Create 2000 Unique Domain Accounts with Profile Photos for a Development SharePoint Environment

Introduction

I like to have a development environment that is as close to a production environment as possible. Having realistic development (or staging) environment helps business users visualise what an end product (or solution) will look like when deployed.

The following PowerShell (and accompanying name files) demonstrates creating 2000 unique Active Directory domain accounts, including setting different locations, departments, phone numbers and gender (male or female). Each domain account has a photo uploaded to Active Directory. Finally, SharePoint User Profile synchronisation is configured, to import the users and their photos.

This article makes use of name files from scrapmaker.com, and people pictures from fotolia.com

Download the full Script from the Microsoft TechNet Gallery

PowerShell script to create 2000 Active Directory uses with Profile Pictures

Process

1. Download the name files (female names, male names, surnames)
2. Format the documents using a notepad editor
3. Import the name files into PowerShell
4. Create a custom PSObject for holding people information
5. Use the name files to create 2000 unique users (1000 males and 1000 females)
6. Download 1000 male and 1000 female photos from fotolia
7. Create the 2000 users (using the Active Directory PowerShell module) and upload the photos for each new user account
8. Configure SharePoint User Profile Synchronisation to import the users from Active Directory, including each users Thumbnail photo
9. Run Update-SPProfilePhotoStore to create the profile photo variations
10. Index the user profiles and view the people results in SharePoint Search

1. Download the name files

To complete this exercise, you need a list of female names, male names and female names. I

searched the internet, and quickly came across a site called scrapmaker, which offers various lists, including the name lists I was after. They can be downloaded here: http://scrapmaker.com/dir/names

2. Formatting the documents

Open each name file, and remove any header information. The file should only contain a single column of names.

3. Import the name files into PowerShell

Here we import the three name files into PowerShell variables. Each variable will hold a collection of names that will be used to create the random male and female users.

```
$ffn = Get-Content "C:\Temp\NameDb\femalefirstnames.txt"
$mfn = Get-Content "C:\Temp\NameDb\malefirstnames.txt"
$ln = Get-Content "C:\Temp\NameDb\surnames.txt"
```

4. Create a custom PSObject for holding people information

Here we create a new psobject that will hold all of the user properties we will be randomly generating.

```
$userobject = New-Object psobject
$userobject | Add-Member -MemberType NoteProperty -Name "FirstName" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "LastName" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "SamAccount" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "Location" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "Country" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "CountryCode" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "City" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "Mobile" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "DDI" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "Ext" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "Department" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "JobTitle" -value ""
$userobject | Add-Member -MemberType NoteProperty -Name "Gender" -value ""
```

5. Use the name files to create 2000 unique users (approximate 1000 males and 1000 females)

Here we use the name files to create 2000 "users". Each user created is added to a userobject (defined in the above step). The PowerShell Get-Random cmdlet is used to randomly select

values for names and locations.

The script is commented, providing additional information.

```
#Create an array to hold the user objects
$users = $null;
$users = @();
#Get the number of names in each name file.
$ffnCount = $ffn.Count;
$mfnCount = $mfn.Count;
$lnCount = $ln.Count;
#Set a based number that will be used when creating the users Sam Account.
$sabase = 1000;
#Get the TextInfo class. This will be used with the name files to change the casing of names
to title case. E.g. john will be changed to John.
$cI = Get-Culture;
$tI = $cI.TextInfo;
$i=1;
#Create 2000 random users
while($i -le 2000){
 #Create a new user object
 $nu = $userobject | Select-Object *;
 #Set a random index value for the last name
 $lnIndex = Get-Random -minimum 0 -maximum ($lnCount -1);
 #Make sure the row (in the last names array) contains a value
 while($ln[$lnIndex].Length -eq 1)
 {
  $lnIndex = Get-Random -minimum 0 -maximum ($lnCount -1);
 }
 #Set the last name, using Title casing
 $nu.LastName = $tI.ToTitleCase($ln[$lnIndex].ToLower());
 #Create a unique value for the SAM Account
 $nu.SamAccount = ([String]::Format("u{0}",$sabase));
 #Randomly select the gender
 $gender = Get-Random -minimum 0 -maximum 1;
 #Set a random index value for the female name
 $ffnIndex = Get-Random -minimum 0 -maximum ($ffnCount -1);
 #Set a random index value for the male name
 $mfnIndex = Get-Random -minimum 0 -maximum ($mfnCount -1);
 if($gender -eq 0){
  #Make sure the row (in the male names array) contains a value
  while($mfn[$mfnIndex].Length -eq 1)
  {
   $mfnIndex = Get-Random -minimum 0 -maximum ($mfnCount -1);
  }
  #Set the fornename, using Title casing
  $nu.FirstName = $tI.ToTitleCase($mfn[$mfnIndex].ToLower());
  #Set the gender
  $nu.Gender = "Male";
 }
 else{
  #Make sure the row (in the female names array) contains a value
  while($ffn[$ffnIndex].Length -eq 1)
  {
   $ffnIndex = Get-Random -minimum 0 -maximum ($ffnCount -1);
  }
  #Set the fornename, using Title casing
  $nu.FirstName = $tI.ToTitleCase($ffn[$ffnIndex].ToLower());
```

```
  #Set the gender
  $nu.Gender = "Female";
 }
 #Use a random number to set the location of the user.
 $li = Get-Random -minimum 0 -maximum 100;
 if($li -le 25){$nu.Location =
"Melbourne";$nu.City="Melbourne";$nu.Country="Australia";$nu.CountryCode="AU";}
 if($li -gt 25 -and $li -le 40){$nu.Location = "Hong Kong";$nu.City="Hong
Kong";$nu.Country="Hong Kong";$nu.CountryCode="HK";}
 if($li -gt 40 -and $li -le 80){$nu.Location =
"London";$nu.City="London";$nu.Country="England";$nu.CountryCode="UK";}
 if($li -gt 80){$nu.Location = "New York";$nu.City="New York";$nu.Country="United States of
America";$nu.CountryCode="US";}
 #Set the users phone numbers using the unique base number $sabase
 $nu.DDI = ([String]::Format("555-{0}",$sabase));
 $nu.Ext = ([String]::Format("{0}",$sabase));
 $nu.Mobile = ([String]::Format("07555-66{0}",$sabase));
 #Set the Department of the user
 if($i -le 20){$nu.Department = "Executive"};
 if($i -gt 20 -and $i -le 100){$nu.Department = "Middle Management";$nu.JobTitle="Manager";};
 if($i -gt 100 -and $i -le 200){$nu.Department = "Accounts";$nu.JobTitle="Accountant";};
 if($i -gt 200 -and $i -le 250){$nu.Department = "Marketing";$nu.JobTitle="Marketing
Executive";};
 if($i -gt 250 -and $i -le 400){$nu.Department = "Sales";$nu.JobTitle="Salesman";};
 if($i -gt 400 -and $i -le 450){$nu.Department = "Information Technology";$nu.JobTitle="IT
Support";};
 if($i -gt 450 -and $i -le 475){$nu.Department = "Human Resources";$nu.JobTitle="HR
Support";};
 if($i -gt 475 -and $i -le 575){$nu.Department = "Engineering";$nu.JobTitle="Engineer";};
 if($i -gt 575 -and $i -le 675){$nu.Department = "Supervisors";$nu.JobTitle="Supervisor";};
 if($i -gt 675 -and $i -le 875){$nu.Department = "Team Leaders";$nu.JobTitle="Team Leader";}
 if($i -gt 875){$nu.Department = "Manufacturing"};
 $users += $nu;
 Write-Host "Added"$nu.FirstName $nu.LastName
 $sabase++;
 $i++;
}
```

6. Download 1000 male and 1000 female photos from fotolia

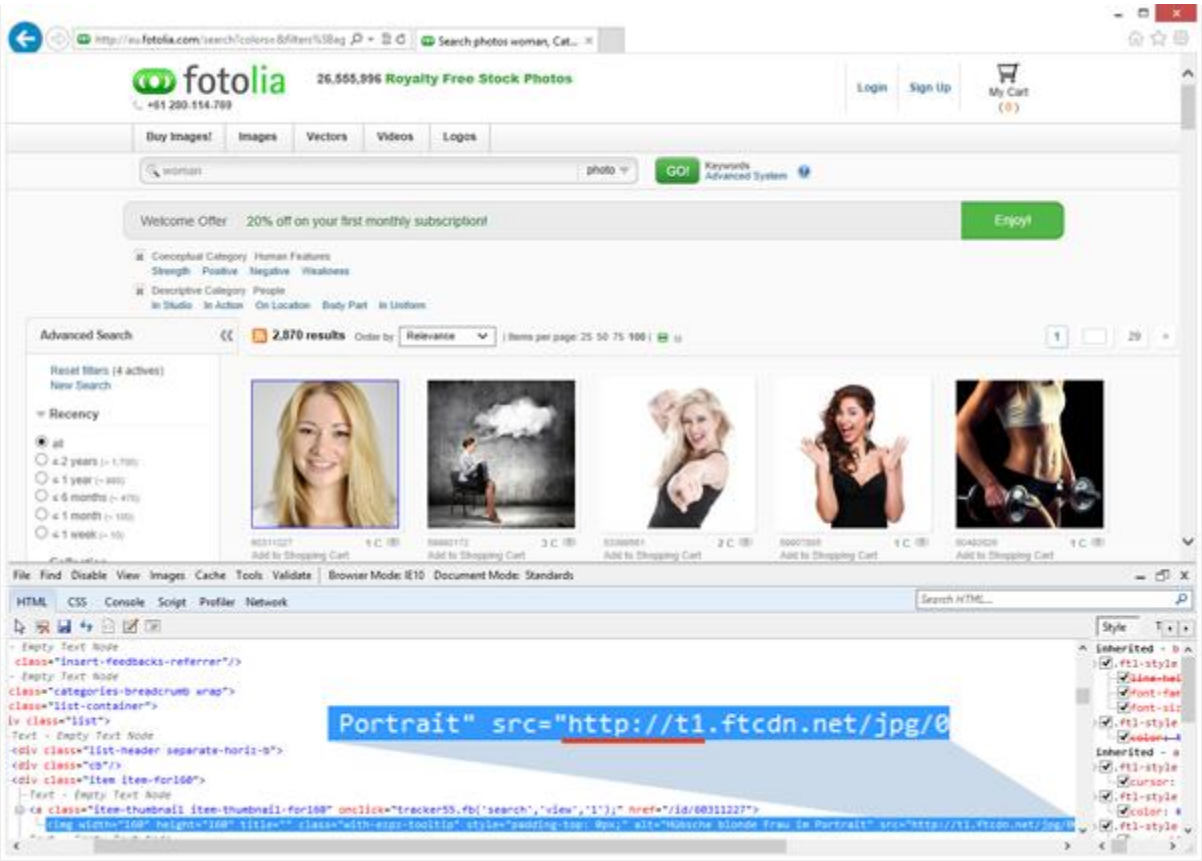
This part of the script uses the Internet Explorer object to search for photos on the Fotolia.com site.

Photos are downloaded from the Fotolia site by specifying a search query in the URL query string. Each page of results has approximately 100 images (search results). The Download-PhotoFromFotolia function takes a URL parameter, an internet explorer object, a base id (used to create unique file names) and directory path to save the images to. If there are no images returned from the search query, -1 is returned from the function.

```
function Download-PhotoFromFotolia{
```

```
 [CmdletBinding()]
  Param(
    [parameter(Mandatory=$true, ValueFromPipeline=$true)][String]$Url,
    [parameter(Mandatory=$true)][object]$InternetExplorer,
    [parameter(Mandatory=$true)][object]$BaseImageId,
    [parameter(Mandatory=$true)][object]$FileDirectoryPath
   )
 #Get the page (of search results) using the provided URL
 $InternetExplorer.Navigate($Url)
 while ($InternetExplorer.ReadyState -ne 4)
 {
  Write-Host "Downloading page. Please wait..." -foregroundcolor DarkYellow;
  sleep -Milliseconds 500
 }
 Write-Host "Getting a collection of images.";
 #Get a collection of all the images on the page
 $images = $InternetExplorer.Document.getElementsByTagName("img")
 Write-Host "Getting all of the portrait thumbnails.";
 #Get a collection of all the images with a src attribute that starts with http://t - this
will be the collection of thumbnail photos from the search results.
 #By examining the search page using Internet Explorer tools, you can see that all of the
thumbnail photos are on a CDN network, starting with the URL http://t1, or http://t2
 #Other images on the page (logos, etc), have a src attribute starting with http://s
 $imageCollection = $images | ?{$_.src -like "http://t*"}
 $wc = new-object System.Net.WebClient
 #Download each image and save it to the specified directory
 foreach($image in $imageCollection )
 {
  Write-Host "Downloading"$image.Src -foregroundcolor DarkYellow
  $filePath = ([String]::Format("{0}\{1}.jpg",$FileDirectoryPath,$BaseImageId));
  try
  {
   $wc.downloadfile($image.Src,$filePath);
   Write-Host "Successfully downloaded"$image.Src"to $filePath" -foregroundcolor Green
   $BaseImageId++;
  }
  catch
  {
   Write-Host "Failed to downloaded"$image.Src"to $filePath. Error:"$_.Exception.Message -
foregroundcolor Red
  }
 }
 #Return the baseimageid (with has been incremented), if images where found on the page.
 #If no images where found, return -1
 if($imageCollection.Count -eq 0)
 {return -1}
 else
 {return $BaseImageId;}
}
```

This image illustrates how to determine what to filter the images on. This is important, so that only photo thumbnail photos are downloaded.



This section of code sets the directories to save the images to. It then calls Download-PhotoFromFotolia repeated to download 2000 female and male images.

```
$page = 1;
$baseImageId = 1;
#Get a reference to Internet Explorer
$ie = New-Object -ComObject "InternetExplorer.Application"
#Set the directories for storing the female and male images (create the directories if they
don't exist)
$femailPhotoDirectory = "C:\temp\femalephotos";
if((Test-Path -Path $femailPhotoDirectory) -eq $false){New-Item -Path $femailPhotoDirectory -
ItemType Directory}
$mailPhotoDirectory = "C:\temp\malephotos";
if((Test-Path -Path $mailPhotoDirectory) -eq $false){New-Item -Path $mailPhotoDirectory -
ItemType Directory}
#Set the baseImageId - this is used to create a unique file name for each photo downloaded
$baseImageId = 1;
#Attempt to download 1000 female images. The search query specifies orientation = square,
contenttype = photo, using the keyword "woman"
#The Download-PhotoFromFotolia function returns -1 if there are no more images to download.
while($baseImageId -ge 1 -and $baseImageId -le 1000)
{
 $baseImageId = Download-PhotoFromFotolia -Url
([String]::Format("http://au.fotolia.com/search?colors=&filters%5Bage%5D=all&filters%5Bcollect
ion%5D=all&filters%5Bhas_releases%5D=true&filters%5Borientation%5D=square&filters%5Bmax_price_
xs%5D=all&filters%5Bmax_price_x%5D=&filters%5Bcontent_type%3Aphoto%5D=1&ca=3000000&cca=2000000
```

```
0&k=woman&offset={0}", $baseImageId)) -InternetExplorer $ie -BaseImageId $baseImageId -
FileDirectoryPath $femailPhotoDirectory
}
#Reset the baseimageid
$baseImageId = 1;
#Attempt to download 1000 male images. The search query specifies orientation = square,
contenttype = photo, using the keyword "man"
#The Download-PhotoFromFotolia function returns -1 if there are no more images to download.
while($baseImageId -ge 1 -and $baseImageId -le 1000)
{
 $baseImageId = Download-PhotoFromFotolia -Url
([String]::Format("http://au.fotolia.com/search?colors=&filters%5Bage%5D=all&filters%5Bcollect
ion%5D=all&filters%5Bhas_releases%5D=true&filters%5Borientation%5D=square&filters%5Bmax_price_
xs%5D=all&filters%5Bmax_price_x%5D=&filters%5Bcontent_type%3Aphoto%5D=1&ca=3000000&cca=2000000
0&k=man&offset={0}", $baseImageId)) -InternetExplorer $ie -BaseImageId $baseImageId -
FileDirectoryPath $mailPhotoDirectory
}
#Close the Internet Explorer application
$ie.Quit();
```

7. Create the 2000 users (using the Active Directory PowerShell module) and upload the photos for each new user account

The following code iterates through the array of user objects, passing the user object and a profile photo path to the Add-ActiveDirectoryUser function to create Active Directory user accounts.

```
$femalepictureIndex = 1;
$femaleMaxPhotos = (Get-ChildItem -Path $femailPhotoDirectory  -Filter *.jpg).Count
$malepictureIndex = 1;
$maleMaxPhotos = (Get-ChildItem -Path $mailPhotoDirectory  -Filter *.jpg).Count
foreach($u in $users)
{
 if($u.Gender -eq "Female")
 {
  if($femalepictureIndex -gt $femaleMaxPhotos){$femalepictureIndex=1;}
  $filePath = ([String]::Format("{0}\{1}.jpg",$femailPhotoDirectory,$femalepictureIndex))
  Add-ActiveDirectoryUser $u $filePath
  $femalepictureIndex++;
 }
 else
 {
  if($malepictureIndex -gt $maleMaxPhotos){$malepictureIndex=1;}
  $filePath = ([String]::Format("{0}\{1}.jpg",$mailPhotoDirectory,$malepictureIndex))
  Add-ActiveDirectoryUser $u $filePath
  $malepictureIndex++;
 }
}
```

The Add-ActiveDirectoryUser function. This function uses the custom user object to get the values for creating the new user account.

The function calls Ensure-OUExists to test if the destination OU exists. If it doesn't, it's created.

Finally, if the profilePhotoFilePath isn't null or empty, the function calls the Add-PhotoToUserAccount function to upload the profile photo for the user.

```
function Add-ActiveDirectoryUser{
 Param(
   [parameter(Mandatory=$true, ValueFromPipeline=$true)][object]$userObject,
   [parameter(Mandatory=$false)][object]$profilePhotoFilePath
  )
 $password = ConvertTo-SecureString -String "1HopeThi$isS3cure" -AsPlainText -Force
 $path =
([String]::Format("OU={0},OU={1},OU=Locations,DC=PANTS,DC=COM",$userObject.Department.Trim(),$
userObject.City.Trim()));
 Ensure-OUExists $path
 $currentUser = $null;
 try
 {
  $currentUser = Get-ADUSer $userObject.SamAccount -ErrorAction:SilentlyContinue;
  Write-Host "User"([String]::Format("{0}
{1}",$userObject.FirstName,$userObject.LastName))"exists." -foregroundcolor Green;
 }
 catch
 {
  Write-Host "User"([String]::Format("{0}
{1}",$userObject.FirstName,$userObject.LastName))"does not exist. The user will be created." -
foregroundcolor DarkYellow;
 }
 if($currentUser -eq $null){
  New-ADUser -UserPrincipalName $userObject.SamAccount -SamAccountName $userObject.SamAccount
-Name $userObject.SamAccount -City $userObject.City -AccountPassword $password  -Surname
$userObject.LastName -OfficePhone $userObject.DDI -MobilePhone $userObject.Mobile -GivenName
$userObject.FirstName -Division $userObject.Department -Department $userObject.Department  -
Enabled $true  -OtherAttributes
@{'ipPhone'=$userObject.Ext;'physicalDeliveryOfficeName'=$userObject.Location;'employeeType'=$
userObject.Gender;'co'=$userObject.Country} -EmployeeID $userObject.SamAccount -Path $path -
DisplayName ([String]::Format("{0} {1}",$userObject.FirstName,$userObject.LastName)) -Title
$userObject.JobTitle -Country $userObject.CountryCode;
  $currentUser = Get-ADUSer $userObject.SamAccount -ErrorAction:SilentlyContinue;
  Write-Host "Created"([String]::Format("{0} {1}",$userObject.FirstName,$userObject.LastName))
-foregroundcolor Green;
 }
 if(([String]::IsNullOrEmpty($profilePhotoFilePath))-eq $false)
 {
  Add-PhotoToUserAccount $currentUser $profilePhotoFilePath
  Write-Host "Added profile picture for "([String]::Format("{0}
{1}",$userObject.FirstName,$userObject.LastName)) -foregroundcolor Green;
 }
}
```

The Ensure-OUExists function

```
function Ensure-OUExists{
```

```
 Param(
   [parameter(Mandatory=$true, ValueFromPipeline=$true)][object]$path
  )
 $ou = $null;
 $domain = get-addomain
 if($domain.DistinguishedName -eq $path){return;}
 try
 {
  $ou = Get-ADOrganizationalUnit -Identity $path -ErrorAction SilentlyContinue;
 }
 catch
 {
  Write-Host "OU $path does not exist." -foregroundcolor DarkYellow;
 }
 if($ou -eq $null){
  $ouParent = [String]$path.Substring($path.IndexOf(",")+1);
  $ouName = [String]$path.Substring(0,$path.IndexOf(",")).Replace("OU=","");
  Ensure-OUExists $ouParent
  New-ADOrganizationalUnit -Name $ouName -Path $ouParent
  Write-Host "Created OU: $path" -foregroundcolor Green;
 }
}
```

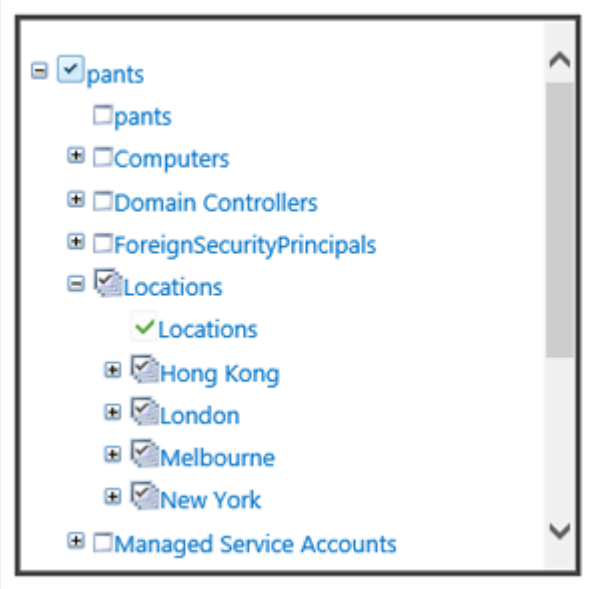
The Add-PhotoToUserAccount function

```
function Add-PhotoToUserAccount
{
 Param(
   [parameter(Mandatory=$true,
ValueFromPipeline=$true)][Microsoft.ActiveDirectory.Management.ADAccount]$Identity,
   [parameter(Mandatory=$true)][object]$fileName
  )
 try
 {
  $Identity | Set-ADUser -Replace @{thumbnailPhoto=([byte[]](Get-Content $fileName -Encoding
byte))}
  Write-Host "Set $fileName as the picture for"$Identity.Name"" -foregroundcolor Green;
 }
 catch
 {
  Write-Host "Failed to set $fileName as the picture for"$Identity.Name -foregroundcolor
DarkYellow;
 }
}
```

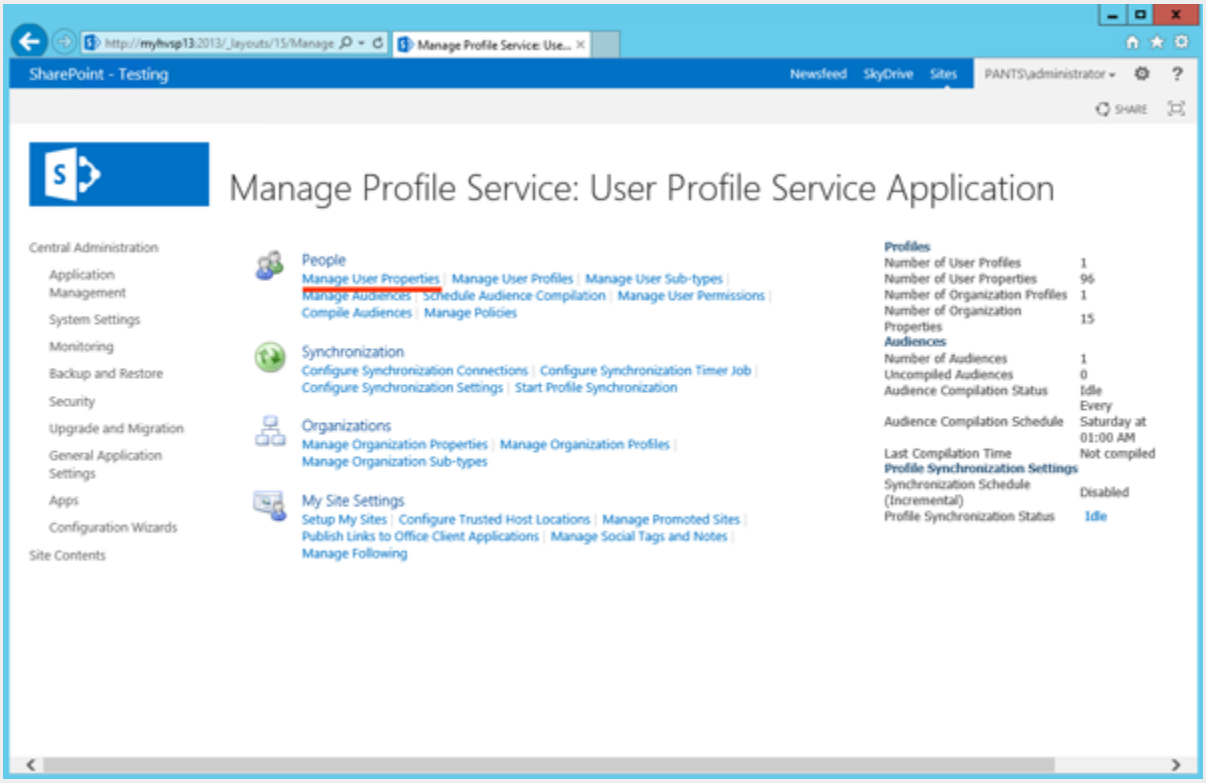
8. Configure SharePoint User Profile Synchronisation to import the users from Active Directory, including each users Thumbnail photo

In this step, we configure the User Profile Application to import the user accounts from Active Directory.

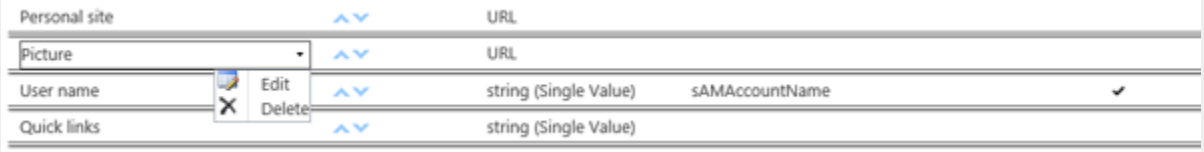
The Locations OU (which contains all of the user accounts in Sub-OU's) is selected as the source for importing user accounts.



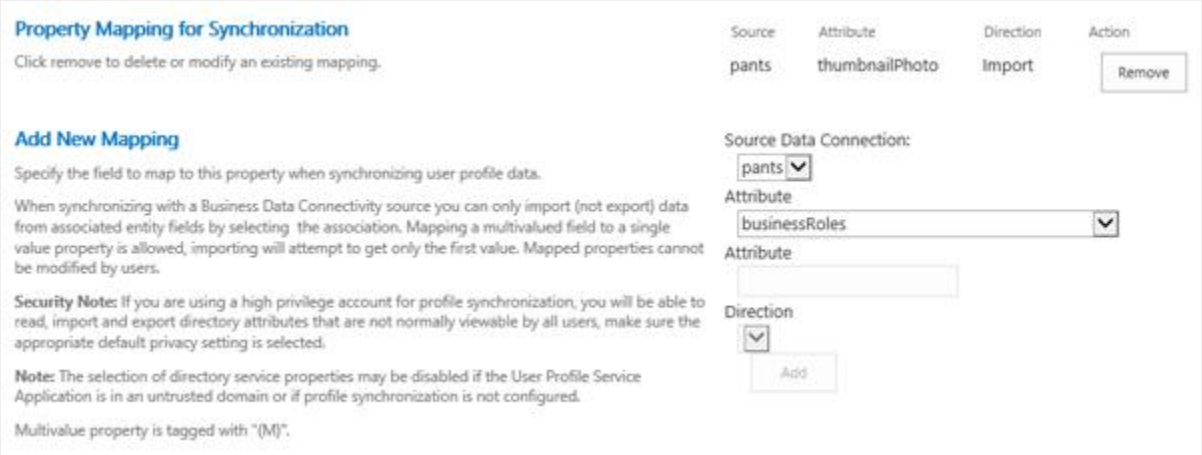
We also need to configure the User Profile properties, to ensure the Active Directory thumbnailPhoto is imported with the user accounts into the SharePoint User Profile store.



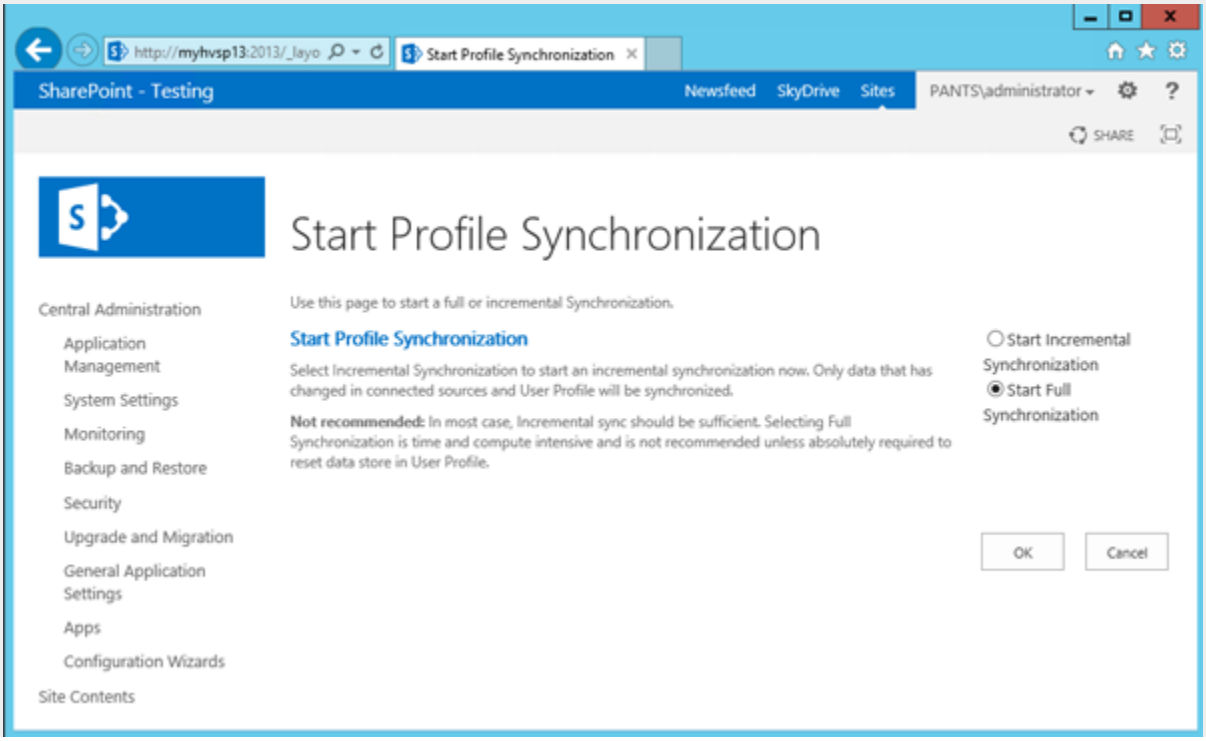
Edit the Picture profile property.



Import the Active Directory thumbnailPhoto attribute to the Picture profile property.



Finally, we need to start full profile synchronisation.



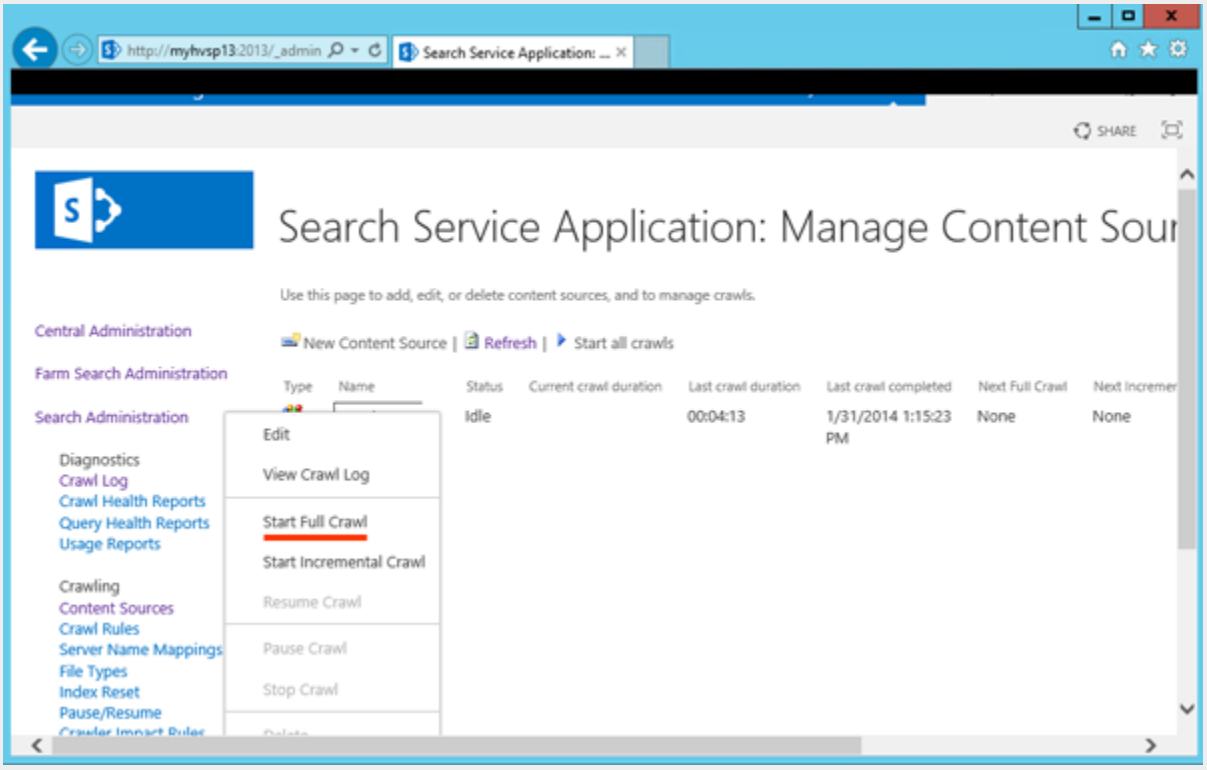
9. Run Update-SPProfilePhotoStore to create the profile photo variations

After the profile synchronisation has completed, run Update-SPProfilePhotoStore to create the profile photo variations.

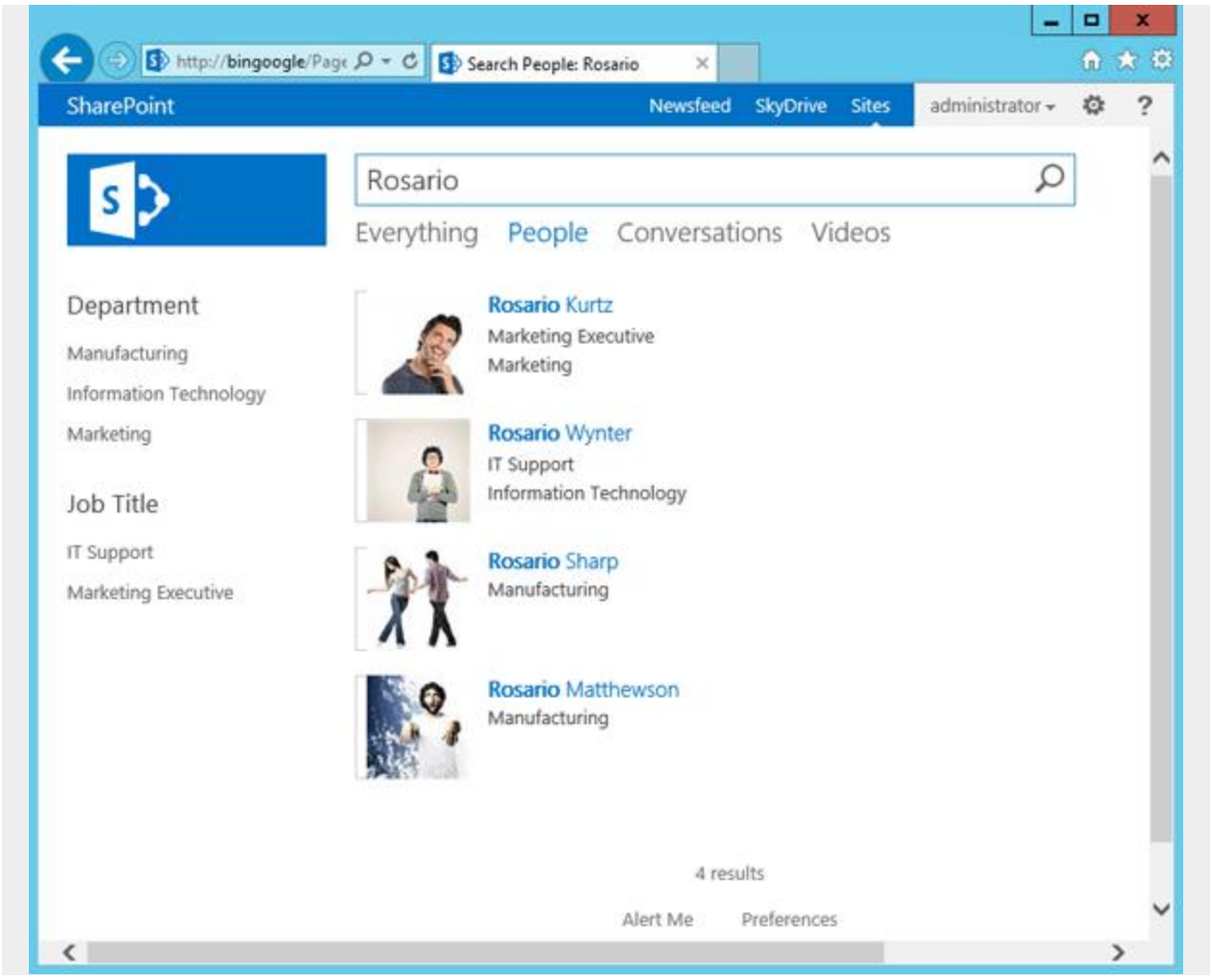
```
Update-SPProfilePhotoStore -CreateThumbnailsForImportedPhotos $true -MySiteHostLocation
"http://ms13"
```

10. Index the user profiles and view the people results in SharePoint Search

After the Update-SPProfilePhotoStore cmdlet complete (this can take a while), run a full crawl against the user profile store (from the SharePoint Search Application).



After the full crawl has completed, view the search results!


http://bingoogle/Page
Search People: Rosario
SharePoint
Newsfeed
SkyDrive
Sites
administrator
Rosario
Everything
People
Conversations
Videos
Department
Manufacturing
Information Technology
Marketing
Job Title
IT Support
Marketing Executive
Rosario Kurtz
Marketing Executive
Marketing
Rosario Wynter
IT Support
Information Technology
Rosario Sharp
Manufacturing
Rosario Matthewson
Manufacturing
4 results
Alert Me
Preferences