Shared calendar on exchange

```
New-SharedCalendar.ps1 -Name "Shared Calendar" -SamAccountName
sharedcalendar -UserprincipalName sharedcalendar@sgdev.stevieg.org -Database
DB01 -OrganizationalUnit sgdev.stevieg.org/People -Owners sgoodman,fstone -
Editors epitts,csutton -Reviewers fcarson -WebPublish:$True
```

```
<#
    .SYNOPSIS
    Creates a mailbox for use as a shared calendar

    Steve Goodman
    .DESCRIPTION
    Creates a shared mailbox, assigns owner(s), with full access, along with
editors and viewers. Restricts mail to authenticated users.
    Additionally generates a HTML/iCal published URL if specified.

    .PARAMETER Name
    Name of the Shared Calendar. Used for the Display Name

    .PARAMETER SamAccountName
    Optional - Pre-Windows 2000 Login name to use when creating the mailbox.
If not specified, uses defaults

    .PARAMETER UserPrincipalName
    Optional - User Principal Name (UPN) to use when creating the mailbox.
If not specified, uses defaults

    .PARAMETER Database
    Optional - Mailbox Database to use

    .PARAMETER OrganizationalUnit
    Optional - AD Organizational Unit to use when creatng the mailbox. If
not specified, uses defaults.

    .PARAMETER Owners
    List of the people who should be the owners of the mailbox. The first
will be specified as the "manager" field, their department used, and all
owners will be given "Full Access" mailbox permissions. The Auto-Mapping
will be removed.

    .PARAMETER Editors
    Optional - List of the people who should have editor access on the
Calendar

    .PARAMETER Reviewers
    Optional - List of the people who should have viewer access on the
Calendar
```

```
    .PARAMETER WebPublish
    Optional - Publish the Calendar via private iCal and HTML URLs. Defaults
to False. Requires Default Sharing Policy to allow Web Publishing.

    .EXAMPLE
    Creates a new calendar, allowing defaults:
    New-SharedCalendar.ps1 -Name "Test Calendar" -Owners steve,lisa -Editors
isabelle,peter -Reviewers liz,drew

    .EXAMPLE
    Creates a new calendar, many options set
    New-SharedCalendar.ps1 -Name "Shared Calendar" -SamAccountName
sharedcalendar -UserprincipalName sharedcalendar@sgdev.stevieg.org -Database
DB01 -OrganizationalUnit sgdev.stevieg.org/People -Owners sgoodman,fstone -
Editors epitts,csutton -Reviewers fcarson -WebPublish:$True

    #>
param(
[parameter(Position=0,Mandatory=$true,ValueFromPipeline=$false,HelpMessage="
Calendar Name")][string]$Name,
[parameter(Position=1,Mandatory=$false,ValueFromPipeline=$false,HelpMessage=
"Logon Name (SAM Account Name)")][string]$SamAccountName,
[parameter(Position=2,Mandatory=$false,ValueFromPipeline=$false,HelpMessage=
"Logon Name (User Principal Name)")][string]$UserPrincipalName,
[parameter(Position=3,Mandatory=$false,ValueFromPipeline=$false,HelpMessage=
"Mailbox Database")]$Database,
[parameter(Position=4,Mandatory=$false,ValueFromPipeline=$false,HelpMessage=
"Organizational Unit")][string]$OrganizationalUnit,
[parameter(Position=5,Mandatory=$true,ValueFromPipeline=$false,HelpMessage="
Mailbox Owners")][array]$Owners,
[parameter(Position=6,Mandatory=$false,ValueFromPipeline=$false,HelpMessage=
"Calendar Editors")][array]$Editors,
[parameter(Position=7,Mandatory=$false,ValueFromPipeline=$false,HelpMessage=
"Calendar Reviewers")][array]$Reviewers,
[parameter(Position=8,Mandatory=$false,ValueFromPipeline=$false,HelpMessage=
"Publish (privately) via the web?")][bool]$WebPublish=$false
    )

# Check all pre-reqs
if ((Get-PSSnapin Microsoft.Exchange.Management.PowerShell.Admin -
ErrorAction SilentlyContinue))
{
    throw "Exchange 2007 Management Console Not Supported"
}
if (!(Get-Command New-Mailbox -ErrorAction SilentlyContinue))
{
    throw "Please launch the Exchange 2010 Management Shell"
}

# Setup splatting hashtable
```

```
$NewSharedMailbox = @{
    "Shared" = $True
    "Name" = $Name
}

# Check parameters OK and add relevant parameters to splatting hashtable
$Recipient=Get-Recipient $Name -ErrorAction SilentlyContinue
if ($Recipient)
{
    $Recipient
    throw "Recipient $($Name) Exists (See above)"
}
if ($SamAccountName)
{
    $Recipient=Get-Mailbox $SamAccountName -ErrorAction SilentlyContinue
    if ($Recipient)
    {
        $Recipient
        throw "Recipient $($SamAccountName) Exists (See above)"
    }
    $NewSharedMailbox.Add("SamAccountName",$SamAccountName)
}
if ($UserPrincipalName)
{
    $Recipient=Get-Mailbox $UserPrincipalName -ErrorAction SilentlyContinue
    if ($Recipient)
    {
        $Recipient
        throw "Recipient $($UserPrincipalName) Exists (See above)"
    }
    $NewSharedMailbox.Add("UserPrincipalName",$UserPrincipalName)
}
if ($Database)
{
    $MailboxDatabase=Get-MailboxDatabase $Database -ErrorAction
SilentlyContinue
    if (!$MailboxDatabase)
    {
        throw "Mailbox Database $($Database) not found"
    }
    $NewSharedMailbox.Add("Database",$Database)
}
if ($OrganizationalUnit)
{
    $objOrganizationalUnit = Get-OrganizationalUnit $OrganizationalUnit -
ErrorAction SilentlyContinue
    if (!$objOrganizationalUnit)
    {
        throw "Organizational Unit $($OrganizationalUnit) not found"
    }
    $NewSharedMailbox.Add("OrganizationalUnit",$OrganizationalUnit)
```

```
}
if ($Owners.Count -eq 0)
{
    throw "You must specify at least one owner"
}
foreach ($Owner in $Owners)
{
    if (!(Get-Mailbox $Owner -ErrorAction SilentlyContinue))
    {
        throw "Owner mailbox $($Owner) not found"
    }
}
if ($Editors)
{
    foreach ($Editor in $Editors)
    {
        if (!(Get-Mailbox $Editor -ErrorAction SilentlyContinue))
        {
            throw "Editor mailbox $($Editor) not found"
        }
    }
}
if ($Reviewers)
{
    foreach ($Reviewer in $Reviewers)
    {
        if (!(Get-Mailbox $Reviewer -ErrorAction SilentlyContinue))
        {
            throw "Reviewer mailbox $($Reviewer) not found"
        }
    }
}

# Create mailbox
Write-Host -ForegroundColor Green "Creating Shared Calendar Mailbox"
$Mailbox = New-Mailbox @NewSharedMailbox
if (!$Mailbox)
{
    throw "An error occurred creating the shared calendar mailbox"
}
$DomainController = $Mailbox.OriginatingServer
$Mailbox

# Set Owner Details including Department to match the first specified Owner
Write-Host -ForegroundColor Green "Setting Shared Calendar Mailbox Owner,
Department and Description"
$Mailbox | Set-User -Manager (Get-User $Owners[0]) -Department ((Get-User
$Owners[0]).Department) -DomainController $DomainController

# Set Description
```

```
$LDAPUser =
[ADSI]"LDAP://$($DomainController)/$($Mailbox.DistinguishedName)"
$LDAPUser.description = "Shared Calendar"
$LDAPUser.SetInfo()

$LDAPUser =
[ADSI]"LDAP://$($DomainController)/$($Mailbox.DistinguishedName)"
Write-Host "Manager: $($LDAPUser.manager)"
Write-Host "Department: $($LDAPUser.department)"
Write-Host "Description: $($LDAPUser.description)"

# Set authenticated mail only
$Mailbox | Set-Mailbox -RequireSenderAuthenticationEnabled:$true


# Set Owner Permissions
Write-Host -ForegroundColor Green "Adding Shared Calendar Mailbox Owner
Permissions and removing Outlook auto-mapping"
foreach ($Owner in $Owners)
{
    # Add Permission
    $Mailbox | Add-MailboxPermission -User $Owner -AccessRights FullAccess -
DomainController $DomainController
    # Remove Auto-Mailbox mapping
$LDAPUser=[ADSI]"LDAP://$($DomainController)/$($Mailbox.DistinguishedName)"
    $LDAPUser.msExchDelegateListLink.Remove(((Get-Mailbox
$Owner).DistinguishedName))
    $LDAPUser.SetInfo()
}

if ($Editors -or $Reviewers)
{
    # Wait until Mailbox is ready before adding folder permissions
    Write-Host -ForegroundColor Green "Waiting until mailbox folder
structure is available before adding folder permissions"
    $MailboxReady=$False
    while ($MailboxReady -eq $False)
    {
        Write-Host -NoNewline "."
        $Result = Get-MailboxFolderStatistics $Mailbox -ErrorAction
SilentlyContinue
        if ($Result)
        {
            $MailboxReady = $True
        }
        sleep 5
    }
    Write-Host
}
# Set Editor Permissions
if ($Editors)
```

```
{
    Write-Host -ForegroundColor Green "Adding Editor permissions to
Calendar"
    foreach ($Editor in $Editors)
    {
        Add-MailboxFolderPermission "$($Mailbox.SamAccountName):\Calendar" -
User $Editor -AccessRights Editor -DomainController $DomainController
    }
}

# Set Reiewer Permissions
if ($Reviewers)
{
    Write-Host -ForegroundColor Green "Adding Reviewer permissions to
Calendar"
    foreach ($Reviewer in $Reviewers)
    {
        Add-MailboxFolderPermission "$($Mailbox.SamAccountName):\Calendar" -
User $Reviewer -AccessRights Reviewer -DomainController $DomainController
    }
}

# Publish Calendar
if ($WebPublish)
{
    if (((Get-SharingPolicy | Where {$_.Default -eq $True}).Domains|Where
{$_.Domain -eq "anonymous"})) {
        Write-Host -ForegroundColor Green "Publishing Calendar using private
URL"
        Set-MailboxCalendarFolder -Identity
"$($Mailbox.SamAccountName):\Calendar" -DetailLevel FullDetails -
PublishDateRangeFrom OneYear -PublishDateRangeTo OneYear -
PublishEnabled:$true -SearchableUrlEnabled:$false -DomainController
$DomainController
        $MailboxCalendarFolder = Get-MailboxCalendarFolder -Identity
"$($Mailbox.SamAccountName):\Calendar" -DomainController $DomainController
        Write-Host -ForegroundColor Yellow -NoNewline "Web URL: "
        Write-Host $MailboxCalendarFolder.PublishedCalendarUrl
        Write-Host -ForegroundColor Yellow -NoNewline "iCal URL: "
        Write-Host $MailboxCalendarFolder.PublishedICalUrl
    } else {
        Write-Host -ForegroundColor Yellow "Skipping Web/iCal Publishing
because Default Sharing Policy does not allow web publishing (to anonymous
domains)"
    }
}
```