Contents

[eDiscovery Microsoft Teams Private Channels SharePoint Site 1](#_Toc26353696)

[Example 1. Use SharePoint Online PowerShell to list all private channel sites 1](#_Toc26353697)

[Example 2. Identify the Private Channels for a specific user account 1](#_Toc26353698)

[Example 3. Use the Microsoft GRAPH API 2](#_Toc26353699)

# eDiscovery Microsoft Teams Private Channels SharePoint Site

Records for messages sent in a private channel are delivered to the mailbox of all private channel members, rather than to a group mailbox. The titles of the records are formatted to indicate which private channel they were sent from.

Searching a custodian’s mailbox will include the private channel messages.

On top of the conversation messages, each channel has its own SharePoint site collection that's separate from the parent team site, files in a private channel are managed independently of the parent team.

This poses a challenge for enterprise organizations that needs to place information in these SharePoint sites on hold. Below are a few samples that can be used to enumerate these sites and apply actions.

They can be easily customized to fit the model that your organization is using for eDiscovery and hold.

## Example 1. Use SharePoint Online PowerShell to list all private channel sites

#Install-Module -Name Microsoft.Online.SharePoint.PowerShell

Connect-SPOService -Url https://”yourtenantname”-admin.sharepoint.com

$sites = get-sposite -template "teamchannel#0"

$sites

The output of $sites.url would look like the below, to get the count view $sites.count. This can then be used in eDiscovery to perform holds and searches specific to the channels.

$sites.url

https://tenant.sharepoint.com/sites/TheGossipTeam-SuperSecretchannel

https://tenant.sharepoint.com/sites/Thesecretteam-Thisisthesecretsecretchannel

https://tenant.sharepoint.com/sites/Thesecretteam-Thisisthesecretsecretchannel2

## Example 2. Identify the Private Channels for a specific user account

The latest SharePoint modules must be installed to return the relatedgroupid. The relatedgroupid is a GUID pointing to the Team that the private channel belongs to. If you don’t have the recent modules the value will not be returned. Change the username to match the account used for the match. If you need to get the information for a group of users create a loop that changes the $username variable.

#Install-Module -Name MicrosoftTeams

#Connect-MicrosoftTeams

#Install-Module -Name Microsoft.Online.SharePoint.PowerShell

#Connect-SPOService -Url https://"tenantname"-admin.sharepoint.com

$privatechannels = @()

$username = "User@domain.com"

$teams = Get-Team -User $username

$sites = get-sposite -template "teamchannel#0"

foreach ($team in $teams) {

foreach ($site in $sites) {

$matchsite = Get-SpoSite -Identity $site.url -Detail; if ($matchsite.RelatedGroupId -eq $team.groupid)

{if ((Get-TeamUser -GroupId $team.GroupId).user -contains $username) {$privatechannels += $site}}

                            }

                            }

$privatechannels

The output will be the same as sample 1. With the difference that only the private channels associated to the user is listed in $privatechannels.

## Example 3. Use the Microsoft GRAPH API

The sample below uses the Microsoft GRAPH to list all the private channels for a Team. It is based on the $groupId variable. This can easily be expanded to all groups by first enumerating all the groups using /groups?$select=id,resourceProvisioningOptions as documented here <https://docs.microsoft.com/en-us/graph/teams-list-all-teams> . Then return private channels for all groupid’s. For the full GRAPH documentation please see <https://docs.microsoft.com/en-us/graph/api/resources/teams-api-overview?view=graph-rest-1.0> This option may be useful if you do other native GRAPH scripting.

$clientID = "Your ClientID”

$clientSecret = "Your ClientSecret"

$loginURL = "https://login.windows.net"

$tenantdomain = "Your tenant"

$resource = "https://graph.microsoft.com"

$groupId = "The GUID of the group of interest"

$SPO = @()

# Get an Oauth 2 access token based on client id, secret and tenant domain

$body = @{grant\_type="client\_credentials";resource=$resource;client\_id=$ClientID;client\_secret=$ClientSecret}

$oauth = Invoke-RestMethod -Method Post -Uri $loginURL/$tenantdomain/oauth2/token?api-version=1.0 -Body $body

#Let's put the oauth token in the header, where it belongs

$headerParams  = @{'Authorization'="$($oauth.token\_type) $($oauth.access\_token)"}

$Sites = Invoke-RestMethod -Headers $headerParams -Uri "https://graph.microsoft.com/beta/teams/$groupId/channels" -Method Get      #?filter=membershipType eq 'private'  if you only want to return private channels

foreach ($Site in $Sites.value) {

$Channel = $Site.id

$SPO += Invoke-RestMethod -Headers $headerParams -Uri "https://graph.microsoft.com/beta/teams/$groupId/channels/$Channel/filesFolder" -Method Get

                                }

#$Sites.value

$SPO.weburl