# Requirements

## Download and install MS Online Utilities

1. Microsoft Online Services Sign-In Assistant
2. Windows Azure Active Directory Module for Windows PowerShell

<http://aka.ms/aadposh> MS site for both pieces of software

## Allow Remotely Signed PowerShell Scripts

Open Windows Azure Active Directory Module for Windows PowerShell as Administrator

Type: set-executionpolicy remotesigned

It should say "Execution Policy Change..."

Hit enter, enter

How to Connect (this is in the templates mentioned below as well)

Open Microsoft Online Services Module for Windows PowerShell

To Connect, copy and paste this into the window:

$LiveCred = Get-Credential

$Session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri https://ps.outlook.com/powershell/ -Credential $LiveCred -Authentication Basic -AllowRedirection

Import-PSSession $Session

Connect-MsolService -Credential $LiveCred

To Disconnect:

Remove-PSSession $Session

# Preconfigured connection templates (optional)

Copy and modify an existing .ps1 Template in \\mcgladrey.rsm.net\bos01data\Dept\Consulting\Consulting-Technology\Shared-Client Info\Helpdesk Client Info\Client\_Information\Office 365

Edit the file and change $Username and $Password to the 365 Portal login

Add the necessary code right below ## Insert Commands Here! ## and Save

Open Microsoft Online Services Module for Windows PowerShell

Either change the “Start in” location to where the script is or use CD to change to that path

Run the .PS1 file by typing (without quotes) “.\script.ps1” and hit enter

# Common Commands

## User/Mailbox Commands

## Create New User

New-MsolUser -UserPrincipalName user@domain.com -DisplayName "Full Name" -FirstName "First" -LastName "Last" -LicenseAssignment LICENSEPACK -UsageLocation US -Password "password" -ForceChangePassword 0

## Remove User

Remove-MsolUser –userprincipalname ‘evilconsultant@contoso.com’

## List All Aliases

**get-mailbox | select -expand emailaddresses alias**

## List All Aliases for Specific User

**get-mailbox MAILBOX| select -expand emailaddresses alias**

## Change a user’s primary email address without changing existing addresses or login

Set-Mailbox MAILBOX –WindowsEmailAddress EMAILADDRESS

## Add an email alias to an existing account (cannot be done on DirSynced users, use ADSI Edit instead)

Set-Mailbox MAILBOX -EmailAddresses @{add="EMAILADDRESS"}

## Delete a mailbox and remove user from all distribution groups

$user = "user@domain.com"

$AllDGs = Get-DistributionGroup

foreach($group in $AllDGs)

{

$member = Remove-DistributionGroupMember -Identity "$group" | where {$\_.PrimarySmtpAddress -eq $user}

}

Remove-Mailbox -identity "$User"

## Find Deleted Users

Get-MsolUser –ReturnDeletedUsers

## Permanently Delete any Deleted Mailboxes that remain in the GAL

Get-MsolUser -ReturnDeletedUsers | foreach { Remove-MsolUser -ObjectId $\_.ObjectId -RemoveFromRecycleBin -Force }

## Forward email to another address

$user = “user@domain.com”

$forwardingaddress = “forward@domain.com”

Set-Mailbox -Identity $user -DeliverToMailboxAndForward $true –ForwardingSMTPAddress $forwardingaddress

## Remove email forwarding

$user = “user@domain.com”

Set-Mailbox -Identity $user -DeliverToMailboxAndForward $false -ForwardingSMTPAddress $null

## Convert User Mailbox to Shared Mailbox

Set-Mailbox -Identity MAILBOX -Type "Shared" -ProhibitSendReceiveQuota 5GB -ProhibitSendQuota 4.75GB -IssueWarningQuota 4.5GB

## Convert User Mailbox to Room Mailbox

Set-Mailbox “MAILBOX” -Type Room

Set-CalendarProcessing -Identity "MAILBOX" -AutomateProcessing AutoAccept -DeleteComments $true -AddOrganizerToSubject $true -AllowConflicts $false

## Password Commands

## Change Password

Set-MsolUserPassword -UserPrincipalName user@yourdomain -NewPassword “PASSWORD” -ForceChangePassword 0

## Change Password for all users

Get-MsolUser | Set-MsolUserPassword -NewPassword “PASSWORD” -ForceChangePassword 0

## Set Password Policy for the entire domain

Set-MsolPasswordPolicy -DomainName contoso.com -NotificationDays 15 -ValidityPeriod 180

## Set Password to Never Expire

Set-MsolUser -UserPrincipalName USER -PasswordNeverExpires 1

## Set Password to Never Expire for all users

Get-MsolUser | Set-MsolUser -PasswordNeverExpires 1

## Permission Commands

## Get Calendar Permissions

Get-MailboxFolderPermission -Identity “TARGETUSER:\Calendar”

## Add Calendar Permissions

Add-MailboxFolderPermission -Identity “TARGETUSER:\Calendar” -User USER -AccessRights PublishingEditor

## Add default Calendar Permissions for all users

$users = Get-Mailbox

foreach($user in Get-Mailbox) {

$cal = $user.alias+":\Calendar"

Set-MailboxFolderPermission -Identity $cal -User Default -AccessRights Reviewer

}

## Remove Calendar Permissions

Remove-MailboxFolderPermission -Identity “TARGETUSER:\Calendar” –User USER -Confirm:$False

## Modify Calendar Permissions

Set-MailboxFolderPermission -Identity “TARGETUSER:\Calendar” -User USER -AccessRights PublishingEditor

## Add Full Access Mailbox Permissions without Automapping (preferred method)

Add-MailboxPermission -Identity TARGETMAILBOX -User USER -AccessRights FullAccess –AutoMapping:$false

## Add Full Access Mailbox Permissions to All Users without Automapping (preferred method)

$Mailboxes = Get-Mailbox

foreach ($Mailbox in $Mailboxes){Add-MailboxPermission -Identity Calendar -User $Mailbox.samaccountname -AccessRights FullAccess -AutoMapping:$false}

## Add Full Access Mailbox Permissions with Automapping

Add-MailboxPermission -Identity TARGETMAILBOX -User USER -AccessRights FullAccess

## Add Full Access Mailbox Permissions for All Users to Administrator

Get-Mailbox | Add-MailboxPermission -user $ADMINUSERNAME -AccessRights FullAccess

## Remove Mailbox Permissions

Remove-MailboxPermission -Identity TARGETMAILBOX -User USER

## Add Send As Permissions

Add-RecipientPermission "grouptoaccess@domain.com" -AccessRights SendAs -Trustee "usertoaccess@domain.com"

## Remove Send As Permissions

Add-RecipientPermission "grouptoaccess@domain.com" -AccessRights SendAs -Trustee "usertoaccess@domain.com"

## Query Send On Behalf Permissions

Get-Mailbox -Identity TARGETMAILBOX | FL GrantSendOnBehalfTo

## Add Send On Behalf Permissions

Set-Mailbox -Identity TARGETMAILBOX -GrantSendOnBehalfTo USER

## Remove Send on Behalf Permissions

Set-Mailbox –Identity TARGETMAILBOX -GrantSendOnBehalfTo @{remove=USER}

## Licensing Commands

## Get list of unlicensed users and export to CSV

Get-MsolUser -UnlicensedUsersOnly | Export-Csv c:\unlicensedUsers.csv

## Get list of licensed users and export to CSV

Get-MsolUser | Where-Object {$\_.isLicensed -eq "TRUE"} | Export-Csv c:\licensedUsers.csv

## Get Total Licenses (available and used)

Get-MsolAccountSku

## Get the License Pack Name and export to CSV

<#

Humboldt: HUMBOLDTMOVINGANDSTORAGE:STANDARDPACK

Watermark: WATERMARKDONUT:STANDARDPACK

ECC: eastcoastcomm:ENTERPRISEPACK

Briar: BriarGroup:ENTERPRISEPACK

BriarGroup:STANDARDWOFFPACK

BriarGroup:DESKLESSPACK

Kettle: kettlecuisine:STANDARDPACK

kettlecuisine:ENTERPRISEPACK

#>

Get-MsolAccountSku #| Export-Csv c:\license.csv

## Distribution Group Commands

## Create a Distribution Group

New-DistributionGroup -Name "Group Name" -DisplayName "Group Name" -PrimarySmtpAddress "group@domain.com"

## Delete a Distribution Group

Remove-DistributionGroup -Identity "GroupAlias" -Confirm:$False

## Set Owner of all distribution groups to specific user

Get-DistributionGroup |Set-DistributionGroup -ManagedBy “USERNAME” –BypassSecurityGroupManagerCheck

## Add an e-mail alias to an existing distribution group (must put the primary as the first address in the list or it will be removed)

Set-DistributionGroup "DISTRIBUTIONGROUPNAME" -EmailAddresses SMTP:EMAIL1,EMAIL2

## Add user to Distribution Group

Add-DistributionGroupMember -Identity “DISTRIBUTIONGROUP” -Member USERTOADD –BypassSecurityGroupManagerCheck

## Add all users to Distribution Group

$users = Get-Mailbox

foreach($user in Get-Mailbox) {

Add-DistributionGroupMember -Identity GROUP -Member $user.alias -BypassSecurityGroupManagerCheck

}

## Remove user from Distribution Group

Remove-DistributionGroupMember -Identity DISTRIBUTIONGROUP -Member USERTOREMOVE –BypassSecurityGroupManagerCheck -Confirm:$False

## Get the list of members to a Distribution Group

Get-DistributionGroupMember -Identity "DistributionGroup" | Select DisplayName

## Find out which Distribution Groups a user belongs to

$usersmtp = "user@domain.com"

write-output ‘ ‘

Write-output "$usersmtp is a member of these groups:"

write-output ‘ ‘

$AllDGs = Get-DistributionGroup

foreach($group in $AllDGs)

{

$member = Get-DistributionGroupMember -Identity "$group" | where {$\_.PrimarySmtpAddress -eq $usersmtp}

If ($member)

{

write-output "$group"

}

}

## Room Mailbox Commands

## Create a Room Mailbox

#Change the 4 variables below and it will automatically allow rooms to be booked 1 year in advanced between 8am and 5pm

$name = "Room Name"

$displayN = "Room Name"

$identity = "email@domain.com"

$email = "email@domain.com"

New-Mailbox -Name $name -DisplayName $displayN -FirstName $name –Room

Set-Mailbox -identity $identity -EmailAddresses $email

#If you get Mailbox Access issues, you need to wait for the 365 AD servers to replicate

Set-CalendarProcessing -identity $identity -AutomateProcessing AutoAccept -MaximumDurationInMinutes 0

Set-MailboxRegionalConfiguration -identity $identity -TimeZone “Eastern Standard Time”

Set-MailboxCalendarConfiguration -identity $identity -WorkDays Weekdays -WorkingHoursStartTime 08:00:00 -WorkingHoursEndTime 19:00:00 -WorkingHoursTimeZone “Eastern Standard Time”

Set-CalendarProcessing $identity -BookingWindowInDays $BookingWindow

Set-MailboxFolderPermission -identity $name”:\Calendar” -User Default -AccessRights Author

## Limit Room Booking Rights

To view how its set up:

* Get-CalendarProcessing –identity “MAILBOX” | fl \*policy

If AllBookInPolicy is True then anyone can book.

* Set-CalendarProcessing –identity “MAILBOX” –AllBookInPolicy:$false

To allow specific people to book this room use below command:

* Set-CalendarProcessing –Identity “MAILBOX” –BookInPolicy “MAILBOX2”

## Contacts Commands

## Create External Contacts

New-MailContact -Name “Full Name” -DisplayName “Display Name” -ExternalEmailAddress “Email” -FirstName “First Name” -LastName “Last Name”

## Delete External Contacts

Remove-MailContact -Identity “Email Address” -Confirm:$False

## Run a command in bulk using a CSV file

<#

You’ll need to create a corresponding .csv file that contains a header value for each variable defined

For example, to [create external contacts](#_#Create_External_Contacts) from CSV, take the basic command below and convert it into the following:

Import-Csv .\ExternalContacts.csv|%{New-MailContact -Name $\_.Name -DisplayName $\_.Name -ExternalEmailAddress $\_.ExternalEmailAddress -FirstName $\_.FirstName -LastName $\_.LastName}

Command becomes New-MailContact

Parameter becomes –Name, -DisplayName, etc.

$\_.Variable is defined as a header in your CSV file

#>

Import-Csv .\File.csv|%{Command -Parameter $\_.Variable}