# Office AMS: User profile bulk property updateR

|  |  |
| --- | --- |
| Summary: | Applies to: |
| This sample shows how to bulk update user profile properties through the use of a CSV file and automate the extraction of user attributes from a given LDAP directory with authentication | * Office 365 Multi Tenant (MT) * Office 365 Dedicated (D) * SharePoint 2013 on-premises * LDAP directory extraction |
| Solution: | Core.BulkUserProfileUpdater, version 2.0 |
| Author: | Amar Bhogal (Microsoft) and Luke Bailey (Microsoft) |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS CODE IS PROVIDED \*AS IS\* WITHOUT WARRANTY OF  // ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING ANY  // IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR  // PURPOSE, MERCHANTABILITY, OR NON-INFRINGEMENT.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

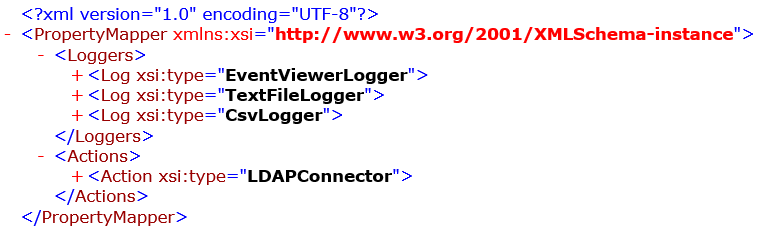
# Overview

The user profile bulk property utility is designed to replicate the synchronisation mechanism implemented by the advanced FIM connector for default attributes provisioned in Windows Azure AD. However, the utility will instead target a configurable and scalable list of custom attributes stored in an LDAP directory that will be mapped accordingly to custom SPO properties. The **System.DirectoryServices.Protocols namespace** will be used to establish a secure SSL connection to a given LDAP directory. A comprehensive explanation of this is provided at: <http://msdn.microsoft.com/en-us/library/bb332056.aspx>. The utility will batch users into multiple CSV files that are then injected directly into SharePoint Online through the **User Profile Web Service (ASMX)** that is referenced on MSDN at: <http://msdn.microsoft.com/en-us/library/aa981571(v=office.12).aspx>. Currently this is the only programmatic option for communicating with SPO and updating profile properties as both the REST and CSOM API’s provide read-only access. As a result of this direct communication with SPO, attributes will not be created in Windows Azure Active Directory and only exist within SPO.

# Scenario 1: bulk updating user profile properties

The single input file to the utility is **UserProfilePropertyMapper.xml** which defines the profile property mappings and connection details for the entire utility. The file is derived from an accompanying XSD schema.

The two primary root elements for the file are <loggers> and <actions>:



**Loggers**

Encapsulates the series of logging objects created by the utility through a subset of <Log> elements. There are three options to select from through the xsi:type attribute:

1. *EventViewerLogger* – captures exceptions from the utility and outputs to event logs
2. *TextFileLogger* – outputs verbose logging information to a text file for review
3. *CsvLogger* – generates a list of user accounts with success/failure outcome for review

**Actions**

Encapsulates the series of actions performed by the utility which can be nested for efficiency and simplification, there are two options to select from through the xsi:type attribute:

1. *LDAPConnector* – performs the automated extraction of specified attributes from LDAP directory
2. *UserProfileMapper* – performs the update of user properties from LDAP directory into SPO
   * 1. Log Elements

|  |  |  |
| --- | --- | --- |
| Parent Element | Child Element | Value and Description |
| TraceTypes | Trace | Ability to define level of logging for the current logging object.  **Values**: Error, Verbose |
| EmailSender | FromAddress  ToAddress  Host  Port | Complex type that encapsulate email details for the TextFileLogger or CsvLogger file to be sent to. If left out then no email will be generated |
| LogFileLocation | n/a | TextFileLogger and CsvLogger specific attribute that governs where the output file will be written to |
| EventSource | n/a | EventViewerLogger specific attribute that defines the source display name in the viewer |
| EventLogName | n/a | EventViewerLogger specific attribute that defines the categorisation of the log |
| EventId | n/a | EventViewerLogger specific attribute that defines the event identification of the log |

* + 1. Action Elements

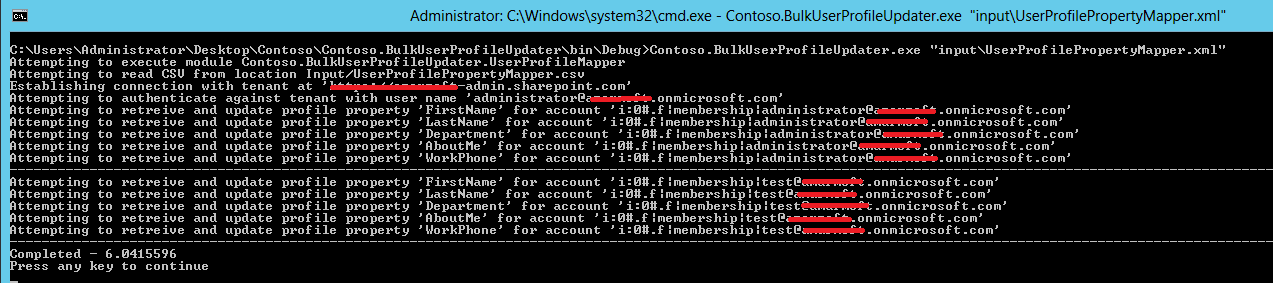
|  |  |  |
| --- | --- | --- |
| Parent Element | Child Element | Value and Description |
| CSVDirectoryLocation | n/a | Defines the location of the CSV containing the profile properties to update. The first index of the CSV file is used to identify the account to update |
| SPOAccountUPN | n/a | Value used in the construction of SPO user account names based on the msDS-PrincipalName extracted from LDAP directory. |
| ServerName | n/a | The name of the LDAP server that utility securely connects to, this is only used for display purposes in the logging entries |
| ServerIP | n/a | IP address of the targeted server to form a secure LDAP connection |
| PortNumber | n/a | Value of the secure SSL port used by the targeted server. |
| SearchRoot | n/a | Search scope value for the targeted OU hierarchy in the LDAP instance to optimise results returned |
| BatchAction | n/a | Mandatory flag to indicate the action being performed by the utility – this can be ‘delta’, ‘bulk’ or a specific LDAP query filter  **Value:** delta, bulk or LDAP filter query |
| SPOClaimsString | n/a | Value used in the construction of SPO user account names to prepend the claims string created by SPO  **Value**: i:0#.f|membership| |
| PageSize | n/a | The page size returned by the LDAP search response that also forms the CSV batch size of extracted users. This should be limited to the threshold of web requests to SPO  **Value**: 500 (can be increased accordingly) |
| CertificatePath | n/a | The full directory path to the LDAP certificate which should already be installed on the machine |
| QueryTimeout | n/a | Threshold limit in seconds for the LDAP search response query. This ensures that request times are within a known period  **Value:** 60 |
| UserName | n/a | The account name used to connect to LDAP |
| Password | n/a | The account password used to connect to LDAP |
| LDAPAuthType | n/a | Enumerated authentication type to connect to LDAP  **Value**: Basic |
| DirectoryType | n/a | Specify the type of directory to connect to  **Value**: DirectoryServer |
| ProtocolVers | n/a | The enumerated protocol version to form a secure connection with the LDAP server  **Value**: 3 |
| UserNameIndex | n/a | The index corresponding to the column in the CSV defining the user account on which the properties are updated. This must be mapped to the msDS-PrincipalName attribute from LDAP directory  **Value:** 0 |
| DeltaPeriod | n/a | Configurable delta period in days for the utility to search for user profile updates in LDAP directory  **Value:** 1 |
| Properties | Property | A collection of Properties, the name of the SPO User Profile Property (**attribute:name**) to update, the corresponding column (**attribute:index**) in the CSV file and the LDAP mapping value extracted from the LDAP directory (**attribute:mapping**) |
| TenantSiteUrl | n/a | The SSL secure administration specific URL for the O365 tenant |
| TenantAdminUserName | n/a | A dedicated O365 service account created to run the UPU that is a global administrator |
| TenantAdminPassword | n/a | Corresponding O365 user account password |
| SleepPeriod | n/a | Optional sleep period in seconds between connections to SPO in case service requests are throttled or limited by the ASMX service. |
| Actions | Action | Collection of recursive actions to run within the current action object so that properties only need to be defined at the root action rather than multiple |

To allow for future scalability and modification of the mappings between LDAP directories and SPO, the XML file exposes the complete lookup mechanism between the two directories. Within the <Properties> tag, each <Property> tag references a **name value (LDAP attribute)** and a **mapping value (SPO Property).** The utility will use this lookup table to optimise the LDAP search request by only requesting the listed attributes and then extract these in a sequential fashion. This will then from the CSV structure that is used to communicate with SPO. This scenario shows how to update user profile properties through the use of a CSV file

**Running the application**

Navigate to the directory of the **Contoso.BulkUserProfileUpdater.exe**, the executable requires a single input parameter which is the UserProfilePropertyMapper.xml



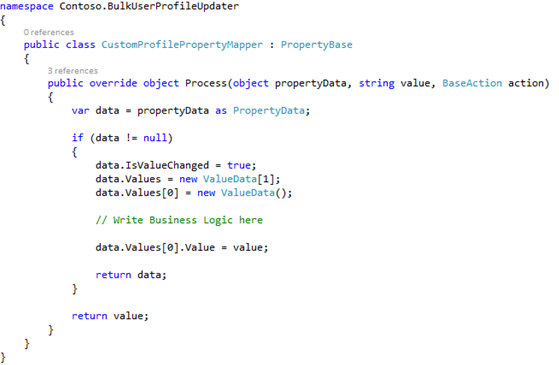


# Scenario 2: customising the profile mappings

The framework on which the BulkUserProfileUpdater allows you to create custom property mappers these are defined by the xsi:type on the Property node. By default the property mapper used is ProfilePropertyMapper.

For example, there maybe a scanerio whereby you might need to add custom logic before updating the profile property. The example below shows how this can be possible by overriding the relevant base classes.

Create a class and inherit PropertyBase. Override the ‘Process’ method



Within the UserProfilePropertyMapper.xml, for the property which requires the custom logic above to execute map the property to the class above. In the example below, when ‘WorkPhone’ property is mapped to the ‘CustomProfilePropertyMapper’ the custom logic will be executed.

