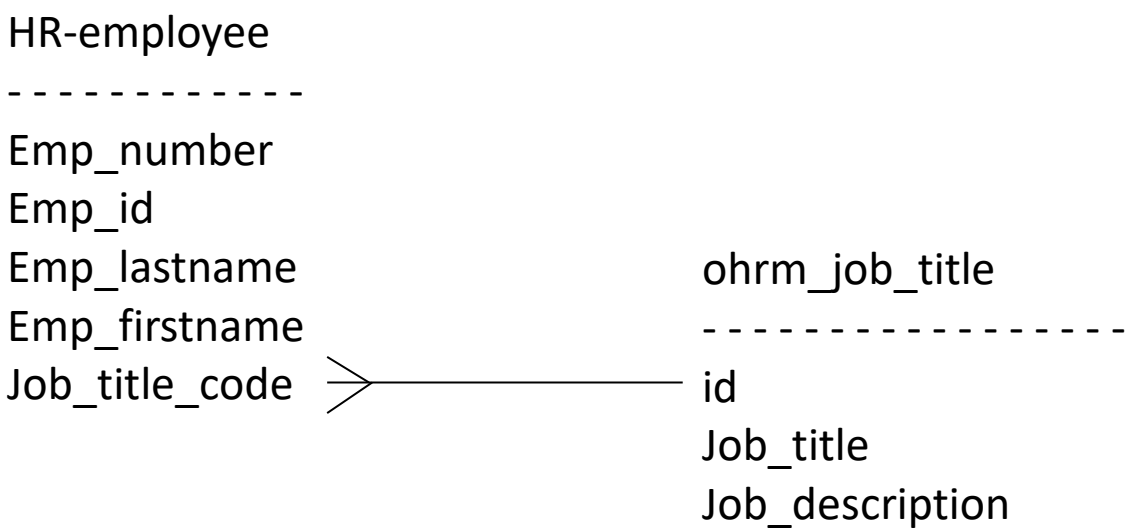
   Using defaults all these are canonic:

   ```
   DatabaseResource2 = db6.corp.example.com:1521

   DatabaseResource2 = db6.corp.example.com/orcl

   DatabaseResource2 = db6.corp.example.com
   ```

7. RoleDeleteDelay, the number of days delay for an account to be removed from a group when RoleMode=strict. Default is 7. See hysteresis for details.
8. RoleGroups, Managed groups or database roles are those groups found in the identity source only.
9. RoleMode, loose or strict. See Managed Groups

Database Subqueries and Joins

Consider a database with 2 tables such as:



Using a subquery or inner select the job_title can be selected using:

```
select employee_id,emp_lastname,emp_firstname,
(select job_title from ohrm_job_title as job where job_title_code = job.id) as job_title
from hr-employee emp
```

The equivalent join is

```
select employee_id,emp_lastname,emp_firstname, job.job_title
from hr-employee emp
left join ohrm_job_title as job on job_title_code = job.id
```

When implementing this in identitySync as a subquery, the subquery is the staging attribute. An equivalent AMX schema is:

```
(select job_title from ohrm_job_title as job where job_title_code = job.id),job
```

The join is defined in the AMX property SelectFrom, for example

```
DatabaseIdentitySelectFrom1 = hr-employee emp left join ohrm_job_title on job_title_code = id
```

Left join is used so that all the records from hr-employee are reported whether or not a matching record in ohrm_job_title. The schema is then

```
job_title,job
```

Multiple left joins can be defined in the SelectFrom property. More information concerning this can be found in the HRMS tutorials.

Database Images

Images are stored as blobs in databases as identity sources. They must have the attribute flag syncFile. Currently only MySQL reads blobs. Blobs require the blob size as a Metaverse attribute in the database schema, with a name prefixed by the blob name and appended with “Size”. For example MySQL the function OCTET_LENGTH AMX Schema is:

```
epic_picture,imageFile;syncFile
OCTET_LENGTH(epic_picture),imageFileSize
```

Best practice is to store the blob size in the database, so when this is available an equivalent is:

```
epic_picture,imageFile;syncFile
epic_size,imageFileSize
```

Excel

A source of identities, the adapter name is “Excel”. The adapter can read a variety of spreadsheet styles, limitations are:

- Header containing the column names must be in the first 10 rows

- Maximum of 30 columns, columns “a” through “ad”.
- Blank, or rows containing too many blanks are skipped based on the property MaxBlankAttributeCount. This allows the worksheet to have embedded notes and summaries, and the adapter will extract just the identity records.
- An adapter instance reads one worksheet. If the identities are contained on multiple worksheets of the same workbook, the extract should be run as multiple adapter instances, one for each worksheet.
- The worksheet to be extracted is identified by its sheet number, rather than its name.

Additional properties:

1. MaxBlankAttributeCount. The row will be skipped if it contains more blank columns than the value in this property. Default is number of columns multiplied by 2 and then divided by 3.
2. Passwd, the name of the file containing the password. Required if the Excel file is encrypted. Spreadsheets that are confidential can have the confidential information filtered using macros and then written to a new public spreadsheet. For details see the AMXPublicExcel document.
3. Worksheet, the worksheet number containing the data starting with the leftmost as 1.

Exchange

A Microsoft Exchange managed resource adapter, the adapter name is “Exchange”. It uses remote PowerShell to manage mailboxes for Exchange Servers Exchange 2010 and later. The manualDo and ManualUndo files contain the Powershell commands used in the remote connection during the load phase.

Mailboxes are disabled by setting HiddenFromAddressListsEnabled to true.

The adapter does not implement delete for mailboxes that have been removed from the authoritative source of identities. The adapter skips any mailboxes of users that the Active Directory adapter has moved to the deleted container.

The account used in ExchangeUser must be a member of the Exchange Group “Organization Management. If ExchangeUser is blank the currently logged on account is used.

Extract uses the Exchange Cmdlet Get-Mailbox:

```
Get-Mailbox [-OrganizationalUnit <AccountContainer>] | Select-Object <List of Staging Attributes>
```

A disabled account has hiddenFromAddressListsEnabled set to true and if the Out of Office Message OOMessage is not blank the recipient will have the message set.

When undoing a create, the mailbox is removed but the account is not, using the Exchange Cmdlet:

```
Disable-Mailbox -Confirm:$false –Identity
```

Additional properties:

1. AccountContainer, the organisational unit in the Active Directory containing the accounts. Format is Active Directory object format, for example:

   ```
   corp.example.com/accounts
   ```

2. DeletedContainer, the container that the Active Directory adapter uses for deleted accounts. Mailbox Identities in this container will be skipped. The format is Active Directory object format, for example:

   ```
   corp.example.com/accounts/Deleted
   ```

3. Domain, the Exchange Identity value of the recipient is of the form domain\accountName. The default value is the domain of the account used to administer Exchange in the property ExchangeUser.

4. ExtractMode, Add, Single. The default mode is Add, the recipient mailboxes of multiple Exchange servers are added or appended and then analysed.

5. MailboxDatabase, the mailbox database for new mailboxes.

6. OOMessage, the Out of Office Message used for External messages when the recipient is disabled. The message cannot contain html.

7. OOInternalMessage, the Out of Office Message used for Internal messages. When blank the OOMessage text is used.

8. Resource is the URI of the remote PowerShell exposed by Exchange 2010 and later. For example:

```
https://AMX2.corp.example.com/Powershell
```

HomeShare

A Windows HomeShare managed resource adapter, the adapter name is “HomeShare”. It uses WMI to create, update, disable and move shares and set permissions of shares.

Homeshare follows Microsoft Best Practices, NTFS permissions are used to restrict access rather than Share permissions. The reason behind this is Share permissions are reset when a share is moved or re-created, NTFS are not.

Homeshare uses a specific directory structure that may not be present or suitable in all instances:

- User shares are subdirectories in the AccountContainer as defined in the Properties.
- User directories are shared, the share name matches the AccountName.
- Permissions are applied to the directory, giving the user access to the directory and files. Permissions are defined in the Permission property.
- Permission inheritance is set to 0x3:
    - OBJECT_INHERIT_ACE (1 (0x1)) Noncontainer child objects inherit the ACE as an effective ACE. For child objects that are containers, the ACE is inherited as an inherit-only ACE unless the NO_PROPAGATE_INHERIT_ACE bit flag is also set.
    - CONTAINER_INHERIT_ACE (2 (0x2)). Child objects that are containers, such as directories, inherit the ACE as an effective ACE.
- When a User share is disabled it is moved to the InactiveContainer as defined in the properties.
- When a User share is logically deleted, it is moved to the DeletedContainer as defined in the properties.
- User shares in the deleted container are un-shared.

Example Structure

E:\ShareE\ CraigA	- shared as CraigA
E:\ShareE\deleted	- the deleted container, as defined in Properties as DeletedContainer.
E:\ShareE\deleted\ JonesB	- not shared
E:\ShareE\engela	- shared as engela
E:\ShareE\inactive	- the Inactive container, as defined in Properties as InactiveContainer.
E:\ShareE\inactive\ BurnsB	- shared as BurnsB
E:\ShareE\inactive\ MilneE	- shared as MilneE
E:\ShareE\ O'BrianS	- shared as O'BrianS
E:\ShareE\ StevensonA	- shared as StevensonA
E:\ShareE\WoodB	- shared as WoodB

To Extract the Homeshares from this example HomeShareResource1 = AMX3.corp.example.com

Note that the Homeshare adapter is able to escape "'" in O'BrianS but WMI fails to handle permissions properly https://msdn.microsoft.com/en-us/library/aa394054%28v=vs.85%29.aspx so the attribute Trustee will be empty and an error message will be written to the debug file. Directory naming convention is described in https://msdn.microsoft.com/en-gb/library/windows/desktop/aa365247%28v=vs.85%29.aspx

identitySync combines each instance into the MetaVerse before synchronising, the ExtractMode is always "add". New shares are always created on the last instance defined in the properties file.

The adapter never deletes a share unless it is undoing a create and then only within 24 hours of the create.

Additional properties:

1. AccountContainer the directory where new shares should be created. For example E: or D:\Shares. Cannot be blank.
2. DeletedContainer is also a subdirectory in AccountContainer. Deleted Homeshares are moved to this subdirectory. For example "inactive\deleted".
3. FilterValueDelim, the default is ":" which will need to be changed with a FilterValue like C:\Shares. Set to "|" for example so that FilterValues for multiple directories are F:\Shares|G:\Shares|H:\Shares

4. InactiveContainer in the form of the name of the subdirectory. Inactive HomeShares are moved to this subdirectory. The move is a pure move not a copy and delete for performance reasons on large shares, so the container must be on each directory of shares. For example “inactive”, where a share E:\Shares\craigc will be moved to E:\Shares\inactive\craigc

5. Permission. The ACE access mask which is the sum of the individual rights in Hex. The default is 0x1301bf which corresponds to Modify. 0x1200A9 is Read and 0x1F01FF is Full.

    - FILE_READ_DATA (file) or FILE_LIST_DIRECTORY (directory) 0x1. Grants the right to read data from the file. For a directory, this value grants the right to list the contents of the directory.
    - FILE_WRITE_DATA (file) or FILE_ADD_FILE (directory) 0x2 Grants the right to write data to the file. For a directory, this value grants the right to create a file in the directory.
    - FILE_APPEND_DATA (file) or FILE_ADD_SUBDIRECTORY (directory) 0x4 Grants the right to append data to the file. For a directory, this value grants the right to create a subdirectory.
    - FILE_READ_EA 0x8 Grants the right to read extended attributes.
    - FILE_WRITE_EA 0x10 Grants the right to write extended attributes.
    - FILE_EXECUTE (file) or FILE_TRAVERSE (directory) 0x20 Grants the right to execute a file. For a directory, the directory can be traversed.
    - FILE_DELETE_CHILD 0x40 Grants the right to delete a directory and all the files it contains (its children), even if the files are read-only.
    - FILE_READ_ATTRIBUTES 0x80 Grants the right to read file attributes.
    - FILE_WRITE_ATTRIBUTES 0x100 Grants the right to change file attributes.
    - DELETE 0x10000 Grants delete access.
    - READ_CONTROL 0x20000 Grants read access to the security descriptor and owner.
    - WRITE_DAC 0x40000 Grants write access to the discretionary access control list (ACL).
    - WRITE_OWNER 0x80000 Assigns the write owner.
    - SYNCHRONIZE 0x100000 Synchronizes access and allows a process to wait for an object to enter the signaled state.

Image File

A source of identities, extracted from a directory containing images. The adapter name is “ImageFileIdentity”. Additional properties:

1. Resource, the name of the directory containing the images.

2. FilterAttribute, the metaverse attribute that is used to filter image files. Images are extracted when part the value of the FilterAttribute matches one of the FilterValues. The match is case insensitive.

3. FilterValue, a list of values for an attribute defined in FilterAttribute, separated by ":" that indicate records to be included. For example:

    FilterAttribute = ImageFile

    FilterValue = .jpeg:.gif

The current version of identitySync can only update images in the Active Directory.

When using the Image File adapter, only ever use LoadMode U, update. The information in the images files is not sufficient to create or disable Active Directory accounts.

JSON

The JSON adapter operates on a file containing a JSON record or a REST (Representational State Transfer) web site as an identity or managed resource. JSON files have a simpler format than XML but are otherwise similar. Most cloud applications will respond to a properly formed request with a JSON response containing the user account information. SCIM (System for Cross-domain Identity Management) also uses JSON. REST is an architectural style rather than a standard and there are a large number of variations. REST uses web service requests (POST = Create, PUT = Update, Delete = delete). Requests require an access token, which is usually obtained from the AuthenticationURL.

If a web proxy is used see the proxyUser and proxyPasswdfile properties to authenticate with the proxy server.

The adapter is at present only able to **Extract** accounts from a limited number of systems. Create, Update and De-activate are not available in this release. Adapter names are:

- JSONIdentity
- JSON

Properties:

1. AuthenticationData: a comma separated string containing the authentication data used to get the token for subsequent access. For example Twitter:

   ```
   accept:application/json,"Authorization: Basic %password%","grant_type=client_credentials"
   ```

   Note that the password is obtained from the encrypted password file and substituted for %password%. Items can be enclosed in “s if the value includes the delimiter “,”. The data is split using the “,” delimiter. Items are identified as part of the request header or the message body using the characters “:” and “=”. The debug file shows how the data is interpreted when the debug level > 1. For example:

   ```
   JSON Extract AuthenticationData Header  accept : application/json

   JSON Extract AuthenticationData Header  Authorization : Basic %password%

   JSON Extract AuthenticationData Message grant_type=client_credentials
   ```

   Messages like username=admin&password=password cannot use ”?”, it must be replaced with “,” in the AuthenticationData.

2. AuthenticationRefreshTokenName: Default is none. If this property is defined it is used as the name of the refresh token in the JSON object and the value is used to update the encrypted value in the Passwd file.

3. AuthenticationRequestType: POST or GET. Default GET.

4. AuthenticationTokenName: the name of the token in the authentication JSON object. For example for SalesForce this is “access_token”

5. AuthenticationTokenPrefix: the string added to the authentication token when submitting it to the resource. For example for SalesForce this is Bearer

6. AuthenticationURL1: the URL used to send the authentication request. A blank value skips the authentication step and uses a persistent token in the password file.

7. ExtractData: a comma separated string containing the data used to GET the extract JSON response. The ExtractData string has the same format as AuthenticationData. For example Twitter:

```
accept:application/json,"Authorization: Bearer %password%","grant_type=client_credentials"
```

If the ExtractData is a JSON string it is not treated as a comma separated string and is uploaded to the API endpoint as-is. For example PeopleHR:

```
{"APIKey": "e86dcfea.etc","Action": "GetAllEmployeeDetail","IncludeLeavers":"true"}
```

8. ExtractSignature the string that will be signed. Substitutes %nonce% if present in the string. For example SageOne:

   GET&https%3A%2F%2Fapi.sageone.com%2F%2Faccounts%2Fv1%2Fcontacts&&%nonce%

9. ExtractSigningAlgorithm HMACSHA1 = HS1, HMACSHA256 = HS256, RS256.

10. ExtractSigningKey, substitutes %password% and %token% when present in the key

11. Name the name that will be copied to the resource metaverse attribute when this is defined in the schema.

12. Passwd: the name of the password file, the encrypted value it contains will be used to substitute %password% in the authenticationData, ExtractData, or if a JWT assertion is used, this is the Certificate's Private Key.

13. Preamble: the number of levels of "{" in the pre-amble to the first attribute of the member, starting from 0. The default is 2. For example The Windows Azure Directory:

```
{
  "users": [
        {
        "id":                   35436,
        "name":                 "James A. Rosen",
        "created_at":           "2009-07-20T22:55:29Z",
        "updated_at":           "2011-05-05T10:38:52Z",
        "time_zone":            "Copenhagen",
        "email":                "james@example.com",
        "phone":                "555-123-8901",
        "locale":               "en-US",
        "locale_id":            1,
        "organization_id":      57542,
```

```
"photo": {
    "id":           928374,
    "name":         "james rosen.png",
    "content_url":  "https://zendesk.example.com/photos/james rosen.png",
    "content_type": "image/png",
    "size":         166144
}
"role":                 "end-user",
etc
```

In this example level 0 contains “users”: and level 1 has the member attribute id: 35436. Preamble = 1. The schema must refer to the attributes at this level such as name, created at, email, phone, role etc. Attributes at higher levels are flattened, for example in the json above “photo” is flattened into:

```
"organization_id":    57542,
"photo.id":           928374,
"photo.name":         "james rosen.png",
"photo.content.url":  "https://zendesk.example.com/photos/james rosen.png",
"photo.content.type": "image/png",
"photo.size":         166144
```

The schema references these as photo.id, photo.name, photo.content.url etc.

14. RequestType, GET or POST. RESTfull should be GET which is the default.

15. Resource, the name of the file containing the JSON data or the URL of a REST API end-point. For example:

    File://json.txt

    https://www.concursolutions.com/api/v3.0/common/users

    The URL may have parameters for example:

    https://na9.salesforce.com/services/data/v20.0/query/?q=SELECT+email+,+name+from+User

16. Schema, the name of the schema file.

JWT Assertions

A JWT consists of a Header + Claim + Signature. The items are separated by a ".". For example:

{"typ":"JWT","alg":"RS256"}.{"iss":"account-1@axntinteractive.iam.gserviceaccount.com","scope":"https://www.googleapis.com/auth/analytics.readonly","aud":"%authenticationURL%","iat":"%issuedAt%","exp":"%expiresAt%"}

1. JWTClaim: in JSON format, for example:

   {"iss":"account-1@axntinteractive.iam.gserviceaccount.com","scope":"https://www.googleapis.com/auth/analytics.readonly","aud":"%authenticationURL%","iat":"%issuedAt%","exp":"%expiresAt%"}

   The variables %authenticationURL% % issuedAt%, and % expiresAt% are substituited. When debug > 1 the Claim is shown after the substitutions. Times are in Unix Epoch time.

2. JWTHeader: Typically {"typ":"JWT","alg":"RS256"}
   If the header is not defined JWT is not used.

3. JWTLifetime: requested lifetime of the token in seconds. Used to create the %expiresAt% variable in the claim.

4. JWTSigningAlgorithm: RS256 or HS256 must match the value in the JWT Header. The algorithm is used to sign the claim.

5. Certificate: the filename of the certificate in Public-Key Cryptography Standard p12 format used for RS256.

6. Passwd: Name of the file containing the certificate private key.

Process

The adapter follows the following process:

If the Resource is a file then:
    Read file
Else
    If JWTheader is defined then
        Build the JWT
        Sign JWT with RS256 or HS256 algorithm as defined in JWTSigningAlgorithm
    End
    If AuthenticatioURL is defined then
        Build Authentication request header and data from AuthenticationData substituting %JWT%, %password%, %authenticationURL%, %issuedAt%, %expiresAt%
        Send request to AuthenticatioURL
        Read response and get the value of access token from the attribute AuthenticationTokenName
        If AuthenticationRefreshTokenName is defined then
            Get the value of refresh token from the attribute AuthenticationRefreshTokenName in the JSON string.
            Replace the encrypted value in Passwd with the refresh token.
        End
    End
    Build data request header and data from ExtractData substituiting %password% from the password file and %token% from the response from the authenticator.
    Send the request to Resource
    Read response into a string

End
Extract the attributes and their values from the JSON string.

LDAP

An LDAP adapter for a managed resource, the adapter name is “LDAP”.

The format of the server parameter is a URL such as

LdapResource1 = ldap://192.168.121.55:389

If the port is omitted it defaults to 389 for ldap: and 636 for ldaps:

When using a Microsoft ADAM or ActiveDirectory ldap server, the samAccountName attribute is unmanaged.

Additional properties:

1. AccountContainer, the DN of the account container, for example ou=accounts,dc=corp,dc=example,dc=com
2. CertificateTrust = Yes, over-ride the certificate checking for SSL connections. Default is No.
3. DeletedContainer, the container that deleted accounts will be moved to for the Active Directory resource type. For example:

   OU=Deleted,OU=accounts,DC=corp,DC=example,DC=com
4. ObjectClasses, a comma separated list of Object Classes used for the create. Defaults to LDAP Type specific values, see below.
5. Pagesize. The LDAP Page Control is used to read the directory in multiple pages, by default each page is 100 entries. This can be varied to improve the performance of the LDAP extract. The upper limit depends on the LDAP implementation, typically it is 5,000 – the Administrative Limit, which can only be exceeded by an administrator account. Larger values increase the load on the LDAP directory, smaller values increase the load on the client. If Pagesize is 0, the LDAP Page Control is not used, the control is not always available in older LDAP implementations, Check with an LDAP Browser.
6. PasswordAlgorithm. SHA or SHA-512. SHA is the default.
7. SearchFilter. The LdapSearchFilter is the RFC implementation and is implemented by the ldap server, FilterAttribute and FilterValue are implemented by AMX . Default value is (objectClass=Person). Microsoft AD and ADAM are more efficient if SearchFilter is set to (objectCategory=Person) because it’s indexed.

   Note that unlike other RFC LDAP implementations, neither Microsoft AD nor ADAM can use the extension (&(objectCategory=Person)(|(ou:dn:=LON)(ou:dn:=EDI).

8. Type = ADAM or blank

LDAP Types

ADAM

- active, attribute userAccountControl
- objectClasses, default top,person,organizationalPerson,user
- password attribute unicodePwd value is a Unicode version of the password enclosed in “\”s

LDAP other implementations OpenLDAP, Oracle DSEE etc.

- active attribute is the standard RFC “Active” or “Inactive” values.
- objectClasses, default top,person,organizationalPerson,inetOrgPerson
- password attribute userPassword with salted SHA.

LDIF

An adapter operating on an LDIF formatted file as a managed resource, the adapter name is “LDIF”.

Remote

An adapter using an instance of identityServer which performs the same extracts, transforms and loads as identitySync including the Remote adapter and uses the same properties and schemas. The adapter names are:

- “RemoteIdentity” for identity sources
- “Remote” for managed resources

It is often easier to develop the transforms on a remote server using identityReport so that where identitySync is run there are no transforms. Additional properties for the Remote adapter are:

1. Network, the ip address of a secondary interface when present and when used to communicate with the system running identityServer. Default is the DNS entry of the hostname.

2. Remote, the adapter name and instance in the identityServer properties file on the remote system, for example RemoteRemote1 = LDAP1

3. Resource, the DNS name or ip address of the identityServer host and port, for example RemoteResource1 = AMX3.corp.example.com:1800

Salesforce

An adapter for Salesforce web services using wsdl as a managed resource. The adapter name is “SForce”. If a web proxy is used see the proxyUser and proxyPasswdfile properties to authenticate with the proxy server.

The account used to manage the Salesforce users must have a security token. Login to the Salesforce site Personsal Setup / My Personal Information / Reset My Security Token. An email will be sent with the token.

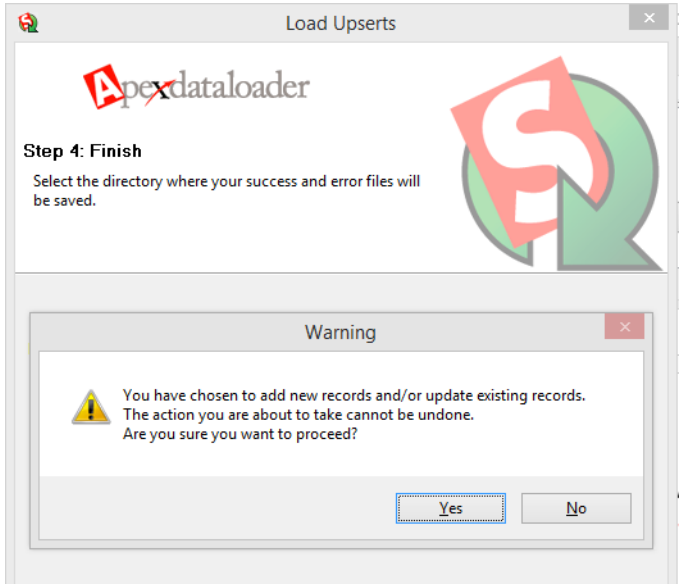
If your password = "mypassword"
And your security token = "XXXXXXXXXX"
You must enter "mypasswordXXXXXXXXXX" in place of your password

Note that you do not enter a security token in place of your password when logging into salesforce.com via a browser.

A new account details are sent directly to the user by Salesforce.

Salesforce does not delete users. When the active flag is not set the user is effectively deleted. Clearing the active flag makes other changes to the Salesforce system and identitySync never clears it. For this reason identitySync does not implement a delete for this adapter. The Freeze user functionality is used to disable users.

IdentitySync and the Salesforce ApexDataLoader https://na9.salesforce.com/ui/setup/sforce/DataLoaderSetupPage?setupid=DataLoader&retURL=%2Fui%2Fsetup%2FSetup%3Fsetupid%3DDataManagement have similar functionality. IdentitySync however avoids this problem:



identitySync is reversible and the ActionFile.txt shows the changes that will be made. Also ApexLoader does not disable user accounts. identitySync uses the "Freeze" function to disable user accounts, which is part of the UserLogin table rather than the User table. Disabling accounts when a person leaves is the most important function of an identity management system, especially when an organisation is using the cloud.

Additional Properties:

1. FilterAttribute. This needs to be carefully selected. For example email which is a mandatory attribute can be used with the mail domain of every email used in the organisation. FilterValue = @axnt.co.uk:@example.com. A blank FilterAttribute is less risky and recommended. This can be used if the builtin accounts are set as inactive or they are added to the salesforce identities by listing them in a separate CSV file with ExtractMode=Add.

2. ProfileID. Should be something like 00eE0000000dVSbIAM. It is a mandatory attribute for an account create in SalesForce and consequently identitySync. Use identityReport to find a suitable value for ProfileID.

3. Pagesize, an optional property. The Salesforce batch control is used to read the records in multiple queries, by default each page is 500 entries. This can be varied to improve the performance of the extract. The minimum is 200 and the maximum is 2000.

Salesforce Manager

The Salesforce manager attribute is ManagerID, in identitySync the Manager's name attribute is used. identitySync handles this in the same way as the ActiveDirectory.

The Identity attribute containing the identity of a person's manager is flagged with ManagerID.

The Identity attribute in the manager's record that contains ManagerID is flagged with ManagerJoin.

For example a person's identity record has the Manager attribute and it contains the employeeID of the person's manager. The Identity schema would contain:

```
Manager,managerEmployeeID;ManagerID
employeeID,employeeID;ManagerJoin
```

ManagerID and ManagerJoin allows the identity record of a person's manager to be found by identitySync. Given the person's manager's record, identity Sync updates the person's Identity attributes flagged with ManagerName. The Identity schema would then contain:

```
,managerName;ManagerName
```

The Salesforce schema:

```
ManagerId,ManagerName
```

identitySync will compare the ManagerName attributes in the Metaverse, and update the Salesforce ManagerID attribute where there is a difference.

Template

The adapter is a synthetic source of Identities and accounts useful for testing. The adapter names are TemplateIdentity and Template. It creates manual ldif-like update file.

Property PageSize is used to specify the number if records generated.

Textfile

An adapter that reads SAP-HR output files as a source of identities. The adapter name is “Textfile”, the file format is:

```
objecttype: person
company: 1000 / axnt Ltd
department: 101 / HR
division: /
employeeid: 01000001
end_date: 06.04.2014
firstname: Aileen
knownas: Aileen
lastname: Small
payrolllocation: UK / Edinburgh
start_date: 06.12.2007
status: Withdrawn
type: FTE / Established

objecttype: person
```

This format is used by SAP HR. Notice dates are in a German format and may have to be reformatted to the current locale in the Textfile schema. Status values are Active or Withdrawn. Additional properties:

1. Delim, the deliminator between the attribute and its value. Default “:”.
2. RecordSeparator, default null

Unix

A Unix adapter for Unix, it uses a remote ssh connection to manage local accounts on a system. HPUX, Linux / Centos and Solaris are supported. The adapter can be a source of identities or a managed resource. As a source of identities a master system can be used to synchronise the local accounts on one or more slaves. The adapter names are:

- “UnixIdentity” for an Identity source
- “Unix” for a managed resource

Unix accounts are disabled by setting the shell to /bin/false, and when re-enabled the shell is set to the system default or the value in the Shell property.

Accounts that have no matching identities are considered deleted, they are marked by making them a supplemental member of the “DeletedGroup” group as defined in the properties. See below. Accounts are never deleted unless a create is being undone. This preserves the reversibility of identitySync.

The manual files ManualDo and ManualUndo contain a copy of the commands that will used by the load phase which uses a ssh connection to the target server to execute them. The manual files do not include the user’s password it must be set manually and manually expired.

When the user account name is generated it is case sensitive. To create lowercase account names using an AccountNameTemplate such as:

```
UnixAccountNameTemplate = %lastName%%firstName%
```

Ensure that the Metaverse values are the correct case by adding Immediate Evaluation flags to the Identity Schema, for example:

```
firstName,firstName;toLower

lastName,lastName;toLower
```

This may cause an issue when assembling the full name in a post processing attribute transform, for example:

```
,name;concat: %firstname% %lastname%
```

In this situation extract the first and last names twice in the Identity Schema:

```
firstName,firstNameLower;toLower
```

```
lastName,lastNameLower;toLower

firstName,firstName

lastName,lastName
```

Then use the firstNameLower and lastNameLower in the account name template:

```
UnixAccountNameTemplate = %lastNameLower%%firstNameLower%
```

Properties:

1. DefaultGroup, optional property. The name or gid of the user's default group on create. If omitted the Unix O/S default group is used as shown by useradd –D.
2. DeletedGroup, optional property. The name of a group which is used to identify users that have left the organisation and no longer have an identity record. When using identityReport leave this blank or don't define it to see all the accounts.
3. FilterAttribute: Metaverse attribute used to filter accounts, for example uid. See FilterAttribute and FilterValue

   It is important that this is set so that builtin accounts are not managed. By default the adapter will not extract:

   - root
   - daemon
   - bin
   - sys
   - sync

   however other builtin accounts must be skipped by defining a suitable FilterAttribute and value. The Unix adapter will extract but will not add changes to the transaction file for the account being used to manage the instances as defined in the property User.

4. Resource: the DNS name or ip address of the Unix host.

5. RoleDeleteDelay, the number of days delay for an account to be removed from a group when RoleMode=strict. Default is 7. See hysteresis for details.

6. RoleGroups, a list of groups that identitySync will manage based on RoleMode. The list of groups must use the deliminator defined in ActionFileDelimList.

7. RoleMode = strict, loose or none, the default is loose. See Roles for details

8. Shell, optional property over-rides the system's default shell when creating or re-enabling accounts in /etc/passwd. The default shell is defined in the useradd command.

9. Sudo = true or false, an optional property, default is true. When Sudo is true the identitySync load process uses sudo to get privileges to manage accounts. The account used to manage the system must be given permissions to allow execution of useradd and usermod.

   When Sudo is false, the root account and password is used to manage the system in the identitySync properties file. Sudo true is the more secure method, permitting root login makes it available to all ssh clients. This will need a change to /etc/ssh/ssh_config to enable PermitRootLogin Yes. Use svcadm restart ssh to activate the configuration change.

10. Type: HPUX, Linux or SunOS

Unix Roles

Roles are activated when the Metaverse Attribute memberOf has the "delta" Attribute flag. When the delta Attribute flag is not used all the groups in a person's Identity record are used to replace group membership in the Unix Resource.

The Unix adapter implements "loose" and "strict" role management modes for Unix groups. The default is "strict".

```
UnixRoleMode1 = loose
```

Loose mode adds but never removes groups defined in the Identity source. In "strict" mode, DeleteDelay delays the removal of groups from Resource accounts. The delete delay is measured in days, the default is 0.

```
UnixRoleDeleteDelay1 = 7
```

Roles and the corresponding Groups can be defined in the identity source using a lookup table based on a person's role. For example a lookup file LookupRole.txt containing:

```
admin,mail:adm:root
dba,mail:adm:dbadmin
web,mail:adm:webadmin
sql,mail:adm:sqladmin
*,adm:mail
```

When used with an identity schema:

```
role,memberOf;delta;lookup=LookupRole.txt
```

Results in the groups that an Identity is a member of, based on their role, found from a lookup in LookupRole.txt.

Only groups that are defined as Managed Groups are added and removed by identitySync. See Managed Groups for how these are defined. For example if the group "Administrator" is not a Managed Group it will never be added or more importantly, never removed in the Resource.

Unix Types

Type as reported by uname:

- SunOS:
    - last used to get lastlogin attribute.
    - groups used to get an account's groups for group management and marking an account as deleted. Response is:
        - Account : group1 group2 …
    - useradd –D for default parameters. UID defaults to the next UID greater than the largest UID in /etc/passwd. For a new system create one user in the intended range
    - useradd is not able to set the password, the –p attribute sets the project rather than the password. The password is set using an ssh shell, the passwd command, and expect which is looking for a ":" such as:

        - New Password:
        - Re-enter new Password:
    - passwd –f to expire the password for a new account and force a password change at the first login

- Linux
    - lastlog used to get lastlogin attribute.
    - groups response
        - group1 group2
    - /etc/login.defs in Linux for the parameters for useradd such as UID_MIN and UID_MAX
    - The passwd is set using useradd –p with the password encrypted using crypt().
    - passwd –e to expire the password for a new account and force a password change at the first login

Windows Local SAM

A Windows Local managed resource adapter. The adapter name is “WindowsLocal”. It uses the .NET DirectoryEntry WinNT container. Manages local accounts and local groups.

identitySync synchronises each instance individually, the ExtractMode is always “single”.

Accounts that have no matching identities are considered deleted, they are marked by making them a supplemental member of the “DeletedGroup” group as defined in the properties. See below. Accounts are never deleted unless a create is being undone. This preserves the reversibility of identitySync.

The RemoteRegistry service is started if necessary, if it is started it is stopped on completion. Reported in the debug file as:

Winlocal RemoteService System <server> remoteRegistry Start Mode Manual
Winlocal RemoteService System <server> remoteRegistry Service State Stopped

Winlocal RemoteService Start Service return status 0

Properties:

1. DeletedGroup, optional property. The name of a group which is used to identify users that have left the organisation and no longer have an identity record. When using identityReport leave this blank or don't define it to see all the accounts.

2. FilterAttribute, the metaverse attribute that is used to filter valid accounts in a resource. Accounts are extracted when part the value of the FilterAttribute matches one of the FilterValues. The match is case insensitive. Blank includes all accounts.

3. FilterValue, a list of values for an attribute defined in FilterAttribute, separated by ":" that indicate records to be included. For example:

   FilterAttribute = description

   FilterValue = managed:Managed

3. Name, the NETBIOS name of the server. Used when adding local accounts to local groups. When the Name is not the NETBIOS name of the server, the following error will be reported:

   Error: WinLocal LoadAttribute MemberOf <group> message: Exception has been thrown by the target of an invocation.

4. Resource, the hostname, may be fully qualified. Fully qualified hostnames provide the best performance.

5. User, the administrative account. This may be a local user or a domain user. The format of User is Domain\User, where Domain may be the NETBIOS name for a local user. For example a local user WFS1\Administrator or a domain user corp\engela. A domain account must be used to enumerate domain groups. If an existing windows connection exists, for example a share is connected, the same account used to make the connection must be used for this property or errors (0x80005004) may occur.

Windows Local Roles

These work in the same way as Unix Roles. The properties are:

1. RoleDeleteDelay, in strict mode, the number of days before a group is removed, default is 7. See role hysterisis.

2. RoleGroups, a list of groups that identitySync will manage based on RoleMode. The list of groups must use the deliminator defined in ActionFileDelimList.

3. RoleMode, strict or loose, default is loose.

For example:

```
WindowsLocalRoleDeleteDelay1 = 7
WindowsLocalRoleMode1 = loose
```

Windows Local Firewall Settings

The Windows Local adapter uses the following services

1. File and Printer Sharing. This feature is used for sharing local files and printers with other users on the network. (Uses NetBIOS, LLMNR, SMB and RPC)

2. Remote Services. This feature allows remote management of local services. (Uses Named Pipes and RPC).

3. Windows Management Instrumentation (WMI). This feature allows remote management of Windows by exposing a set of manageable components in a set of classes defined by the Common Information Model (CIM) of the distributed management task force. (Uses DCOM).

Schema

The schema is defined in a file whose name is the adapter property “schema”. Its format is:

[StagingAttributeName],[MetaverseAttributeName[;AttributeTransformr[;AttributeTransform][..]]]

Staging Attribute Names may be blank in cases when the Metaverse Attribute Value is derived from a transform of other attributes. Derived MetaVerse attributes may be the result a transform, for example a concatenation of other Metaverse attributes to create a distinguishedName:

,distinguishedName;concat:CN=%lastName%\, %preferredName% OU=%ou%,OU=accounts,DC=corp,DC=example,DC=com

Staging Attributes can be defined more than once, for example in the identity schema, using fullName for both the CN and the displayName:

fullName,CN

fullName,displayName

Metaverse Attribute names may be blank, in which case the Staging Attribute name is used.

Metaverse Attribute names can only be defined only once. For example the schema below cannot be used because the Metaverse attribute CN is defined twice:

CN,CN

displayName,CN

Attributes

Staging Attributes names are those found in resources, for example the column names in an Excel file or the Active Directory attribute names. Depending on the resource type, these may be case sensitive.

Metaverse attributes are created in databases of identities and resources by identitySync. The Metaverse Attribute Names are the names that are given to the attributes in the Metaverse , these are case sensitive. Attribute names that match between the identity and resource Metaverses are synchronised in the resource. Synchronisation of an attribute can be disabled by using the noSync Attribute Flag. The debug file reports attributes that will be synchronised in lines starting AMXlib GetSynchronisedAttributes when the logging level is 2 or greater.

The Metaverse has the following reserved Attribute Names:

1. accountName, the user login name. In all Adapters the accountName value must be unique, and must be present in both the identity and resource schemas for an account create. During the create:
    a. If the accountName attribute has a value in the identity metaverse it is used.
    b. Otherwise a unique accountName is created using the adapter properties AccountNameTemplate and AccountNameFormat. See Adapter Properties.
2. distinguishedName, mandatory in the Active Directory and LDAP schemas. Like accountName, during create:

a. If the distinguishedName attribute has a value in the identity metaverse it is used.

b. Otherwise a unique distinguishedName is created using the adapter properties DNTemplate and DNNameFormat

3. memberOf, group membership used for Roles. When memberOf is defined, Roles are managed by the adapter. The two types of role management are ActiveDirectory Role Templates and Identity based Roles.

4. resource, a singleton Metaverse Attribute having no equivalent Staging Attribute. The attribute value is copied from the adapter Name property in the properties file. The attribute is not synchronised. For example:

```
,resource
```

Attribute Values

Some adapters can recognize and extract multivalue attributes, these are assembled into a string delimited by the property ActionFileDelimList. The transform phase uses ActionFileDelimList as an indicator that the attribute value is a list, decomposes it and applies the immediate evaluation transform to each component of the list. For example:

Schema with immediate transform

```
memberOf,;replace/$/More/
before   group1<delimlist>group2<delimlist>group3
after   group1More<delimlist>group2More<delimlist>group3More
```

Schema with post extract transform

```
memberOf,;concat:More
before   group1<delimlist>group2<delimlist>group3
after   group1<delimlist>group2<delimlist>group3More
```

Metaverse attribute values are compared using case sensitivity unless the ignoreCase attribute flag is used.

Attribute Flags

Flags mark Meta attributes with special meanings:

1. active, mandatory for resources only if the load mode is D (disable). Flags the attribute as the definition of the active attribute. It is used to extract and load the active attribute from specific adapters:

    - Active Directory
    - Exchange
    - HomeShare
    - LDAP, type = ADAM
    - Unix
    - WindowsLocal

    All the above transform to values Y and N when the active flag is present. Y indicates an active or enabled account, N inactive or disabled. Lookup or replace can be used to transform the values to load into resources, for example an Identity source such as an ActiveDirectoryIdentity with a schema schema:

    ```
    userAccountControl,status;active;replace/Y/True/;replace/N/False/
    ```

    Will load True or False into a resource with a schema:

    ```
    personStatus,status;active;
    ```

    In situations where an identity source is used to update multiple resources with differing formats for active, use additional metaverse attributes. For example a HR database:

    ```
    Employee Status,status1;replace/Active/Y/;replace/Inactive/N/
    Employee Status,status2;replace/Active/True/;replace/Inactive/False/
    ```

    Use status1 with Active Directory, Exchange etc and status2 for an adapter that uses a value True or False.

    See below, the attribute flag enddate which also transforms to Y or N.

2. displayName, this flag is mandatory it marks this attribute for use as a record identifier when logging information or errors concerning the record, and for the debugUser property.

3. ignoreCase for resources. The resource attribute check for attribute value updates is case insensitive. Default is case sensitive. For example with ignorecase a value LONDON -> London will not be updated.

4. join for resources and for identities used in ExtractMode=Merge. The attribute that will be used to join the resource records to the identity records, and to join the merged records with the original identity records. The join attribute must be present in both schemas.

5. nosync for resources. The resource attribute will not be synchronized by accountSync or identitySync. This is ignored on identity sources.

6. objectName, the name that will used in the Transaction file. Defaults to accountName or distinguishedName for the Active Directory and LDAP adapters.

7. roleGroup for resources, the attribute that contains group membership for roles.

8. unique, uniqueSilent checks that the attribute value is unique and builds a table of values and attributes before the extract filter is applied. Unique causes a message and termination of the process when a duplicate is found. UniqueSilent only builds the table and does not cause a termination. For managed resources these tables are used during the creation of unique accountNames, distinguishedNames and mail aliases. See the adapter properties AccountNameTemplate etc.

   This check for identity attributes should be done during the extract to prevent a failure during the load phase caused by a duplicate attribute value. Unique values are for example:

```
Identity attributes: employeeNumber
ActiveDirectory: distinguishedName, mail, UPN, accountName
LDAP: distinguishedName, UID
```

   The uniqueness check is across all the records for an extract mode add, if a Filter is used to build the metaverse from multiple instances of the same system, for example an Active Directory where organisational units have different formats of the description or display names needing different schemas. In this case it will be necessary to use the uniqueSilent attribute flag for the second and subsequent extract schemas. Because adding the values to the table will fail the uniqueness test for the second adapter instance since the value was added by the first instance.

```
ActiveDirectoryResource1 = dc.corp.example.com
ActiveDirectoryAccountContainer1 = ou=accounts,DC=corp,DC=example,DC=com
ActiveDirectoryFilterAttribute1 = distinguishedName
ActiveDirectoryFilterValue1 = ou=EDN
ActiveDirectorySchema1 = ActiveDirectorySchema1.txt

ActiveDirectoryResource2 = dc.corp.example.com
ActiveDirectoryFilterValue2 = ou=LON
ActiveDirectorySchema2 = ActiveDirectorySchema2.txt
```

It is recommended that UniqueSilent is always used for managed resources, because the resource itself will enforce uniqueness when it is required, and a failed uniqueness check will be caused by the situation described above or something similar.

Attribute Transforms

Attribute value transforms are done in the following order:

1. Immediate transforms from left to right as they are defined in the schema. These transforms operate on single Attribute Values, for example ToLower.

2. Record transforms are done after the complete record is extracted in the order they are defined in the schema, that is top to bottom. These transforms operate on multiple values in the same record, for example concatenate.

3. Database transforms are done after the all the records are loaded into the database. These transforms operate on the values of multiple records, for example creating a unique account name.

Immediate Transforms

The transforms have no effect on anything but themselves. Each attribute value is transformed as it is extracted. Transforms can be performed independently on each managed resource. For example employee identities extracted from a HR database and contractors from a Procurement database may have different name attributes:

1. Name in the HR Database source is 2 attributes Firstname and Lastname.
2. Name in the Procurement database is a single attribute Fullname. Create 2 new attributes Firstname and Lastname during the extract using the left and right transforms on Fullname.
3. The identity metaverse will contain attributes Firstname and Lastname for identities extracted from both the HR and the Procurement databases

When using identitySync transform the identity attributes rather than the managed resource attributes. For example:

1. Fullname in the identity source is Firstname Lastname.
2. Fullname in the resource is Lastname, Firstname.
3. Transform the Fullname in the **identity source** using the comma attribute transform, see below.
4. Compare the values of the Fullname attributes in the identity and resource metaverses.
5. Load updates into the resource.

Using attribute transforms on **resources** may have unintended consequence

1. Fullname in the identity source is Firstname Lastname.
2. Fullname in the resource is Lastname, Firstname.
3. Transform the Fullname in the **resource** using the nocomma attribute transform.
4. Compare the values of the Fullname attributes in the identity and resource metaverses.
5. Load updates into the resource. Any updates will use the value in the metaverse and the format will be FirstName LastName which is different to all the other records in the resource.

6. Subsequent identitySync runs at step 3 above will find no comma and therefore make no change, so the identity and resource attribute values will match and again there will be no update.

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

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

operating on a value “EMEA UK / Edinburgh”

right returns Edinburgh

lookup with a file city.csv containing Edinburgh,Scotland, returns Scotland

Attribute transforms that are evaluated immediately as each attribute is extracted 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

   Notice that this does not remove a comma, this can be done with replace

2. date will convert an attribute to a date string in the current locale format. Dates are adapter specific:

   a. Active Directory adapter automatically converts staging attributes from the internal representation of a date to a string representation. The flag is not required as all the date staging attributes are well known

   b. Excel converts an Office Automation date to a string representation

   c. Unix converts Epoch time to a date string in the current locale format

d. Winlocal truncates a date and time to a date

e. Other adapters use .Net DateTime.Parse which will convert strings to the locale format for dates. Such as:

f. 1/4/ 16

g. 02/04/16

h. 03/04/2016

i. 4-4-16

j. 5/Apr/16,

k. 06.04.2016

l. Apr 7, 2016

m. 2016-04-08

n. Sat, 09 Apr 2016

o. Apr/10/16,11:4:2016

If the time is included the ActionFileListDelim must be changed from ":" or a truncate transform used to remove the time from the attribute value. The value 11:04:2016 cannot be parsed, use replace to change ":" to something like "/". If US formatted dates are used in the form 04/30/2016 use replace to re-order month, day and year:

```
lastLogin,month;replace?(\d{1,2})/(\d{2})/(\d{2,4})?$1?
lastLogin,day;replace?(\d{1,2})/(\d{2})/(\d{2,4})?$2?
lastLogin,year;replace?(\d{1,2})/(\d{2})/(\d{2,4})?$3?
, lastLogin;concat:%day%/%month%/%year%
```

3. dateEpoch will convert Unix Epoch times, milliseconds since 1/1/1970 to the current locale date time format. The Unix adapter does this by default, this is intended for adapters such as databases which may use this format. For example:

   1463225309706 epoch time converted to 29/06/2016

9. endDate. The attribute value is the date and time of the leaving date and usually used to set the active metaverse attribute. If the date is in the future the value returned is "Y", otherwise "N". If the value is a only a date, the time is 00:00:00 so the person will be marked inactive at the beginning of the last day. This is probably not the behaviour that is required. To set the leaving date with a time matching the end of the working day use a transform such as:

   ```
   endDate,active;replace/(\S{10})/$1 17:30:00/;endDate;Lookup:Active.txt
   ```

   Regex appends 17:30:00 to a string of 10 characters, leaving blank values unchanged. Note that the concatenation attribute transform cannot be used because it will always append, including zero length values, and concat cannot be used because it is performed after end date is evaluated. See Transform Sequence. Lookup is used to set any null value to Y.

4. 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.

5. left<n><delimiter>, split the string using the specified delimiter returning substring number n starting 0 from the left. The Escape character "\" is respected. Also see also right.

   If the delimiter is not found the whole string is returned. If the substring number is greater than the number found in the string a null string is returned.

   For example:

   ```
   Name,firstName;left0,
   ```

   Evaluating Name containing a value "Small, Alan"

   Uses "," to split the value and returns the first part.

Sets firstName to “Alan”

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

Evaluating distinguishedName containing a value "cn=Brown\, Alistair,ou=lon,ou=accounts,dc=corp,dc=example,dc=com",

First uses “,” to split the value and returns the first substring cn= Brown\, Alistair

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

```
memberOf,groups;left0,;left1=
```

Returns Sales & Marketing:Sales Reporting when evaluating groups from a string containing "CN=Sales & Marketing,OU=Groups,DC=corp,DC=example,DC=com:CN=Sales Reporting,OU=Groups,DC=corp,DC=example,DC=com”

```
displayName,firstName;left0
displayName,lastName;right0
```

Returns firstName “Alan”, and lastName “Small” when evaluating a DisplayName from a string containing “Alan J Small”. This is useful when generating accountNames

```
Email,firstName;left0@;left0.
```

Returns firstName Philip when evaluating Email from a string containing Philip.nesfield@example.com

6. 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
```

A file containing:

```
// Comment
```

```
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 or the value contains a comma enclose it in “s, such as:

```
“London, England”, UK
```

Or

```
Sales,"CN=Sales & Marketing,CN=Users,DC=corp,DC=example,DC=com:CN=Sales Team Read
Only,CN=Users,DC=corp,DC=example,DC=com”
```

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

8. 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”

9. 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. For example:

```
Comment,comment;replace/,/:/
```

Will replace a ‘,’ with a ‘:’.

```
Comment,comment;replace|/| |
```

Will replace a “/” with a “ “. Note that this implementation of replace cannot replace “;”.

```
telephone,telephone ;replace/^225/0131 225/;replace/^212/01256 212/;replace/^01/+44(0)1/
```

Will normalise phone numbers adding an STD code to local numbers beginning 225 and 212 and then adding country code to any numbers without one. Evaluation is from left to right, more can be added.

```
telephone,telephone;replace/ //;replace/(\d{4})(\d{3})(\d{3,6})/$1 $2 $3/
```

Will rewrite phone numbers by removing all the spaces and then reformatting them 4 digits 3 digits 3 or more digits

```
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

```
active,;active;replace/^$/Y/
```

Will replace a blank value with Y

10. right<n><delimiter>, split the string using the specified returning substring number n starting 0 from the left. This is the mirror image of left. For example:

    ```
    CN,ou;right2,
    ```

    will return the value "OU=Edinburgh" from a string containing

    ```
    "CN=Philip Nesfield,OU=Users,OU=Edinburgh,DC=example,DC=com"
    ```

    ```
    distinguishedName,ou;right2,;left1=
    ```

    will return the value "Edinburgh" from a string containing

    ```
    "CN=Philip Nesfield,OU=Users,OU=Edinburgh,DC=example,DC=com"
    ```

11. sort, the attribute value is sorted when it is deliminated by the value property ActionFileDelimList. This is useful in identityReport, but not identitySync which can compare unsorted multivalued attributes. Adapters that recognize multivalued attributes such as the Active Directory sort them during extract, adapters such as Excel or Textfile have no information concerning multivalued attributes and consequently are not sorted unless the sort attribute transform is used.

12. title, capitalise the first character of each word

    ```
    description,description;title
    ```

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

13. toLower, convert the value to lower case.
14. toUpper, convert the value to upper case.
15. trim, remove leading and trailing spaces from a value
16. Truncate, truncate a string to the first n characters. For example:

    ```
    middleName,initial;truncate1
    ```
    operating on a value “Ian”, returns “I”

Record Transforms

These transforms use multiple attributes of the same record and are evaluated when the complete record is extracted.

For managed resources, these transforms are evaluated after all the attributes of a record have been extracted.

For identities, these transforms are evaluated after all the identity sources have been extracted, so they can use values added by an ExtractMode=Merge as well as values processed in the immediate category above.

1. concat, concatenate metaverse attributes and store them into a new one. This is the last transform to be evaluated,

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

   ```
   ,name;concat:%lastName%\, % firstName%
   ```

   ```
   ,comment;concat:ActiveDirectory schema version 3.1 - 10/12/2018
   ```

   Concats are processed in the order that they appear in the Schema. So, both of these transforms 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
```

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

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

Will store the preferred name in the firstName attribute, unless it is blank in which case the givenName will be used.

3. Copy, copy the value of the flagged attribute from the managed resource to the identity metaverse where a valid join is found.

4. CopyIfNull, copy, only if the value of the flagged attribute is blank or undefined in the Identity Metaverse.

It is possible to create account or distinguishedName metaverse attributes values directly from Identity Attributes but it is not guaranteed to be unique. Adapters requiring unique names will abort during the Database Transform phase if the Unique flag is set for the attribute. At this point the identity record can be updated manually so that the transform makes a unique attribute value. Use the Database Transform “CreateTemplate” to automatically create a unique value. The advantage of doing this manually is that a more sensible name can be used.

For example when creating email addresses, the initials are usually blank, in the case of a name collision one or more initials are added to the Identity record:

```
,eMail;unique;concat:%firstName%%initial%.%lastName%@example.com
```

To create a distinguishedName in the Identity schema:

```
,distinguishedName;unique;concat:CN=%lastName%\, %firstName%,ou=accounts,dc=corp,dc=example,dc=com
```

Or

City,OU;lookup=City.csv

,distinguishedName;unique;concat:CN=%CN%,OU=%OU%,ou=accounts,dc=corp,dc=example,dc=com

In situations where the OU cannot be reliably derived in the Identity Schema this could cause existing accounts to be moved between OUs. If this is not desirable add “nosync” to the Schema, this will stop the distinguishedName being updated and it will only be used for an Account Create.

Database Transforms

Database transforms are done after the all the records are loaded into the database.

Unique Name Create Flags

Unique names are created when the resource attribute has the CreateTemplate defined in the schema **and** the identity attribute value is blank. It is used in the account creation process, for example when creating an account the identity record will have no matching resource record, however the

1. CreateTemplate used to create a unique name for the attribute. The Template is of the form CreateTemplate:(format1,format2…)%attributeName%%attributeName2%....

   The format is a comma separated numeric in the form X.Y defining the default length (X) and the maximum length (Y) for each attribute defined in the template. The default length (X) is used when the resultant name is unique otherwise additional characters are added.

   - If Y is not defined the maximum is the default length X and the value may be truncated.
   - If X is not defined the attribute value is only used to create a unique name.
   - If neither X nor Y is defined the default length is the length of the attribute value.
   - If X or Y is longer than the attribute value, the attribute length is used.

   AttributeName contains literals or metaverse attributes enclosed in %s. Special attributes are:

   - %*% is a numeric descriminator

- %random% is a random string made up from characters defined in the property RandomString.

For example:

CreateTemplate:(7,1.2)%lastName%%firstName%

Creates accounts such as smithj, smithjo

CreateTemplate:(7,1,0.1)%lastName%%firstName%%*%

ActiveDirectoryAccountNameFormat1 =

The discriminator %*% has a preferred length of 0, so the discriminator will not be used unless it is required to make the account name unique. Creates accounts such as smithj, smithj1

CreateTemplate:(6.6)E %employeeID%

Creates accounts E075151, E075152 etc and if the accounts are not unique the record is nor added to the transaction file.

CreateTemplate:(,6.10,,6.10)x%random%-%random%

With the property ActiveDirectoryRandomString1 = 0123456789ABCDEF

The maximum value in this case defines the number of retries to get a unique account name. Retries = max value – preferred length. Creates accounts xC5FAE6-8F4790, xF74A83-D93C68 etc

CreateTemplates can be used to create unique distinguished names for the reserved attribute distinguishedName of new LDAP or ActiveDirectory accounts when they are not present in the identity source. For example:

CreateTemplate:(,4.6,,4.6,)cn=C527.98AC.187E.0000.%random%.%random%,ou=people,o=C527.98AC.187E.020B.7099.D2F6,o=example

Manager Flags

Manager attribute transforms are done on Identities only and the results are used to update an account's manager attributes flagged with IsaManager, ManagerMail and ManagerName. Manager mail is used to send emails to managers for new starts. The key flags are the managerID and the managerJoin flags which are used to search for an identity's manager in the metaverse.

1. IsaManager, the metaverse attribute in the identity source is set to "IsaManager" if it is referenced by other identity records as a person's manager. Otherwise it is set to "NotaManager". This attribute must be a derived attribute with a blank StagingAttributeName. Attribute flags managerID and managerJoin must also be defined. The attribute is evaluated after all the extracts are finished so that all identity records are evaluated.

   Flags managerID and managerJoin must also be defined so that the metaverse can be searched for the empnum of an identity's manager in the attribute flagged by managerJoin. For example:

   ```
   manager,managerEmpnum;managerID
   empNum,employeeID;managerJoin;unique
   ,ManagerFlag;isaManager;lookup=ManagerStatus.csv
   ```

   To change the attribute value of the isaManager attribute, lookup the value in a file containing:

   ```
   IsaManager,TRUE
   NotaManager,FALSE
   *,
   ```

   This changes the attribute value to TRUE or FALSE

   An alternative to lookup is to use replace:

   ```
   ,ManagerFlag;IsaManager;Replace/NotaManager//;Replace/IsaManager/Manager/
   ```

   This changes the attribute value to "Manager" or blank.

2. managerID, the attribute that contains the identifier of the person's manager. For example the Identity Source may define a person's manager by the manager's employeeID or Full Name. When ManagerID is not defined manager updates are disabled, if ManagerID is defined an error will occur if ManagerJoin is not also defined.

3. managerJoin, the attribute that ManagerID matches. For example if the managerID attribute values are employee numbers, managerJoin is the employee number attribute.

4. mail, the attribute containing the email address of an identity

5. managerMail, the attribute of an identity which is set to the email address of the identity's manager. This attribute must be a derived attribute with a blank StagingAttributeName:

   ```
   ,managerEmail;managerMail
   Mail,mail;mail
   ```

   If this flag is defined Mail must also be defined.

6. name, the attribute containing the name of an identity. The attribute will be used to process the name of an identity's manager, and it will be used to synchronize a managed resource account's manager. For example in the Active Directory this name would be the distinguishedName. In most cases the source of identities does not contain the distinguidehedName, use the Copy Post Extract Transform on the distinguishedName in the managed resource schema.

7. managerName, , the attribute of an identity which is set to the name of the identity's manager. This attribute must be a derived attribute with a blank StagingAttributeName. In the example above it will contain the distinguishedName of the identity's manager and would be synchronized with the Active Directory account manager attribute.

ActiveDirectory

For an account create by identitySync either the CN or the distinguishedName must be defined in the identity source.

Staging Attributes:

1. accountControlTemplate, accountControl for create account. Default 512.

```
ADS_UF_SCRIPT                           = 1,        // 0x1
ADS_UF_ACCOUNTDISABLE                   = 2,        // 0x2
ADS_UF_HOMEDIR_REQUIRED                 = 8,        // 0x8
```

```
ADS_UF_LOCKOUT                                = 16,       // 0x10
ADS_UF_PASSWD_NOTREQD                         = 32,       // 0x20
ADS_UF_PASSWD_CANT_CHANGE                     = 64,       // 0x40
ADS_UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED        = 128,      // 0x80 Y
ADS_UF_TEMP_DUPLICATE_ACCOUNT                 = 256,      // 0x100
ADS_UF_NORMAL_ACCOUNT                         = 512,      // 0x200
ADS_UF_INTERDOMAIN_TRUST_ACCOUNT              = 2048,     // 0x800
ADS_UF_WORKSTATION_TRUST_ACCOUNT              = 4096,     // 0x1000
ADS_UF_SERVER_TRUST_ACCOUNT                   = 8192,     // 0x2000
ADS_UF_DONT_EXPIRE_PASSWD                     = 65536,    // 0x10000
ADS_UF_MNS_LOGON_ACCOUNT                      = 131072,   // 0x20000
ADS_UF_SMARTCARD_REQUIRED                     = 262144,   // 0x40000
ADS_UF_TRUSTED_FOR_DELEGATION                 = 524288,   // 0x80000
ADS_UF_NOT_DELEGATED                          = 1048576,  // 0x100000
ADS_UF_USE_DES_KEY_ONLY                       = 2097152,  // 0x200000
ADS_UF_DONT_REQUIRE_PREAUTH                   = 4194304,  // 0x400000
ADS_UF_PASSWORD_EXPIRED                       = 8388608,  // 0x800000
ADS_UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION = 16777216 // 0x1000000
```

2. accountName. Updates blank matching identity account names after the extract from a managed resource.

3. CN, part of the canonical name of the distinguished name. If both CN and distinguishedName are given values in the schema, the distinguishedName has precedence and a warning will be recorded in the debug file during the identitySync load phase.

   The CN must be escaped in the Identity schema if the value contains a “,”. For example if the format in the identity source is “Lastname, firstName”:

   ```
   CN,CN;escape
   ```

   The escape attribute transform will change the value to “lastName\, firstName” which is consistent with the ActiveDirectory.

4. distinguishedName, is mandatory in the ActiveDirectory schema, it is used to identify the ActiveDirectory Account in the Transaction File. It can be defined in the identity source. Updates blank matching identity distinguishedNames after the extract from an managed resource..

5. jpegPhoto images loaded from files.

6. mail, is a mandatory Identity attribute when the managerID attribute flag is defined. When an account is created it is used to email a person's manager with the account details Updates blank matching identity mail after the extract from an managed resource..

7. ManagerMail, is not mandatory. identitySync will add it to the Identity schema if it is not defined.

8. memberNested, an extension of memberOf, the distinguishedNames of the groups that an account name is a member of and the groups that that group is a member of. See Nested Groups for further details. This attribute uses a recursive extract of all group membership and is resource intensive, it is recommended to be used with identityReport for role analysis and tidying up the Active Directory. It doubles the run time of identityReport. This attribute is not intended to be used with identitySync.

9. memberOf, the distinguishedNames of the groups that an account name is a member of. Use the "left" Attribute transform to convert the distinguishedName to a name, for example:

```
memberOf,;left0,;left1=
```

10. name, treated like CN, and like the CN should not be defined in the Identity Schema with the distinguishedName.

11. sAMAccountName maps to accountName in the Metaverse and is mandatory in the ActiveDirectory schema for identitySync. The entry in the ActiveDirectory schema is:

```
sAMAccountName,accountName;unique
```

12. thumbnailPhoto images loaded from files.

13. userAccountControl transformed to Active Y or N, for example:

```
userAccountControl,active
```

Images

IdentitySync loads images into the Active Directory attributes jpegPhoto and thumbnailPhoto from images files extracted by the Image File adapter. The schema entries for these attributes must have the attribute flag syncFile. For example:

thumbnailPhoto,ImageFile;syncFile

The Active Directory adapter extracts images from the Active Directory into jpeg files and stores them in a subdirectory jpegPhoto or thumbnailPhoto with a filename constructed from the attribute with the DisplayName flag.

Images are compared as files and loaded only when there is a difference between the source image and the one extracted from the Active Directory. This prevents unnecessary reloading and replication in the Active Directory. Source images are resized according to the Active Directory adapter property ImageSize. The source image files are read using .Net Image Class which is able to read image formats such as jpeg, gif, png etc and resize them to the ImageSize Active Directory property. The default image size is 104 pixels. Once the images are loaded into the Active Directory, do not change ImageSize without consideration of the impact it will have on replication caused by doing a re-load of every resized image in the Active Directory.

Source of images is the Image File and Database adapters.

When the source image is blank, the Active Directory is not blanked during update. If the source image is present and the Active Directory is blank it is updated, and an image is only removed if identitySync is run in the undo mode having loaded it in the do mode.

Nested Groups

Organisations that use nested groups can use identityReport to report a user's complete group membership including group memberships that are implied through nesting. For example the UK Sales group has members Glasgow, Leeds and London. London has a member AUser:

- UK Sales Group
    - Glasgow Sales Group
    - Leeds Sales Group
    - London Sales Group
        - AUser User Account

When the groups that a user account is a "memberOf" are conventionally reported, they will show membership of the regional sales groups, but not the "UK sales" group. For example AUser will be reported as a member of the London Sales Group.

The activeDirectory adapter is able to report all the groups, explicit and implied that a user account is a member of. This is done using the “memberNested” Staging Attribute. It is intended for use with roleAnalyzer rather than identitySync where it can discover user accounts added to both regional and “UK Sales” groups making removal of access haphazard. In the example above:

AUser is reported as a member of London Sales and UK Sales.

memberNested is not intended to be used with identitySync because it will report groups that a user is an implied member of. In the example above A User is reported as an implied member of the group “UK Sales”. If identitySync found that this group needed to be removed, it would be unable to remove it.

The adapter is also able to identify circular nesting. In the above example:

- UK Sales Group
    - Glasgow Sales Group
    - Leeds Sales Group
    - London Sales Group
        - AUser User Account
        - UK Sales Group

That is where an administrator has added the group “UK Sales” to “London Sales” (which in this example is already a member of “UK Sales”) probably when trying to repair an access issue. The ability to detect circular nesting was added to the adapter after extracting accounts from a mid-size organisation where this had occurred several times, only to find that it is quite common in other organisations. It is impressive that the Active Directory firstly lets this situation occur and secondly doesn’t throw an exception when it does occur. In this case the following error is reported during Extract:

```
Error: ActiveDirectory ExtractMembers circular recursion of CN=UK
Sales,CN=Users,DC=corp,DC=example,DC=com in CN=London Sales,CN=Users,DC=corp,DC=example,DC=com
```

The identityReport report will correctly show AUser as a member of London Sales and UK Sales because the circular nesting was identified when the membership of London Sales was enumerated.

memberNested uses a recursive extract of all group membership and is resource intensive, making the run time of identityReport twice as long as usual.

This is not a Best Practices suggestion, but nesting groups does need to have an agreed organisational standard, and identityReport can check that it is being adhered to.

Database

The staging names are used to create the select statement. When quotes are required these can be added to the staging name. It is never necessary to use "AS". For example:

"Last Name",LastName

MSSQL

The adapter creates and extracts logins and not users.

Staging attributes in the sys.server_principals table:

1. isDisabled, True or False is mapped by the adapter to active N or Y in the metaverse.
2. principal_id, useful for the FilterAttribute to ignore system logins. During create the SQL server assigns a principle_id automatically.
3. Name, when using the recommended select statement as defined in the accountContainer property this is the role name which should be used with the memberOf metaverse attribute:

   ```
   spr2.name,memberOf;delta
   ```

   The roles are the server roles. Public, sysadmin, securityadmin, serveradmin, setupadmin, processadmin, diskadmin, dbcreator, bulkadmin

HomeShare

HomeShare uses the Win_Share properties, they are:

1. Name, usually the accountName.

2. AllowMaximum, True or False. When false limits connections to the value in MaximumAllowed
3. Caption, shortened version of Description. About 233 characters.
4. Description.
5. InstallDate always returns null / blank.
6. MaximumAllowed, the maximumum number of simultaneous connections when AllowMaximum is false
7. Type, always 0 = Disk, as opposed to 1 = printer queue. Used at an automatic filter.
8. Path, the directory path on the remote server. Not synchronised by identitySync.
9. Active, Y or N based on parent directory, see Adapter Properties InactiveContainer.

Also the Trustees, those accounts with access to the directory path. These are obtained by enumerating the Win32_SecurityDescriptor DACL property.

10. Trustee, names of accounts that have access rights to the directory. The type of access, read, write, read_attributes, update_owner etc are not reported. Trustees are reported in a sorted list separated by the ":" unless ActionFileDelimList is defined in the properties file. Not synchronised by identitySync.

Image File

Staging attributes:

1. EXIF properties. Most photographs have Exchangeable Image File Format (EXIF) properties. See the example schema file for the standard EXIF properties. Others can be added as necessary. The properties are extracted using .Net PropertyItems. Note that this method occasionally fails to read some malformed values. See StackOverflow article. The staging attribute names of the EXIF properties are used in the schema. For example:

```
270,Title

271,Camera maker

272,Camera model
```

```
305,Program name

306,Date created
```

2. FileCreateDate, the file create date which may be different to the EXIF property Date Created.

3. FileName. This has to be also used to create an identity name using attribute transforms to join to the Resource. For example:

   If the image files are called “UserImages\firstName LastName.jpg” and the identity name “firstName lastName” use:

   ```
   FileName,fullName;replace/^.*\\//;replace/.jpg//
   ```

   The first replace will remove the directory and the second will remove “.jpg”.

4. Height, number of vertical pixels in the image in the file.

5. Size, Width and height of the image in the file. Format is {Width=<width> , Height=<height>}

6. Width, number of horizontal pixels.

LDAP

Dependant on LDAP type. Use Apache Directory Studio or similar to enumerate the attribute names.

Remote

The Remote adapter has two schemas, the identitySync schema and the identityServer schema. The remote schema used by identityServer should do all the attribute renaming rather than the local schema used by identitySync. This is because the transaction file is created by identitySync locally and contains Metaverse attribute names. These are transferred to the remote system and looked-up by identityServer in its Metaverse to associate the corresponding Staging name. When the attribute name is changed locally by identitySync, this lookup will fail with the error message:

```
Error: Homeshare Load <Staging Attribute> Not updatable
```

Salesforce

Staging attribute names are case sensitive. Additional Attributes:

1. Alias. Suggest that this name matches the ActiveDirectory sAMAccountName

2. Username is the accountName

3. IsActive. Effectively a deleted user when IsActive is false. When IsActive is defined in the schema deleted users are included in the extract, when IsActive is not present in the schema the user record is not included. A schema for identityReport would normally define IsActive in the schema, identitySync would not. For usage see below.

4. IsFrozen. An attribute of the userLogin object rather than the user object. See Salesforce API documentation. A frozen user cannot login and is effectively disabled. IsFrozen = false maps to active = Y and vice-versa. The schema should contain:
   IsFrozen,active
   IsActive,IsActive

Salesforce Flags

These flags are for Salesforce but are applied to the Identity Schema, since it is the Identity Metaverse which needs to be updated with ManagerAccount.

1. ManagerAccount Salesforce manager attribute uses the AccountName of the manager. The Identity schema must contain an attribute with this flag when managerID is defined and the manager attribute is intended to be synchronised by identitySync.

Unix

Unix Schema for /etc/passwd
Unix has fields and a GECOS or Comment string in the 4th field. Fields are referenced in the schema as fn when separated by a ":" and gn as subfields of f4 when separated by ",". For example:

```
   albonw:x:1000:1000:Alban Wilson,Edinburgh,0845 085 5555,0781 409 5555,00075151:/home/albonw:/bin/bash
f0 --^
f2 -----------^
f3 ----------------^
```

```
f4 ----------------------------------------------^
g0 ------------------------^
g1 ----------------------------------^
g2 ------------------------------------------------^
g3 ----------------------------------------------------------------^
g4 ------------------------------------------------------------------------^
f5 ----------------------------------------------------------------------------^
f6 ------------------------------------------------------------------------------------^
```

The Unix adapter assembles the GECOS fields into a metaverse attribute gecos. This must be added to the schema as a singleton attribute in both the identity and the Unix schemas. The Unix schema for using gecos fields is for example:

```
f0,accountName;displayName
f2,uid;nosync
f3,gid;nosync
f5,homedir
f6,shell
g0,fullName
g1,location
g2,telephone
g3,mobile
g4,employeeID;unique;join
,gecos
```

Alternatively the full comment string can be synchronized, when a subfield of the comment is required for example as a join, the Left or Right substring functions can be used.

```
f4,employeeID;left4,;unique;join;nosync;displayName
f4,comment
```

The nosync attribute flag prevents the f4 field being updated with the employeeID.

The Unix comment is then synchronized with the comment from the identity source, which can be assembled from its components using concat, for example an identity schema:

```
fullName,fullname
firstName,firstName
lastName,lastName
```

```
location,location
telephone,telephone
mobile,mobile
memberOf,memberOf;delta
employeeID,employeeID;unique
,comment;concat:%fullName%,%location%,%telephone%,%mobile%,%employeeID%
```

Attributes:

1. Gecos, a singleton attribute that concatenates the g0-gn fields using a "," as a separator. It is used to update the gecos or comment field in the Unix passwd file.

2. Lastlogon, a singleton attribute that is the date of the last logon for identityReport. See the Unix adapter properties for details of how this is obtained.

3. memberOf is a singleton attribute load with the supplemental groups obtained from the /etc/group file during the extract process. The update process uses usermod –G which replaces **all** the supplemental groups.
   Alternatively use the delta flag to manage groups as roles. See Unix Roles

4. uid, usually managed by the O/S. identitySync can use the attribute in create and update modes, this is useful for duplicating a set of users on a new system.

Unix Flags

1. delta, identitySync manages the groups in memberOf as Roles. During the synchronize phase, differences between the attribute values that are lists will be recorded as deltas in the Action File created by identitySync. For example:
   An Identity with memberOf "Sales" synchronized with an Account having memberOf "Marketing" and "Sales".
   Will create a delta Action memberDel:Marketing

Windows Local SAM

Date attributes such as LastLogin do not require the "date" attribute flag.

Staging Attributes:

1. Name is the accountName

2. PasswordExpired 0 = No, 1 = Y. A lookup can be used to normalise these, for example

```
PasswordExpired,PasswordExpired;lookup=boolean.csv
```

The file Boolean.csv containing:

```
0,N

1,Y

*,Error
```

Will change the values to Y or N

3. UserFlags transformed to active Y or N, for example

```
UserFlags,active
```

Metaverse Attributes:

MemberOf, list of Groups that the user is a member of.

Roles

Roles are implemented by identitySync by adding and removing responsibilities (group membership) based on the value of an attribute in the Identity Resource.

AMX implements two mechanisms for managing roles:

- Role Templates which define the groups associated with the Roles defined in the managed resource as a template or model user for the role. Implemented for the ActiveDirectory and CSV, see ActiveDirectory Role Templates.
- Identity based Roles. The Roles and consequent group membership is defined in lookup tables used when extracting the Identities. Implemented for the ActiveDirectory, Database, Unix and Winlocal.

Identity based Roles

Roles are managed when the Schema Attribute memberOf is defined.

Identity based Roles are defined in the Identity source by using a lookup table to find groups associated with a Role. For example a lookup table based on location:

```
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 will 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 users.

It is implemented by using the "memberOf" Metaverse Attribute, fetching its values based on the value of a Role Attribute such as location in the Identity Source schema:

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

Multiple Roles can be defined by concatenating them. If there is no duplication the Roles.txt file can contain both lookups:

```
title,memberOf1;lookup=Roles.txt
location,memberOf2;lookup=Roles.txt
,memberOf;concat:%memberOf1%:%memberOf2%
```

Identity based Roles can use Role Hysteresis, however all groups have the same mode and delete delay. For example:

```
UnixRoleMode1 = strict
UnixRoleDeleteDelay1 = 7
```

Managed Groups

identitySync limits adding and removing group membership to groups defined in the Managed Groups table. Managed Groups are formed from:

- Groups defined in the property RoleGroups.
- Groups found in the Source of Identities for Identity based Roles.
- Groups found in Templates for Role Templates.

The table of Managed Groups can be found in the debug file in lines starting “AMXlib GetManagedGroups” when LoggingLevel is 2 or greater.

It is only necessary to add Groups to the RoleGroups property when a group is intended to be removed from every Identity. When RoleGroups is not used, only groups that are present in the Source of Identities will be Managed Groups and only those will be removed from Resource Accounts. Adding a group to the RoleGroups Property allows identitySync to remove it when it is not present in the Source of Identities, and it will only remove it if the Group Management mode is “strict”.

Leaving RoleGroups blank is the safe operational mode.

Strict, Loose and No Group Management

Management of roles can be done in 3 ways. This is intended to assist with the introduction of AMX into a production environment.

1. None.
2. Loose, Users are immediately added to groups but never removed.
3. Strict, Users are immediately added to groups based on their roles, and they will be removed at some time in the future based on the value in days of RoleDeleteDelay property in identitySync.properties. This is the normal production mode.

The usual release to production process is to run identitySync in Loose mode and introduce Strict with a long RoleDeleteDelay as data quality improves.

Hysteresis in Strict Roles

In production situations the timing for changing a person's role and consequently access rights can cause disruption. When this is done too early the person may be unable to complete the tasks for the old role, and a role change is sometimes not a sudden break from the old responsibilities. To avoid this sort of disruption AMX has role hysteresis which changes role in the following way:

1. The new groups are added immediately that the role changes. This allows the person to access documents and do some background reading concerning the new role.
2. For a defined period of time the person is a member of their existing groups and the new groups associated with the new role.
3. After the person starts the new role they are still able to access documents from their old role to answer occasional questions concerning it.
4. After the period of time has expired, the groups associated with the old role are removed. The length of time that the person has both roles is defined in identitySync.properties as RoleDeleteDelay and is measured in days.

Hysteresis is implemented by adding a future action to the Transaction File.

Transform Sequence

The resources are processed in the order that they appear in the properties file. When using the copy transform it is necessary to extract identities before managed resources. After each Identity record extract:

1. Transform using attribute value transforms
2. Concatenation
3. Filter

After all identity extracts

4. Merge identity extracts. This needs the complete set of identity records so that the merge can update all the records of all the extracts
5. Check Unique attributes and values

6. Update IsaManager. This is done last because it needs the complete set of all the identity records to find every manager.

After each managed resource record extract:

1. Transform using immediate attribute transform
2. Concatenation
3. Check Unique attributes and values
4. Filter

After each managed resource extract in ExtractMode = Single or the last extract in ExtractMode = Add

5. Merge account attributes into identity metaverse. Adapter specific
6. Copy flagged attributes from the managed resource Metaverse to the Identity Metaverse where there is a valid join between them.
7. Update Manager name and mail
8. Create transaction file

Transaction File

identitySync creates a Transaction File which is used to do, undo and redo changes to the resources during the Load phase. The file name is ActionFile.txt.

Format

The record consists of 3 fields for create and delete, and 4 for update. The field separator is defined in the properties file as ActionFileDelim1. Fields may contain multiple values, the separator is defined in the properties file as ActionFileDelim2.

Field1: date, the action will be processed on or after this date. Future actions are deferred and are used in role management.

Field2: Resource name and object or account name. The resource name is defined in the schema as the Metaverse Attribute “resource”, its value is copied from the ResourceName in the properties file.

Field3: Load mode, Create, Update, or Delete. A disable is an update.

Field4: Metaverse Attributes and their values separated by the “=” character. Attribute values may be multivalued, and if so the values are separated by ActionFileDelimList as defined in the properties file.

Field5: present only in an Update, attribute values in Field4 are the current values and Field5 are the new values. Field4 is used in the undo action.

```
12/07/2013<field>corp<attr>cn=brian jones,ou=lon,ou=accounts,dc=corp,dc=example,dc=com
<field>Create<field>employeeID=2142<attr>comment=<attr>mobile=<attr>telephoneNumber=<attr>memberOf=<attr>
accountName=<attr>lastName=Jones<attr>initials=<attr>active=Y<attr>mail=brian.Jones@example.com<attr>dist
inguishedName=cn=brian jones,ou=lon,ou=accounts,dc=corp,dc=example,dc=com<attr>description=<attr>CN=brian
jones<attr>firstName=brian<attr>postCode=<attr>location=London<attr>displayName=Jones,
Brian<attr>street=<attr>title=<attr>department=<attr>manager=<attr>ManagerMail=<attr>passwdTemplate=CvcnC
vcn

12/07/2013<field>corp<attr>cn=Andrew Wilson,ou=lon,ou=accounts,dc=corp,dc=example,dc=com
<field>Update<field>location=London<attr>title=Sales<field>location=Glasgow<attr>title=Marketing

12/07/2013<field>corp<attr>cn=Alison Small,ou=lon,ou=accounts,dc=corp,dc=example,dc=com
<field>Update<field>active=Y<field>active=N

12/07/2013<field>corp<attr>cn=Annie Brown,ou=lon,ou=accounts,dc=corp,dc=example,dc=com
<field>Delete<field>
```

Security

The transaction File is inspected prior to being processed, and will be processed only if it passes the following tests:

1. Tripwire, the number of updates, disables and deletes are less than the maximums defined in the identitySync properties file.

2. Tamper check, the Action File has not been manually changed or edited.

3. Current version, the action date is today or prior to today.

When creating the transaction file each change is reviewed and only added if:

1. The account being changed is not the account being used to manage the resource as defined in the adapter properties.
2. An account being deleted is currently disabled. AMX never deletes an object, When an identity is removed from the authoritative source, AMX checks that the account is disabled and moves it to a recycle bin, or marks it as un-managed.

Archive

Transaction files are archived in a subdirectory Archive with names based on the date and time that they were written so that the newest transaction files are at the end of the directory, The format is ArchiveyyMMddHHmmss, for example a transaction file archived on 03 August 2016 at 9:17:37 am:

```
Archive160803091737.txt
```

Audit File

identitySync writes an audit file in its working directory called ActionAudit.txt. The records contain the date, the resource name, the Load mode (CRUD), the run mode (do, redo, undo), the record name and the attributes' new values. The file is not tamperproof, the emails containing the same information are a more secure audit trail.

The audit file is used by AMXUser.

Errors

identitySync detects many configuration and data issues, and to prevent a misconfiguration making inappropriate changes to a resource, an error message is reported and identitySync will usually abort. See Error Message document for details.

In situations where identitySync does not produce the expected results and there is no error message, the debug file may contain the answer. The debug detail is controlled by the "LoggingLevel" property. The highest level is 5 and this will produce large amounts of detail. To reduce the size of the debug file reduce the following techniques can be used:

1. If the issue is with the create, update, disable or delete process run identitySync in redo mode. This uses the existing transaction file and avoids the extract and transform steps.
2. If the issue is in the extract and transform process, run identitySync in the info mode which is the default. Disable the creation of the manual do and undo files by removing the properties ManualDo and ManualUndo for the relevant adapter.
3. Reduce the number of identities being processed by using the adapter's filter properties.
4. Use identityReport with the same adapter properties and schemas and inspect the output files rather than the debug file.

If the issue can be characterised by a specific user or account use the DebugUser property to see the attribute values in the debug file. Search for lines starting "AMXlib DebugUser"

Number of Accounts extracted is less than expected

If the number of accounts extracted is less than expected, check the debug file with the logging property >= 4 which will contain

```
<adapter> ExtractAttribute Skipped ..... philn
```

This is a result of the filter attribute and value in the properties file. See the adapter property FilterAttribute and FilterValue.

Number of joins is less than expected

The effect of this is 1 account delete and 1 account create for each failed join. The number of joins is reported in the debug file when the logging level is 2 or greater in the properties file with a message:

```
ActionFile CheckAccounts. Number of identities M number of joins N
```

Where N is greater than M. This can be caused by:

1. A missing or incorrect value in the join attribute in either the identities or the resource. Change the LoadMode to CRUDD and run identitySync in info mode. Each identity that failed to join to its account will be reported as a new account and each account that failed to join to its identity as a ghost. Compare the join attribute value in both identities and accounts and transform one of them, usually the resource. This can be done for one identity at a time by using the debug user property in the identitySync properties file, or using identityReport to check all the accounts.

Not updating an Account

Check that the load mode for the adapter in the properties file includes u (update).

Or

Check accounts is finding more than one account matching the identity. identitySync will only update when there is a one to one match. This could be caused by the attribute value of the join being blank in multiple records in both the source of identities and the managed resource. The number of matches is reported in the debug file when the logging level is 2 or greater in the properties file with a message:

```
ActionFile CheckAccounts Update suppressed for <Account> on <Resource> found N matches"
```

Not updating an Attribute

Check that the Metaverse attribute is in the ActionFile.txt as expected. If not:

1. Check the debug.txt file. Search for AMXlib GetSynchronisedAttributes Common Updatable Attribute. If the attribute is not present in this list it won't be synchronised. If it is not present make sure the attribute Metaname is in both the identity and the resource schemas.

2. Check that the nosync flag has not been added as an attribute flag in the resource schema

Not updating group membership

identitySync only updates managed groups, those groups discovered in the identity source and / or for specific adapters, those in the property roleGroups. If a group is not managed it will not be updated. This is most obvious when removing group membership.

Also roleMode=loose will add group membership, but never remove it. For details see Managed Groups.

Excessive Number of Updates

Can be caused by case sensitivity, use the ignoreCase attribute flag or transform the case to the desired format using ToLower, Title etc.