HEADLINE: STAFF DIRECTORY – DE-CLUTTERING

The Need

More often than not, we hear from nearly every organisation using SharePoint that they are interested in creating a single, easy place to access information of all employees.

Our solution

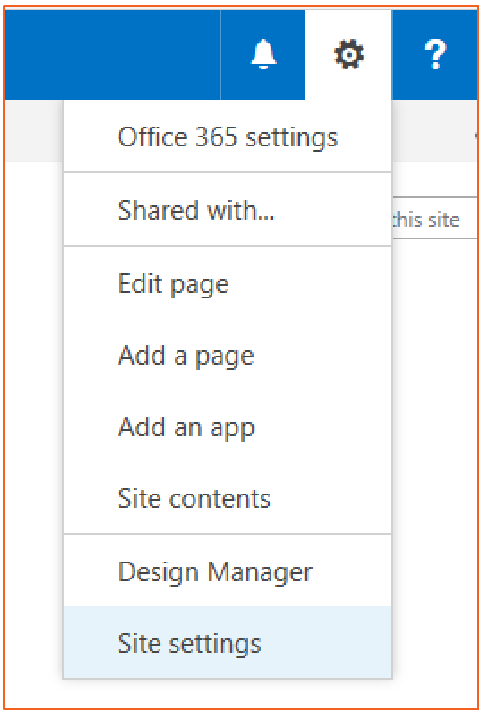
We have created a simple solution, leveraging SharePoint Online custom lists to store employee's information with the idea of share control with end users to arrange, sort or hide information as needed.

Our source of information of employees, in this example is AD and use PowerShell script to make sure all changes in AD are synchronised with SharePoint Online list. This synch will be unidirectional from AD to SharePoint Online as we don't want to allow end users to make changes in employee information in SharePoint Online list.

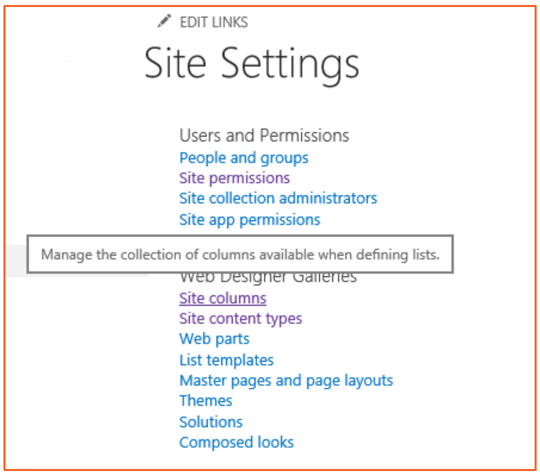
Creating the Site columns

First step is to create site columns in SharePoint Online and the selection criteria based on attributes of user in AD.

Go to SharePoint Online and "Site Settings"



Click on "Site Columns" under "Web Designer Galleries"


EDIT LINKS

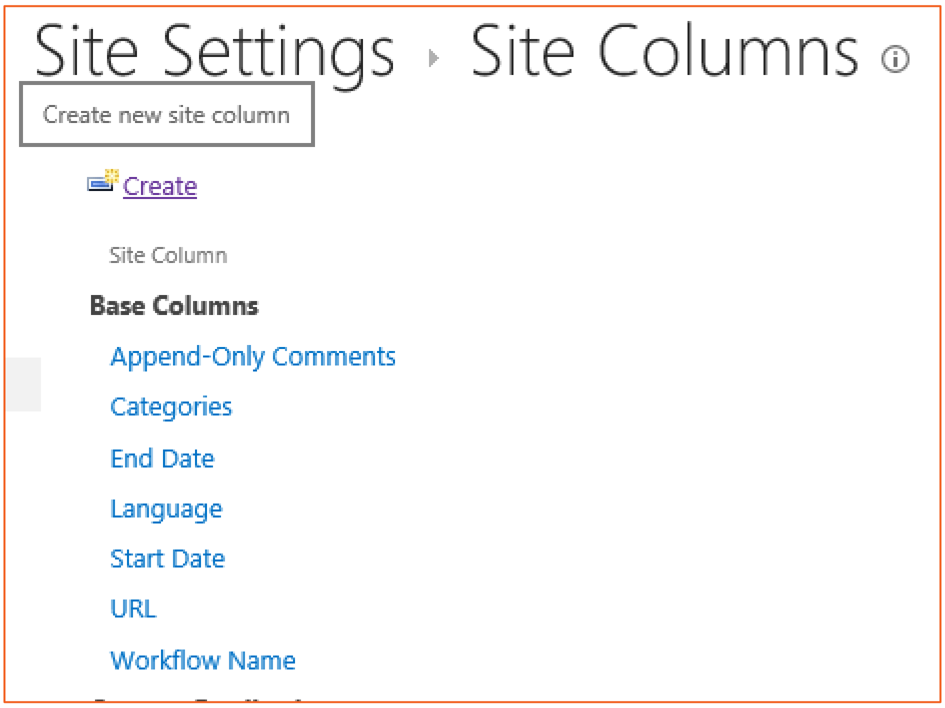
Site Settings

Users and Permissions
People and groups
Site permissions
Site collection administrators
Site app permissions

Manage the collection of columns available when defining lists.

Web Designer Galleries
Site columns
Site content types
Web parts
List templates
Master pages and page layouts
Themes
Solutions
Composed looks

Click on “Create” to create new site column


Site Settings ▸ Site Columns

Create new site column

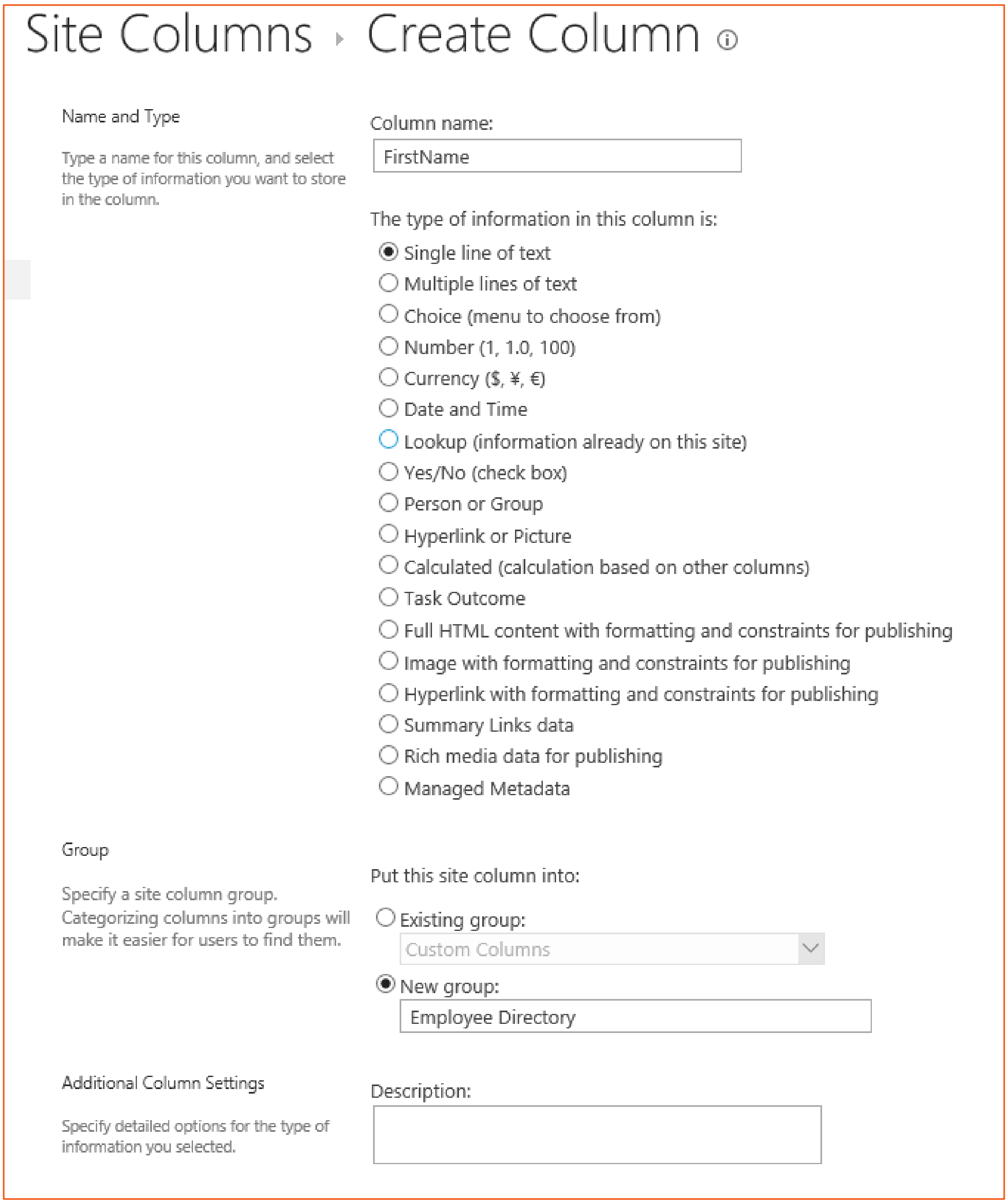
Create

Site Column

Base Columns
Append-Only Comments
Categories
End Date
Language
Start Date
URL
Workflow Name

Enter column name. In our example, it is *FirstName*.

Select “Single Line of Text” as type and under “Group”, select “New Group” and enter “Employee Directory” and click “OK”



Site Columns ‣ Create Column

Name and Type

Type a name for this column, and select the type of information you want to store in the column.

Column name: FirstName

The type of information in this column is:

- [✓ Single line of text]
- Multiple lines of text
- Choice (menu to choose from)
- Number (1, 1.0, 100)
- Currency ($, ¥, €)
- Date and Time
- Lookup (information already on this site)
- Yes/No (check box)
- Person or Group
- Hyperlink or Picture
- Calculated (calculation based on other columns)
- Task Outcome
- Full HTML content with formatting and constraints for publishing
- Image with formatting and constraints for publishing
- Hyperlink with formatting and constraints for publishing
- Summary Links data
- Rich media data for publishing
- Managed Metadata

Group

Specify a site column group. Categorizing columns into groups will make it easier for users to find them.

Put this site column into:

- Existing group: Custom Columns
- [✓ New group:] Employee Directory

Additional Column Settings

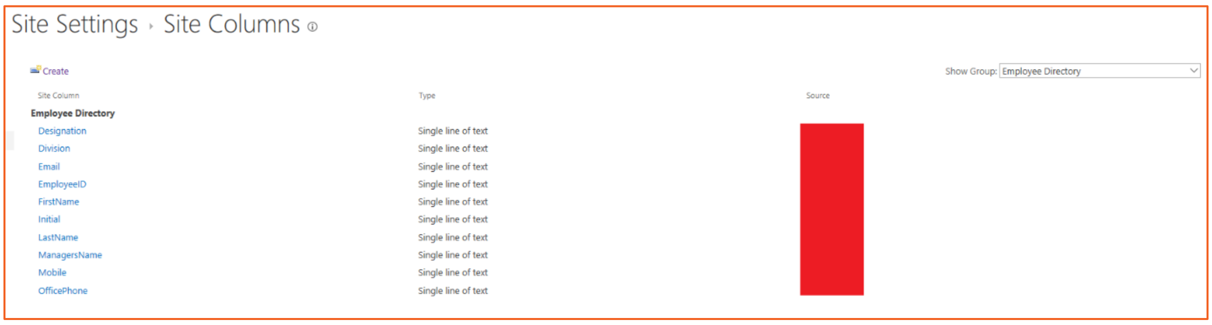
Specify detailed options for the type of information you selected.

Description:

Repeat above steps to create site columns for user's attribute mentioned below and make sure to select "Employee Directory" as group instead of creating new group.

- LastName – Type – Single line of text
- Initial– Type – Single line of text
- Email – Type – Hyperlink or Picture
- Designation – Type – Single line of text
- Division – Type – Single line of text
- ManagersName – Type – Single line of text
- OfficePhone – Type – Single line of text
- Mobile – Type – Single line of text
- EmployeeID – Type – Single line of text

Once all site columns are created, group "Employee Directory" will look something like:



Site Settings › Site Columns

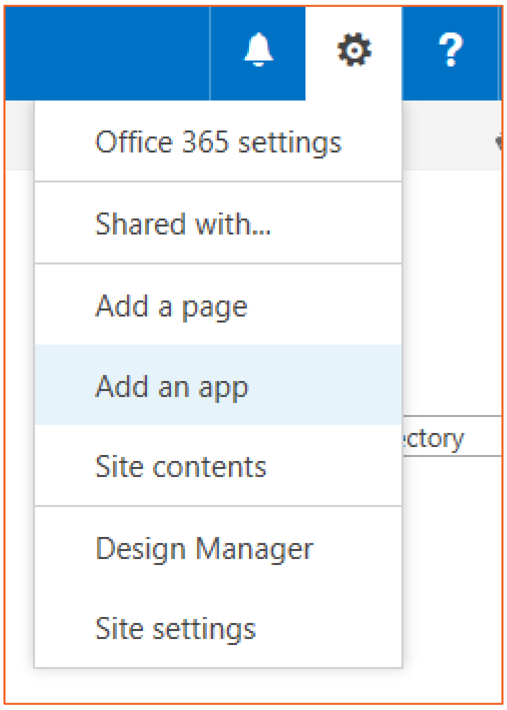
Create

Show Group: Employee Directory

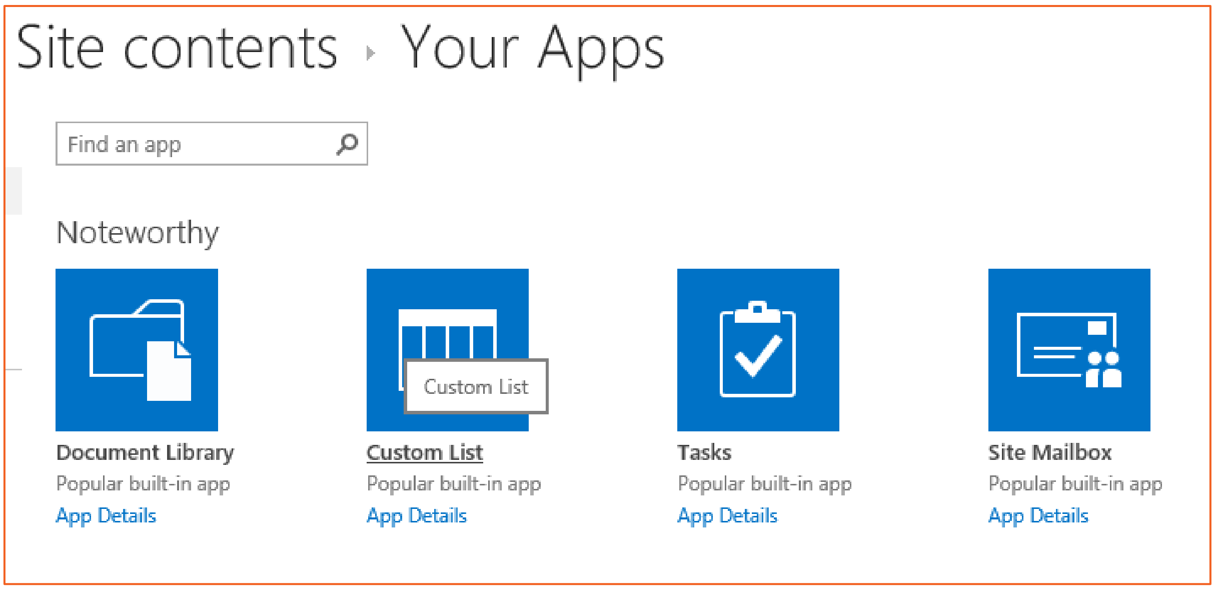
Site Column	Type	Source
Employee Directory		
Designation	Single line of text	[redacted]
Division	Single line of text	[redacted]
Email	Single line of text	[redacted]
EmployeeID	Single line of text	[redacted]
FirstName	Single line of text	[redacted]
Initial	Single line of text	[redacted]
LastName	Single line of text	[redacted]
ManagersName	Single line of text	[redacted]
Mobile	Single line of text	[redacted]
OfficePhone	Single line of text	[redacted]

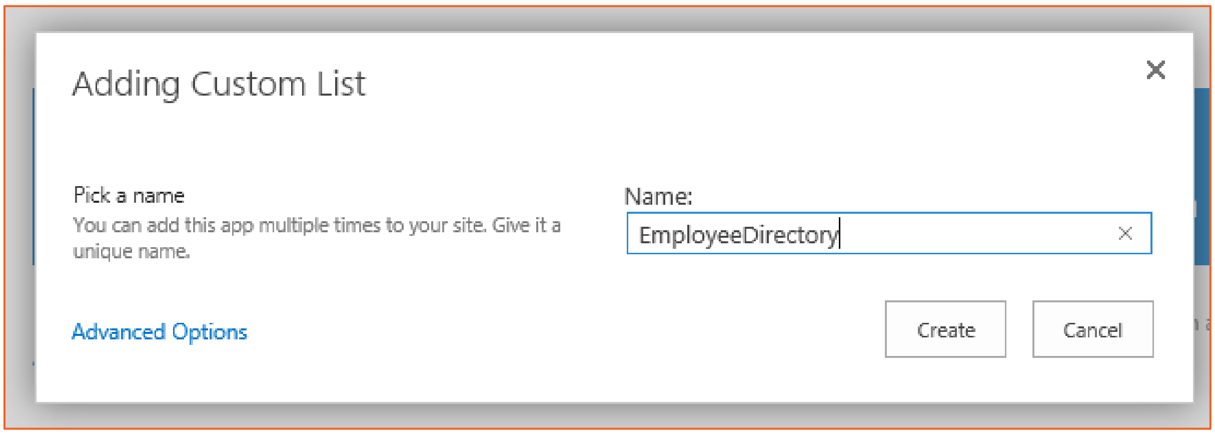
Creating a Custom list

Next step is to create custom list. Click on "Add an app"



Select “Custom List”. Specify name of the custom list to “Employee Directory”



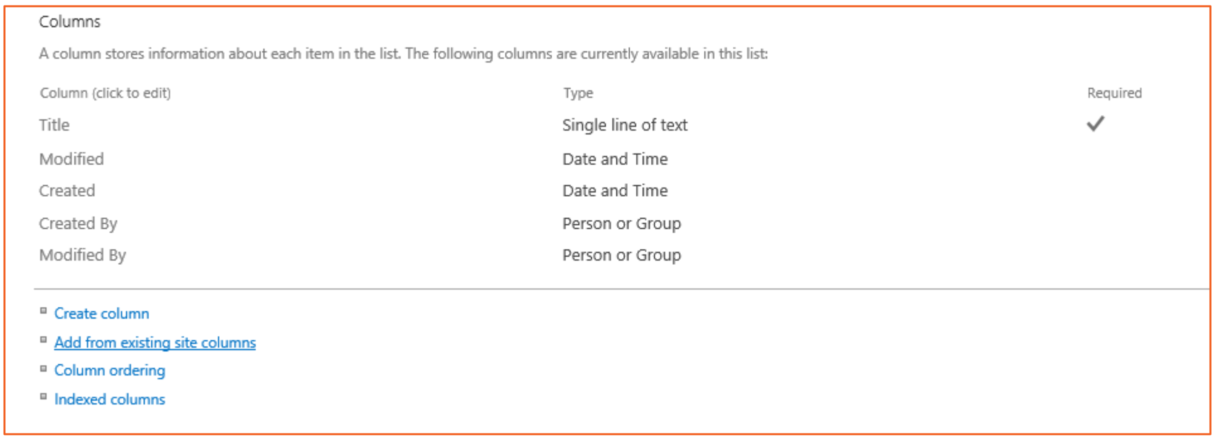

Adding Custom List

Pick a name

You can add this app multiple times to your site. Give it a unique name.

Name: EmployeeDirectory

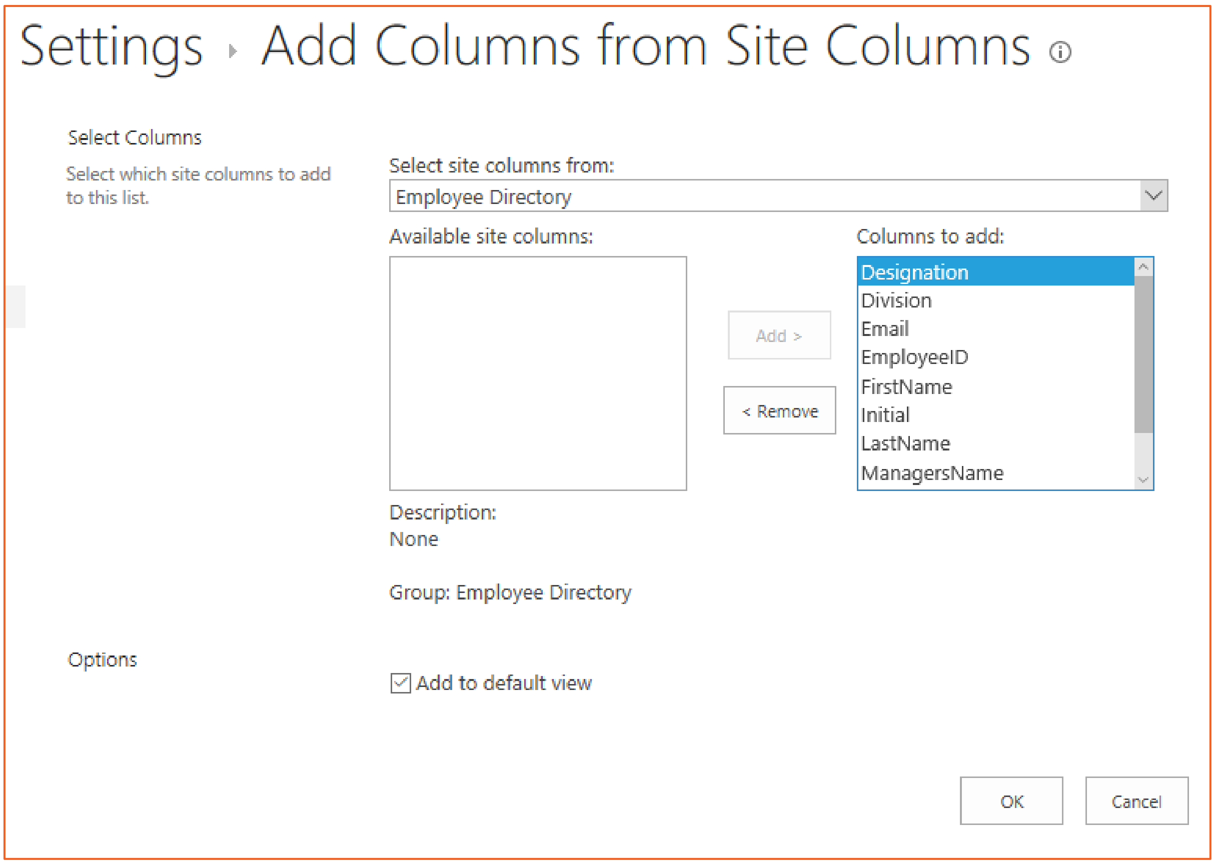
Go to “Employee Directory” list settings and click “Add from existing site columns”


Columns

A column stores information about each item in the list. The following columns are currently available in this list:

Column (click to edit)	Type	Required
Title	Single line of text	✓
Modified	Date and Time	
Created	Date and Time	
Created By	Person or Group	
Modified By	Person or Group	

Select group “Employee Directory” and add all columns.



Settings › Add Columns from Site Columns

Select Columns

Select which site columns to add to this list.

Select site columns from: Employee Directory

Available site columns:

Columns to add:
- Designation
- Division
- Email
- EmployeeID
- FirstName
- Initial
- LastName
- ManagersName

Description:
None

Group: Employee Directory

Options

☑ Add to default view

Great!

So far, we have created site columns and added the site columns in custom list….let’s get into the fun bit :-)

The PowerShell script

Next step is to create PowerShell script. We have divided PowerShell script into two different scripts:

- First script will run and get data from AD and sych with SharePoint Online list “Employee Directory”
- Second script gets disable users (employees) from AD and delete them from SharePoint Online list “Employee Directory”

1 Let’s start with first script. We have divided script in three functions: First for logging, second for getting the data from AD and third for synch’ing users (employees) data with SharePoint Online list “Employee Directory”

A The Function below log script progress and you need to specify LogPath although, we also specified path when logging:

```
function Write-Log{
    [CmdletBinding()]
    Param
    (
        [Parameter(Mandatory=$true,
                   ValueFromPipelineByPropertyName=$true)]
        [ValidateNotNullOrEmpty()]
        [Alias("LogContent")]
        [string]$Message,

        [Parameter(Mandatory=$false)]
        [Alias('LogPath')]
        [string]$Path='C:\ED\EmployeeDirectory.log',

        [Parameter(Mandatory=$false)]
        [ValidateSet("Error","Warn","Info")]
        [string]$Level="Info"
    )

    Begin{

    }
    Process{

        # If attempting to write to a log file in a folder/path that doesn't exist
create the file including the path.
        if (!(Test-Path $Path))
        {
            $NewLogFile = New-Item $Path -Force -ItemType File
        }
        # Format Date for our Log File
        $FormattedDate = Get-Date -Format "yyyy-MM-dd HH:mm:ss"

        # Write message to error, warning, or verbose pipeline and specify
$LevelText
        switch ($Level) {
            'Error' {
                Write-Error $Message
                $LevelText = 'ERROR:'
                }
            'Warn' {
                Write-Warning $Message
                $LevelText = 'WARNING:'
                }
            'Info' {
                Write-Verbose $Message
                $LevelText = 'INFO:'
                }
            }

        # Write log entry to $Path
        "$FormattedDate $LevelText $Message" | Out-File -FilePath $Path -Append
    }
    End{

    }
}
```

B The Function below gets data from AD. You need to specify values of two parameter based upon environment. One is `-SearchBase` and other is `-Server`. Your IT can help on this.

```
function GetUserFromAD{

        $employeeDirectory = Get-ADUser -filter 'enabled -eq $True'
        -Properties
                displayname,
                initials,
                employeeid,
                department,
                description,
                officephone,
                othertelephone,
                mail,
                office,
                extensionAttribute13,
                whenchanged,
                mobile,
                manager,
                postalCode
       -SearchBase "OU=General,OU=Standard Accounts,OU=Stand-
       ard,DC=XXXX,DC=XXXXXX,DC=com"
       -Server XXXXXX  | where $_.employeeid -gt ""  |
        Select-Object
                DisplayName,
                enabled,
                Surname,
                GivenName,
                Initials,
                Department,
                EmployeeID,
                UserPrincipalName,
                OfficePhone,
                mobile,
                manager,
                whenchanged,
        @{expression={$_.otherTelephone};label="Extension";}

        return $employeeDirectory
}
```

C This third Function will synch employee information retrieved from above function with SharePoint Online list "Employee Directory". You need to specify

I Update log file path - `$LogFile`

II Make sure that
`Microsoft.SharePoint.Client.dll`
`Microsoft.SharePoint.Client.Runtime.dll`
are at path `C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\16\ISAPI\`

III Find `$Login` and `$Password` and update with your login and password for SharePoint Online site

IV Find statement `$totaltime.Days -le 2`. This condition specific number of days to look for changes in attributes of user AD profile

V Used regex to find manager name - `$row.Manager,'\=([^\=]*)\,').Groups[1].Value.Trim().` `Please update accordingly.`

```
function UpLoadToEmployeeDirectory{
    [CmdletBinding()]
    param(
        [Parameter(Mandatory=$true,
                   ValueFromPipeline=$true,
                   ValueFromPipelineByPropertyName=$true)]
        [String]
        $siteURL,
        [Parameter(Mandatory=$true,
                  ValueFromPipeline=$true,
                  ValueFromPipelineByPropertyName=$true)]
        [String]
        $EDListName

    )

    Begin{

        Add-Type -Path "C:\Program Files\Common Files\Microsoft Shared\Web Server 
Extensions\16\ISAPI\Microsoft.SharePoint.Client.dll"
        Add-Type -Path "C:\Program Files\Common Files\Microsoft Shared\Web Server 
Extensions\16\ISAPI\Microsoft.SharePoint.Client.Runtime.dll"

        #create log file
        $LogFileName = Get-Date -Format "yyyy-MM-dd HH-mm-ss"
        $LogFile = 'C:\ED\EmployeeDirectory_' + $LogFileName + '.log'
        $LogFilePath = $LogFile

        #Creating Data Table
        $dtTable = New-Object System.Data.DataTable
        $col1 = New-Object System.Data.DataColumn Title,([string])
        $col2 = New-Object System.Data.DataColumn SurName,([string])
        $col3 = New-Object System.Data.DataColumn GivenName,([string])
        $col4 = New-Object System.Data.DataColumn Initials,([string])
        $col5 = New-Object System.Data.DataColumn Department,([string])
        $col6 = New-Object System.Data.DataColumn Position,([string])
        $col7 = New-Object System.Data.DataColumn EmployeeID,([string])
        $col8 = New-Object System.Data.DataColumn eMail,([string])
        $col9 = New-Object System.Data.DataColumn OfficePhone,([string])
        $col10 = New-Object System.Data.DataColumn Mobile,([string])
        $col11 = New-Object System.Data.DataColumn Manager,([string])
        $col12 = New-Object System.Data.DataColumn ItemID,([string])

        $dtTable.Columns.Add($col1);
        $dtTable.Columns.Add($col2);
        $dtTable.Columns.Add($col3);
        $dtTable.Columns.Add($col4);
        $dtTable.Columns.Add($col5);
        $dtTable.Columns.Add($col6);
        $dtTable.Columns.Add($col7);
        $dtTable.Columns.Add($col8);
        $dtTable.Columns.Add($col9);
        $dtTable.Columns.Add($col10);
        $dtTable.Columns.Add($col11);
        $dtTable.Columns.Add($col12);
    }

    process{

        # Starting Log
        Write-Log -Message 'Script Started ... ' -Path $LogFilePath


        #Retrieving user data from AD
        Write-Log -Message "Retrieving user data from AD"  -Path $LogFilePath
        try
        {
            $tblData = GetUserFromAD
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to retrieve user data from AD"  -Path 
$LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level 
Error
            Exit
        }
        Write-Log -Message "Retrieved user data from AD"  -Path $LogFilePath

        #Read password and key
        try
        {
            $Login = 'XXXXXX@XXXXXXX.com'
            $Password = "XXXXXXX"
            $Password1 = ConvertTo-SecureString $Password -AsPlainText -Force
        }
        catch [Exception]
        {
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level
Error
            Exit
        }
        Write-Log -Message "Retrieved password from file"  -Path $LogFilePath

        #Getting Context of site collection
        try
        {
            $Context = New-Object Microsoft.SharePoint.Client.ClientContext($siteURL)
            $Creds = New-Object
Microsoft.SharePoint.Client.SharePointOnlineCredentials($Login,$Password1)
            $Context.Credentials = $Creds
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to get Context of SharePoint Online"  -Path
$LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level
Error
            Exit
        }
        Write-Log -Message "Site Collection Context Obtained"  -Path $LogFilePath

        #Get Employee Directory
        try
        {
            $PDList = $Context.Web.Lists.GetByTitle($EDListName)
            $Context.Load($PDList)
            $Context.ExecuteQuery()
            Write-Log -Message 'SharePoint Employee Directory List loaded' -Path
$LogFilePath
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to list Policy Directory List form SharePoint
Online"  -Path $LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level
Error
            Exit
        }
        Write-Log -Message "Employee Directory context obtained"  -Path $LogFilePath

        #Download all items from Employee Directory
        try
        {
            $qry = New-Object Microsoft.SharePoint.Client.CamlQuery
            $qry.ViewXml = "<View><Query><Where></Where><OrderBy></OrderBy></Query>" +
                                "<ViewFields>" +
                                "<FieldRef Name='Title' /> " +
                                "<FieldRef Name='LastName' /> " +
                                "<FieldRef Name='FirstName' /> " +
                                "<FieldRef Name='Initial' /> " +
                                "<FieldRef Name='Division' /> " +
                                "<FieldRef Name='Designation' /> " +
                                "<FieldRef Name='EmployeeID' /> " +
                                "<FieldRef Name='Email' /> " +
                                "<FieldRef Name='OfficePhone' /> " +
                                "<FieldRef Name='Mobile' /> " +
                                "<FieldRef Name='ManagersName' /> " +
                                 "</ViewFields>" +
                            "</View>"

            $items = $PDList.GetItems($qry)
            $Context.Load($items)
            $Context.ExecuteQuery()
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to get all list items from Employee Directory"
-Path $LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level
Error
            Exit
        }
        Write-Log -Message "All Items from Employee Directory retrieved"  -Path
$LogFilePath

        #Populating DataTable from Employee Directory List Items
        try
        {
            foreach($item in $items)
            {
                $row = $dtTable.NewRow()
                $row.Title =
if([string]::IsNullOrEmpty($item["Title"])){"####"}else{$item["Title"]}
                $row.SurName =
if([string]::IsNullOrEmpty($item["SurName"])){"####"}else{$item["LastName"]}
                $row.GivenName =
if([string]::IsNullOrEmpty($item["GivenName"])){"####"}else{$item["FirstName"]}
                $row.Initials =
if([string]::IsNullOrEmpty($item["Initials"])){"####"}else{$item["Initial"]}
                $row.Department =
if([string]::IsNullOrEmpty($item["ol_Department"])){"####"}else{$item["Division"]}
                $row.Position =
if([string]::IsNullOrEmpty($item["Position"])){"####"}else{$item["Designation"]}
                $row.EmployeeID = $item["EmployeeID"]
                $row.OfficePhone =
if([string]::IsNullOrEmpty($item["OfficePhone"])){"####"}else{$item["OfficePhone"]}
                $row.Mobile =
if([string]::IsNullOrEmpty($item["Mobile"])){"####"}else{$item["Mobile"]}
                $row.eMail = $item["Email"].Description
                $row.Manager =
if([string]::IsNullOrEmpty($item["ManagersName"])){"####"}else{$item["ManagersName"]}
                $row.ItemID = $item["ID"]
                $dtTable.Rows.Add($row)
            }
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to populate data table from Employee Directory
List itesm"  -Path $LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level
Error
            Exit
        }
        Write-Log -Message "Data Table populated from Employee Directory List Items"  -
Path $LogFilePath

        #Entry to log file
        $startTime = Get-Date
        Write-Log -Message "Loop start Time $startTime" -Path $LogFilePath

        foreach ($row in $tblData)
                {
                    $totaltime = NEW-TIMESPAN -Start $row.whenchanged -End $startTime

                    if ($totaltime.Days -le 2)
                    {
                        $strEmployeeID = "EmployeeID='" + $row.EmployeeID + "'"
                        $dtrow = $dtTable.Select("EmployeeID='" + $row.EmployeeID +
"'")

                        if($dtrow.Count -eq 1)
                        {
                            $strRowTitle =
if([string]::IsNullOrEmpty($row.DisplayName)){"####"}else{$row.DisplayName}
                            $strRowSurName =
if([string]::IsNullOrEmpty($row.SurName)){"####"}else{$row.LastName}
                            $strRowGivenName =
if([string]::IsNullOrEmpty($row.GivenName)){"####"}else{$row.FirstName}
                            $strRowInitials =
if([string]::IsNullOrEmpty($row.Initials)){"####"}else{$row.Initial}
```

```
                              $strRowDepartment =
if([string]::IsNullOrEmpty($row.Department)){"####"}else{$row.division}
                              $strRowDescription =
if([string]::IsNullOrEmpty($row.Description)){"####"}else{$row.Designation}
                              $strRowOfficePhone =
if([string]::IsNullOrEmpty($row.OfficePhone )){"####"}else{$row.OfficePhone }
                              $strRowMobile = if([string]::IsNullOrEmpty($row.mobile
)){"####"}else{$row.mobile }
                              $strManager =
if([string]::IsNullOrWhiteSpace($row.Manager)){"####"}else{[regex]::matches($row.Manage
r, '\=([^\=]*)\,').Groups[1].Value.Trim()}

                              if(
                                  ($strRowTitle -eq $dtrow.Title) -and
                                  ($strRowSurName -eq $dtrow.LastName) -and
                                  ($strRowGivenName -eq $dtrow.FirstName) -and
                                  ($strRowInitials -eq $dtrow.Initial) -and
                                  ($strRowDepartment -eq $dtrow.Division) -and
                                  ($strRowDescription -eq $dtrow.Designation) -and
                                  ($row.EmployeeID -eq $dtrow.EmployeeID) -and
                                  ($strRowOfficePhone -eq $dtrow.OFficePhone) -and
                                  ($strRowMobile -eq $dtrow.mobile) -and
                                  ($row.UserPrincipalName -eq $dtrow.Email) -and
                                  ($strManager -eq $dtrow.Manager)
                              )
                              {
                                  $days = $totaltime.Days
                                  Write-Log -Message "Ignoring $row.DisplayName as all
values are same although updated $days days ago"  -Path $LogFilePath
                              }
                              else
                              {
                                  Write-Log -Message "Updating $row.DisplayName" -Path
$LogFilePath
                                  try
                                  {

                                      $spoListItem = $PDList.GetItemById($dtrow.ItemID)
                                      $spoListItem["Title"]=$row.DisplayName
                                      $spoListItem["LastName"]=$row.LastName
                                      $spoListItem["FirstName"]=$row.FirstName
                                      $spoListItem["Initial"]=$row.Initial
                                      $spoListItem["Division"]=$row.Division
                                      $spoListItem["Designation"]=$row.Designation
                                      $spoListItem["EmployeeID"]=$row.EmployeeID
                                      $spoListItem["Email"]= "mailto:" +
$row.UserPrincipalName + ", " + $row.UserPrincipalName
                                      $spoListItem["OfficePhone"]=$row.OfficePhone
                                      $spoListItem["Mobile"]=$row.mobile
                                      $spoListItem["ManagersName"]=$strManager
                                      $spoListItem.Update()
                                      $Context.ExecuteQuery()
                                  }
                                  Catch [Exception]
                                  {
                                      #log exception
                                      Write-Log -Message "Not able to update user data
for $row.DisplayName"  -Path $LogFilePath -Level Error
                                      $strException = $_.Exception
                                      Write-Log -Message "Exception - $strException" -
Path $LogFilePath -Level Error
                                  }
                              }
                          }
                          else
                          {
                              Write-Log -Message "Adding $row.DisplayName" -Path
$LogFilePath
                              try
                              {
                                  $strManager =
if(![string]::IsNullOrWhiteSpace($row.Manager)){[regex]::matches($row.Manager,
'\=([^\=]*)\,').Groups[1].Value.Trim()}
                                  $spoListItemCreationInformation = New-Object
Microsoft.SharePoint.Client.ListItemCreationInformation

$spoListItem=$PDList.AddItem($spoListItemCreationInformation)
                                  $spoListItem["Title"]=$row.DisplayName
                                  $spoListItem["LastName"]=$row.Lastname
```

```
                            $spoListItem["FirstName"]=$row.FirstName
                            $spoListItem["Initials"]=$row.Initial
                            $spoListItem["ol_Department"]=$row.Division
                            $spoListItem["Designation"]=$row.Designation
                            $spoListItem["EmployeeID"]=$row.EmployeeID
                            $spoListItem["Email"]= "mailto:" +
$row.UserPrincipalName + ", " + $row.UserPrincipalName
                            $spoListItem["OfficePhone"]=$row.OfficePhone
                            $spoListItem["Mobile"]=$row.mobile
                            $spoListItem["ManagersName"]=$strManager
                            $spoListItem.Update()
                            $Context.ExecuteQuery()
                        }
                        Catch [Exception]
                        {
                            #log exception
                            Write-Log -Message "Not able to add user data for
$row.DisplayName"  -Path $LogFilePath -Level Error
                            $strException = $_.Exception
                            Write-Log -Message "Exception - $strException" -Path
$LogFilePath -Level Error
                        }
                    }
                }
                else{
                    $days = $totaltime.Days
                    Write-Log -Message "Ignoring $row.DisplayName as last modified
$days days ago"  -Path $LogFilePath
                }
            }

        $endTime = Get-Date
        $totallooptime = NEW-TIMESPAN -Start $startTime -End $endTime
        Write-Log -Message "Loop End Time $endTime " -Path $LogFilePath
        $strTotalTime = "Total Loop Time - " + $totallooptime.Hours + " hours, " +
$totallooptime.Minutes + " minutes and "  + $totallooptime.Seconds + " seconds"
        Write-Log -Message $strTotalTime -Path $LogFilePath
        Write-Host "Total Time - " $totallooptime.Hours " hours, "
$totallooptime.Minutes " minutes and " $totallooptime.Seconds " seconds"
    }

    end{
        #Log end processing
        Write-Log -Message "Script ended" -Path $LogFilePath
    }
}
```

D Copy above three functions in same script file and start the script by calling function UploadToEmployeeDirectory with following parameters

```
UpLoadToEmployeeDirectory -siteURL "XXXXXXXXX" -EDListName "Employee Directory"
```

2 The second script makes sure that users (employees) disabled in AD are removed from the Employee Directory. The structure of script is the same as first script, with three functions: First for logging, second for getting data from AD and third for deleting users (employees) from SharePoint Online list “Employee Directory”.

A We will not go in details of the logging function as it is same as of first script

B In second function, we have change filter from `'enabled -eq $True'` to `'enabled -eq $false'`.

```
function GetUserFromAD{

        $employeeDirectory = Get-ADUser -filter 'enabled -eq $false'
        -Properties
                displayname,
                initials,
                employeeid,
                department,
                description,
                officephone,
                othertelephone,
                mail,
                office,
                extensionAttribute13,
                whenchanged,
                mobile,
                manager,
                postalCode
       -SearchBase "OU=General,OU=Standard
       Accounts,OU=Standard,DC=XXXX,DC=XXXXXX,DC=com"
       -Server XXXXXX   | where $_.employeeid -gt ""   |
        Select-Object
                DisplayName,
                enabled,
                Surname,
                GivenName,
                Initials,
                Department,
                EmployeeID,
                UserPrincipalName,
                OfficePhone,
                mobile,
                manager,
                whenchanged,
        @{expression={$_.otherTelephone};label="Extension";}

        return $employeeDirectory
}
```

C Third function delete employee information retrieved from above function from SharePoint Online list “Employee Directory”. You need to specify

I Update log file path - `$LogFile`

II Make sure that

a. `Microsoft.SharePoint.Client.dll`
b. `Microsoft.SharePoint.Client.Runtime.dll`
are at path `C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\16\ISAPI\`

III Find `$Login` and `$Password` and update with your login and password for SharePoint Online site.

```
function RemoveDisableUsersFromEmployeeDirectory{
    [CmdletBinding()]
    param(
        [Parameter(Mandatory=$true,
                   ValueFromPipeline=$true,
                   ValueFromPipelineByPropertyName=$true)]
        [String]
        $siteURL,
        [Parameter(Mandatory=$true,
                   ValueFromPipeline=$true,
                   ValueFromPipelineByPropertyName=$true)]
        [String]
        $EDListName

    )

    Begin{

        Add-Type -Path "C:\Program Files\Common Files\Microsoft Shared\Web Server
Extensions\16\ISAPI\Microsoft.SharePoint.Client.dll"
        Add-Type -Path "C:\Program Files\Common Files\Microsoft Shared\Web Server
Extensions\16\ISAPI\Microsoft.SharePoint.Client.Runtime.dll"

        #create log file
        $LogFileName = Get-Date -Format "yyyy-MM-dd HH-mm-ss"
        $LogFile = 'C:\ED\EmployeeDirectory_' + $LogFileName + '.log'
        $LogFilePath = $LogFile

        #Creating Data Table
        $dtTable = New-Object System.Data.DataTable
        $col1 = New-Object System.Data.DataColumn EmployeeID,([string])
        $col2 = New-Object System.Data.DataColumn ID,([string])

        $dtTable.Columns.Add($col1);
        $dtTable.Columns.Add($col2);
    }

    process{

        # Starting Log
        Write-Log -Message 'Script Started ... ' -Path $LogFilePath


        #Retrieving user data from AD
        Write-Log -Message "Retrieving user data from AD"  -Path $LogFilePath
        try
        {
            $tblData = GetUserFromAD
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to retrieve user data from AD"  -Path
$LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level
Error
            Exit
        }
        Write-Log -Message "Retrieved user data from AD"  -Path $LogFilePath

        #Read password and key
        try
        {
            $Login = 'XXXXXX@XXXXXXX.com'
            $Password = "XXXXXXXX"
            $Password1 = ConvertTo-SecureString $Password -AsPlainText -Force
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to open password or key files"  -Path
$LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level
Error
            Exit
        }
        Write-Log -Message "Retrieved password from file"  -Path $LogFilePath


        #Getting Context of site collection
        try
        {
            $Context = New-Object Microsoft.SharePoint.Client.ClientContext($siteURL)
            $Creds = New-Object
Microsoft.SharePoint.Client.SharePointOnlineCredentials($Login,$Password1)
            $Context.Credentials = $Creds
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to get Context of SharePoint Online"  -Path 
$LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level 
Error
            Exit
        }
        Write-Log -Message "Site Collection Context Obtained"  -Path $LogFilePath


        #Get Employee Directory
        try
        {
            $PDList = $Context.Web.Lists.GetByTitle($EDListName)
            $Context.Load($PDList)
            $Context.ExecuteQuery()
            Write-Log -Message 'SharePoint Poepple Directory List loaded' -Path 
$LogFilePath
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to list Employee Directory List form 
SharePoint Online"  -Path $LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level 
Error
            Exit
        }
        Write-Log -Message "Employee Directory context obtained"  -Path $LogFilePath


        #Download all items from Employee Directory
        try
        {
            $qry = New-Object Microsoft.SharePoint.Client.CamlQuery
            $qry.ViewXml = "<View><Query><Where></Where><OrderBy></OrderBy></Query>" +
                                "<ViewFields>" +
                                "<FieldRef Name='EmployeeID' /> " +
                                "</ViewFields>" +
                           "</View>"

            $items = $PDList.GetItems($qry)
            $Context.Load($items)
            $Context.ExecuteQuery()
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to get all list items from Employee Directory" 
-Path $LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
            Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level 
Error
            Exit
        }
        Write-Log -Message "All Items from Employee Directory retrieved"  -Path 
$LogFilePath

        #Populating DataTable from Poeple Directory List Items
        try
        {
            foreach($item in $items)
            {
                $row = $dtTable.NewRow()
                $row.EmployeeID = $item["EmployeeID"]
                $row.ID = $item["ID"]
                $dtTable.Rows.Add($row)
            }
        }
        catch [Exception]
        {
            Write-Log -Message "Not able to populate data table from Peolple Directory 
List itesm"  -Path $LogFilePath -Level Error
            Write-Log -Message "Terminating script"  -Path $LogFilePath -Level Error
            $strException = $_.Exception
```

```
                Write-Log -Message "Exception - $strException" -Path $LogFilePath -Level
Error
                Exit
            }
            Write-Log -Message "Data Table populated from Employee Directory List Items"  -
Path $LogFilePath

            #Entry to log file
            $startTime = Get-Date
            Write-Log -Message "Loop start Time $startTime" -Path $LogFilePath

            foreach ($row in $tblData)
                    {
                        $totaltime = NEW-TIMESPAN -Start $row.whenchanged -End $startTime
                        $strEmployeeID = "EmployeeID='" + $row.EmployeeID + "'"
                        $dtrow = $dtTable.Select("EmployeeID='" + $row.EmployeeID + "'")

                        if ($dtrow.Count -eq 1)
                        {
                            Write-Log -Message "Deleting $row.DisplayName" -Path
$LogFilePath
                            try
                            {
                                $listItem = $PDList.GetItemById($dtrow.ID)
                                $listItem.Recycle()
                                $Context.ExecuteQuery()
                                Write-Log -Message "Deleted Emplpyee - $row.DisplayName
from Employee Directory" -Path $LogFilePath

                            }
                            Catch [Exception]
                            {
                            #log exception
                            Write-Log -Message "Not able to delete user $row.DisplayName
from Employee Directory"  -Path $LogFilePath -Level Error
                            $strException = $_.Exception
                            Write-Log -Message "Exception - $strException" -Path
$LogFilePath -Level Error
                            }
                        }
                        else
                        {
                            Write-Log -Message "Not able to delete user $row.DisplayName
from Employee Directory. User doesn't exist in Employee Directory"  -Path $LogFilePath
                        }
                    }

            $endTime = Get-Date
            $totallooptime = NEW-TIMESPAN -Start $startTime -End $endTime
            Write-Log -Message "Loop End Time $endTime " -Path $LogFilePath
            $strTotalTime = "Total Loop Time - " + $totallooptime.Hours + " hours, " +
$totallooptime.Minutes + " minutes and "  + $totallooptime.Seconds + " seconds"
            Write-Log -Message $strTotalTime -Path $LogFilePath
        }

        end{
            #Log end processing
            Write-Log -Message "Script ended" -Path $LogFilePath
        }
}
```

D Copy above three functions in same script file and start the script by calling the function *RemoveDisableUsersFromEmployeeDirectory* with following parameters:

```
RemoveDisableUsersFromEmployeeDirectory -siteURL "XXXXXXXXXX" -EDListName "Employee
Directory"
```

Once employee information is uploaded or synchronised with SharePoint list “Employee Directory”, you can create and update views and arrange employee information as you need.

Like it? Let us know.

Follow us to see more tips and tricks in our next blog series….