WHITE PAPER

How to Manage Microsoft Teams: An Admin Guide

Author: Greg Jones

Quadrotech

Contents

Native Management of Microsoft Teams 3
The Configuration 3
PowerShell 3
Install the Teams PowerShell Module 3
Connect PowerShell to the Tenant 3
Show Available Cmdlets 4
New Team PowerShell Options 5
Create the Team 6
Add Members and Channels 6
Modify Settings 7
Remove the Team 7
Scripting the Team Creation 7
Using the Graph API 8
Create the Group and Add Members 8
Create the Team with Settings 9
Add the Channel 10
A Single Call? 11
Conclusion 11
Teams Templates 12
What are Teams Templates? 12
Before Teams Templates: Using Existing Teams as a Template 12
Base Templates 16
Properties Supported by Teams Templates 16
PowerShell and Templates 16
Graph API and Templates 17
Application Permissions 17
User Permissions with Advanced Requests Using the Standard Template 15
Takeaways from API Testing 21
Delegating Teams Creation 21
Azure AD Roles for Delegation 21
The Teams Service Administrator Role 23
How Does This Work? 23
Granular Delegated Administration 25
The Use Case 25
The Solution 25
The Result 28
Conclusion 29

Native Management of Microsoft Teams

There are now three ways to manage Microsoft Teams natively. The first is through the **Microsoft 365 Admin Center**. This will allow you to manage the users and groups associated with the Teams and you will also have a link in the Admin Center to get to the Teams Admin Center. This won't be covered here except to mention that if you are a Global Admin you have full rights. If not, you will need to be added to the "Teams Service Administrator" role in AAD. The other 2 options that will be covered here will be management via PowerShell and via the Graph API.

The configuration

For both methods, we will be creating the "UK Finance Team" with a "Regulatory Compliance" channel with members allowed to create and update channels, but not allowed to delete them. Users will also not be allowed to add or remove apps.

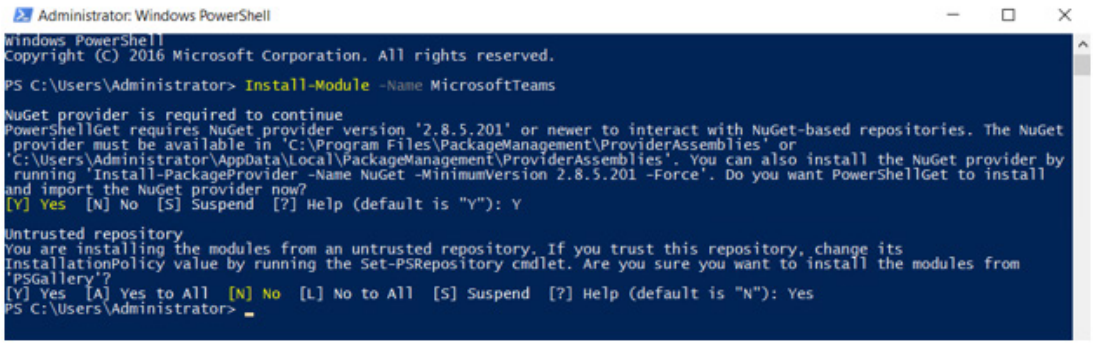
PowerShell

Let's deal with PowerShell first since it is a more common approach for administration.

Install the Teams PowerShell Module

The first step in managing Teams via PowerShell is to install the Teams Cmdlets module. Running PowerShell as an Administrator, you will need to run the following and allow install from the untrusted repo as well as installing the NuGet package:

```
Install-Module -Name MicrosoftTeams
```

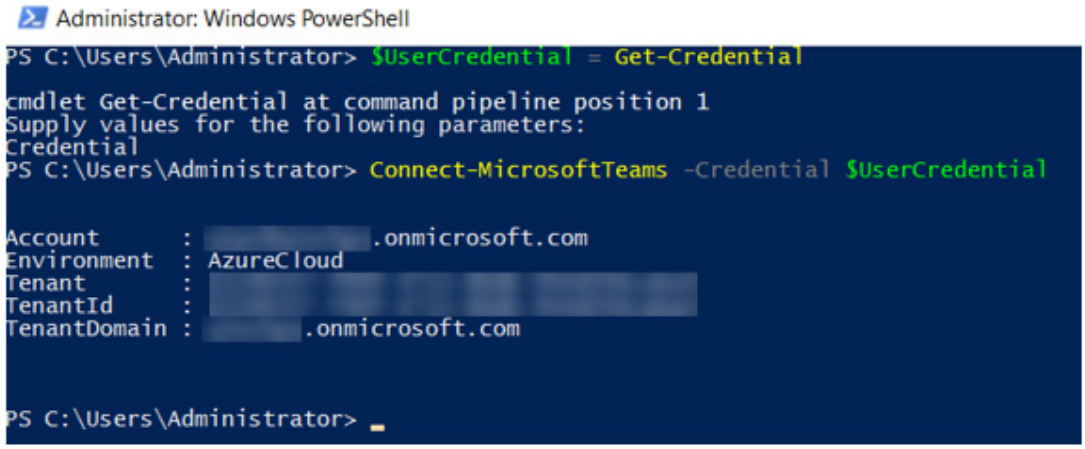


Connect PowerShell to the Tenant

The next step is to connect to Teams in the tenant. This can be done with the following script.

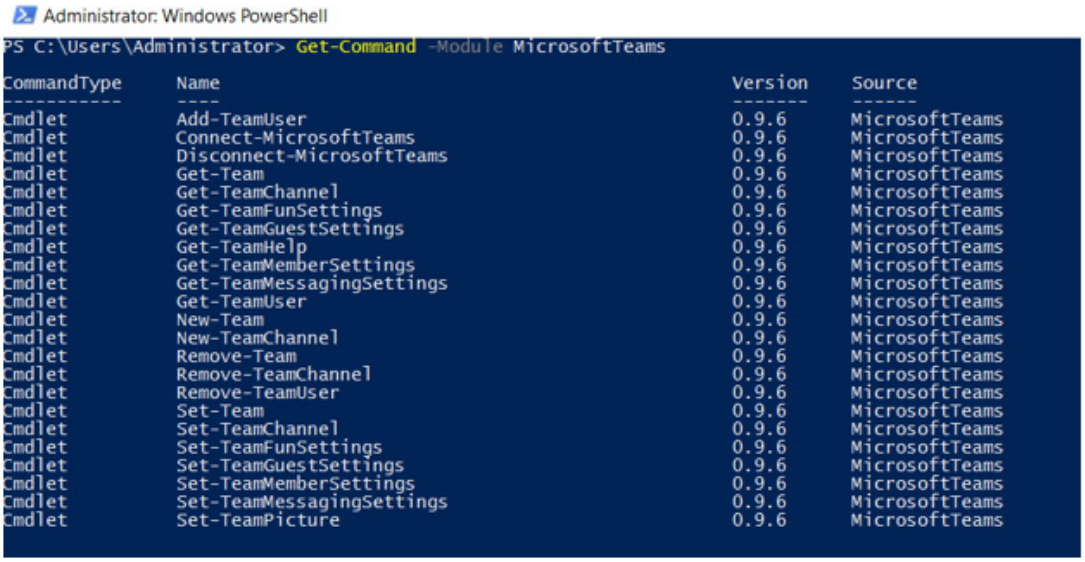
```
$UserCredential = Get-Credential
Connect-MicrosoftTeams -Credential $UserCredential
```

Voila...you are connected to Microsoft Teams as long as you have the appropriate rights.



Show Available Cmdlets

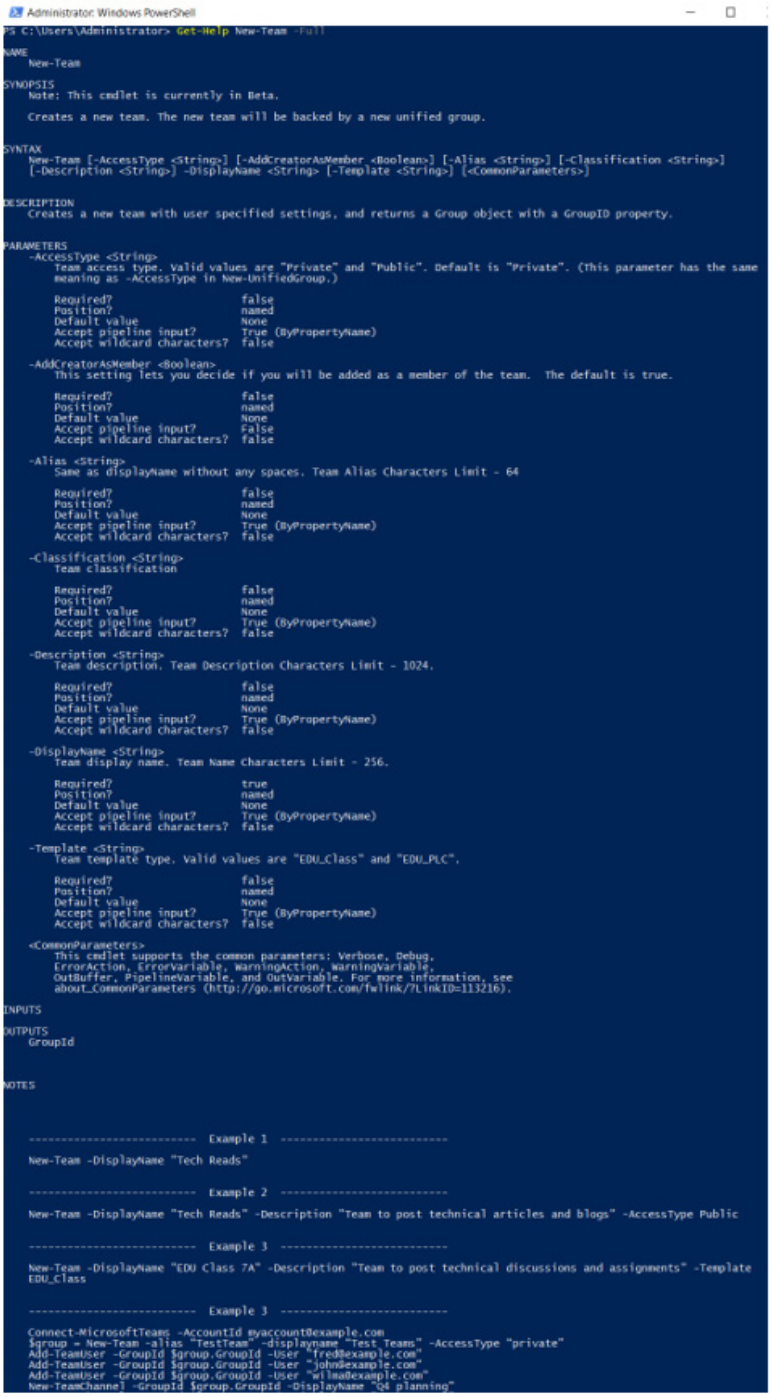
You can now run Get-Command -Module MicrosoftTeams to list the available cmdlets.



CommandType	Name	Version	Source
Cmdlet	Add-TeamUser	0.9.6	MicrosoftTeams
Cmdlet	Connect-MicrosoftTeams	0.9.6	MicrosoftTeams
Cmdlet	Disconnect-MicrosoftTeams	0.9.6	MicrosoftTeams
Cmdlet	Get-Team	0.9.6	MicrosoftTeams
Cmdlet	Get-TeamChannel	0.9.6	MicrosoftTeams
Cmdlet	Get-TeamFunSettings	0.9.6	MicrosoftTeams
Cmdlet	Get-TeamGuestSettings	0.9.6	MicrosoftTeams
Cmdlet	Get-TeamHelp	0.9.6	MicrosoftTeams
Cmdlet	Get-TeamMemberSettings	0.9.6	MicrosoftTeams
Cmdlet	Get-TeamMessagingSettings	0.9.6	MicrosoftTeams
Cmdlet	Get-TeamUser	0.9.6	MicrosoftTeams
Cmdlet	New-Team	0.9.6	MicrosoftTeams
Cmdlet	New-TeamChannel	0.9.6	MicrosoftTeams
Cmdlet	Remove-Team	0.9.6	MicrosoftTeams
Cmdlet	Remove-TeamChannel	0.9.6	MicrosoftTeams
Cmdlet	Remove-TeamUser	0.9.6	MicrosoftTeams
Cmdlet	Set-Team	0.9.6	MicrosoftTeams
Cmdlet	Set-TeamChannel	0.9.6	MicrosoftTeams
Cmdlet	Set-TeamFunSettings	0.9.6	MicrosoftTeams
Cmdlet	Set-TeamGuestSettings	0.9.6	MicrosoftTeams
Cmdlet	Set-TeamMemberSettings	0.9.6	MicrosoftTeams
Cmdlet	Set-TeamMessagingSettings	0.9.6	MicrosoftTeams
Cmdlet	Set-TeamPicture	0.9.6	MicrosoftTeams

New Team PowerShell Options

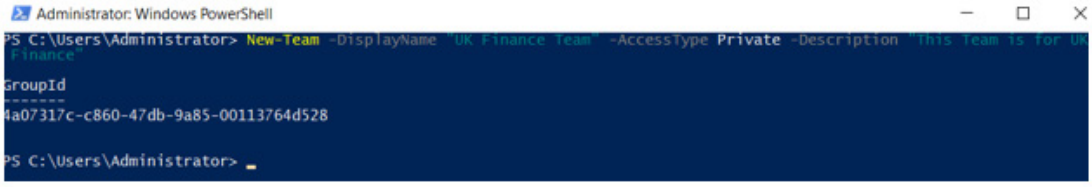
For this example, we will create a new Team. If we look at the results of a Get-Help New-Team -Full, you can see the syntax of the command as well as the parameters.



Quadrotech

Create the Team

We can create the new team.
New-Team -DisplayName “UK Finance Team” -AccessType Private -Description “This Team is for UK Finance”



This will provide us with the GroupID for the Team so we know the cmdlet was successful. The Team is created, but is does not have membership, channels, or settings.

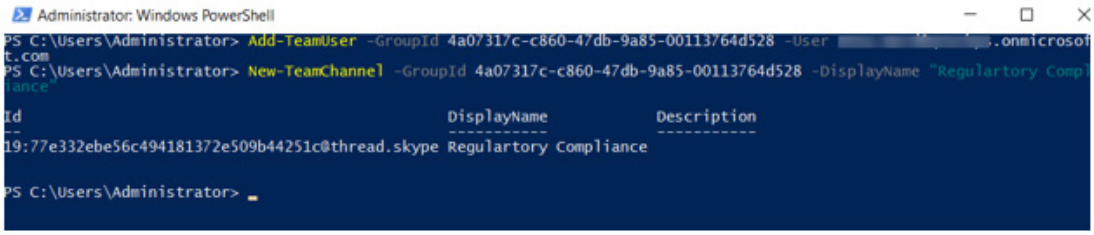
Add Members and Channels

To do this we can run Add-TeamUser, New-TeamChannel and a host of Set-Team other settings that aren’t covered here, but you can find more details of these in the Microsoft documentation.

```
Add-TeamUser -GroupId 4a07317c-c860-47db-9a85-00113764d528 -User user@domain.onmicrosoft.com

New-TeamChannel -GroupId 4a07317c-c860-47db-9a85-00113764d528 -DisplayName “Regulatory Compliance”

Id                                                DisplayName          Description
--                                                -----------          -----------
19:77e332ebe56c494181372e509b44251c@thread.skype Regulatory Compliance
```

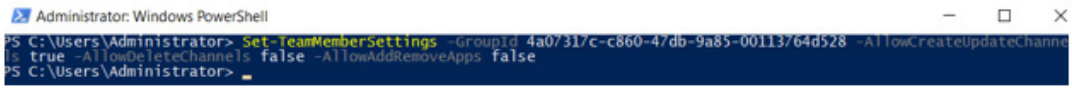


A member has been added to the Team, and a new channel has been created. You will notice that the channel has an ID associated with it that will be needed for the management of that channel.

Modify Settings

I mentioned I would not cover all the settings cmdlets, but I will briefly cover the TeamMemberSettings as an example because I want my users to create channels but not delete them, and I also do not want them to add/remove Apps. I can run Set-TeamMemberSettings to do this.

```
Set-TeamMemberSettings -GroupId 4a07317c-c860-47db-9a85-00113764d528 -AllowCreateUpdateChannels true -AllowDeleteChannels false -AllowAddRemoveApps false
```



I now have my team created 'mostly' as I would like but it has taken me a while and a lot of commands to do this. Let me delete the team.

Remove the Team

Remove-Team -GroupId 4a07317c-c860-47db-9a85-00113764d528

Scripting the Team Creation

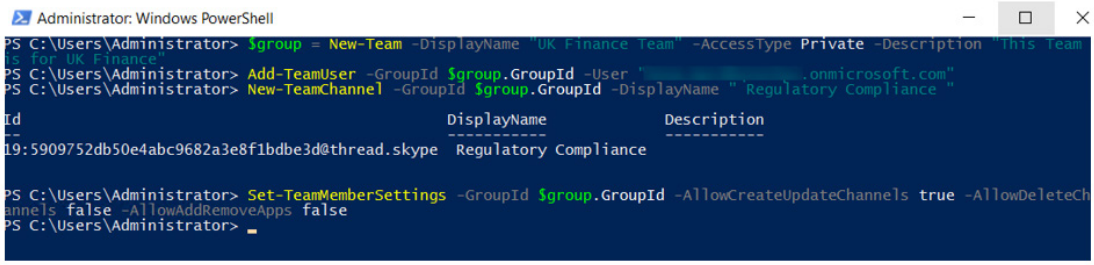
Now I can script the entire thing like this with the groupid captured from the initial creation.

```
$group = New-Team -DisplayName "UK Finance Team" -AccessType Private -Description "This Team is for UK Finance"

Add-TeamUser -GroupId $group.GroupId -User "user@domain.onmicrosoft.com"

New-TeamChannel -GroupId $group.GroupId -DisplayName "Regulatory Compliance"

Set-TeamMemberSettings -GroupId $group.GroupId -AllowCreateUpdateChannels true -AllowDeleteChannels false -AllowAddRemoveApps false
```



That is a bit more straightforward, but it could still be easier. Microsoft has impended a beta feature for now to easily create a team based on a template. However, this will be covered later.

Using the Graph API

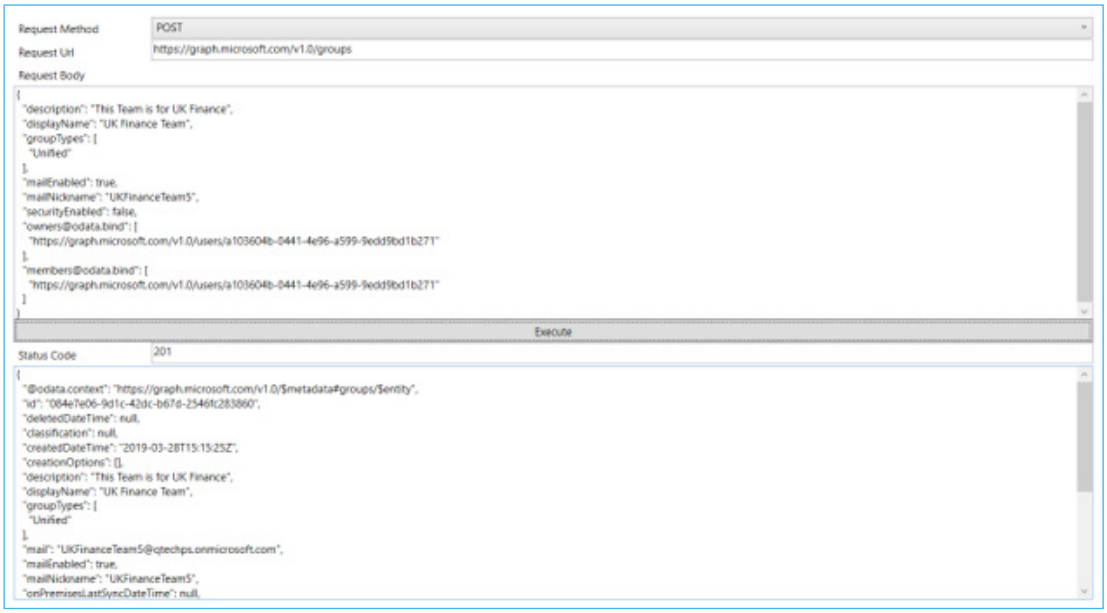
We can do the same thing via the Graph API with either user-level or application-level permissions. If we look at the documentation you can see that, just like PowerShell, there are different resources to create and modify Teams.

Create the Group and add Members

Using the same example but via Graph, I need to create the group before I can create the Team, and the group needs to be an Office 365 Group. I will also add my user as an owner and a member in this post.

POST https://graph.microsoft.com/v1.0/groups
Content-Type: application/json

```
{
  "description": "This Team is for UK Finance",
  "displayName": "UK Finance Team",
  "groupTypes": [
    "Unified"
  ],
  "mailEnabled": true,
  "mailNickname": "operations2019",
  "securityEnabled": false,
  "owners@odata.bind": [
    "https://graph.microsoft.com/v1.0/users/a103604b-0441-4e96-a599-9edd9bd1b271"
  ],
  "members@odata.bind": [
    "https://graph.microsoft.com/v1.0/users/a103604b-0441-4e96-a599-9edd9bd1b271"
  ]
}
```

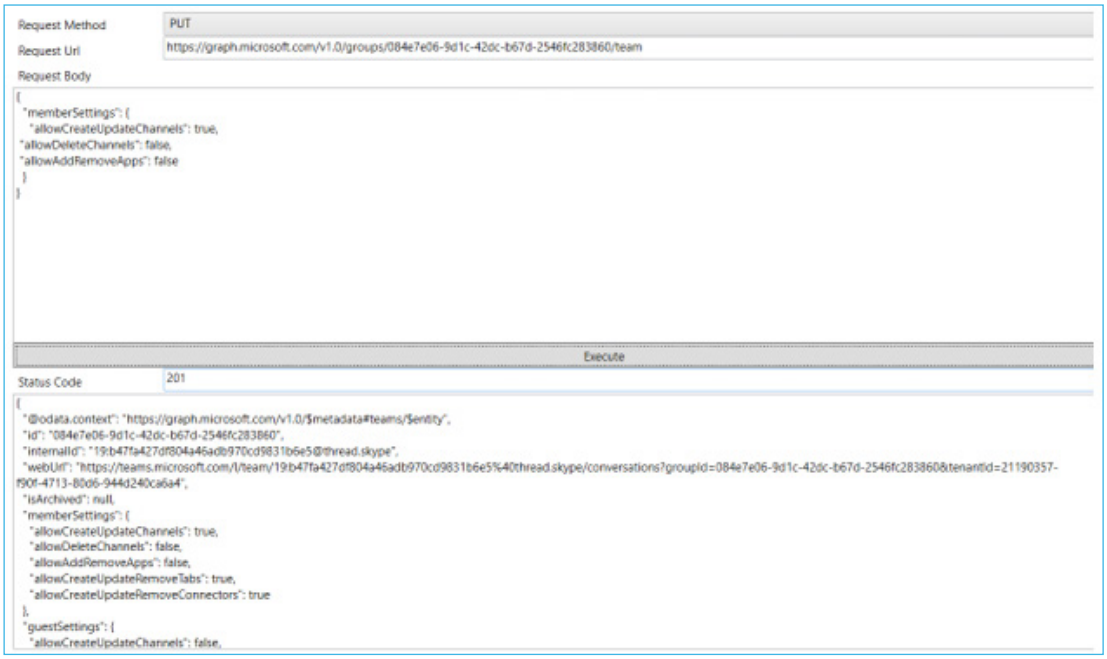


The important part of the response is the ID of the group. In this example it is: "id": "084e7e06-9d1c-42dc-b67d-2546fc283860"

Create the Team with Settings

With the Group ID we can now create the team along with our settings with a Put:
PUT https://graph.microsoft.com/v1.0/groups/084e7e06-9d1c-42dc-b67d-2546fc283860/team
Content-type: application/json

```
{
  "memberSettings": {
    "allowCreateUpdateChannels": true,
          "allowDeleteChannels": false,
          "allowAddRemoveApps": false
  }
}
```



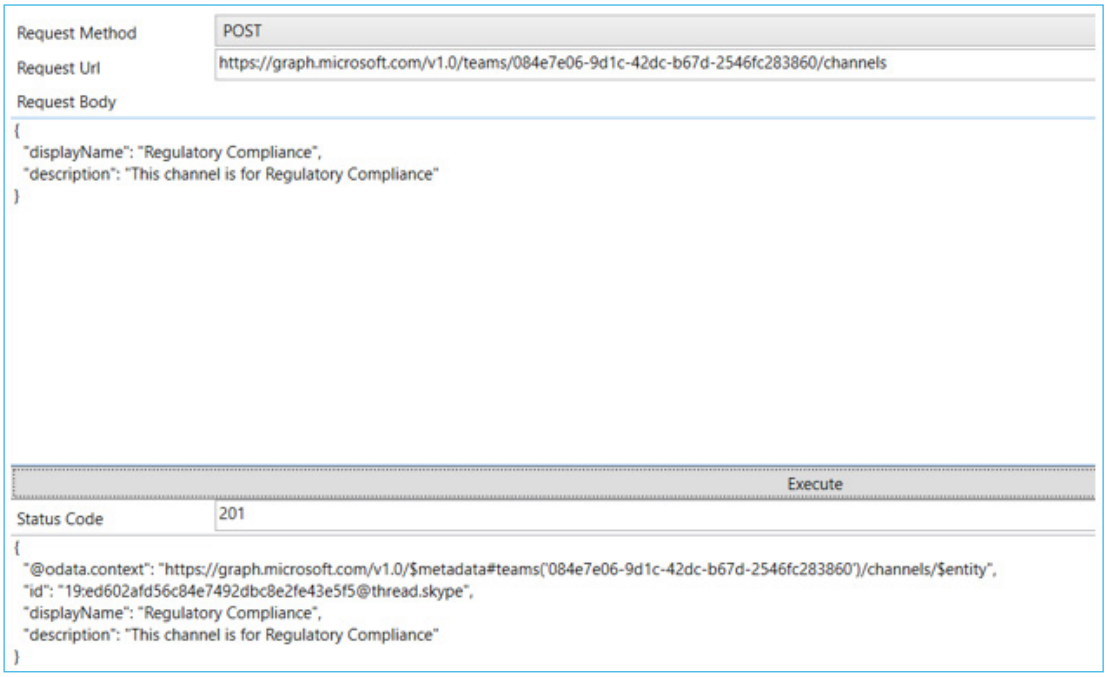
Add the Channel

Now we have a group with members and a team with settings, but we still don't have any Channels. We can now create the channel with a POST.

POST https://graph.microsoft.com/v1.0/teams/084e7e06-9d1c-42dc-b67d-2546fc283860/channels

Content-type: application/json

```
{
  "displayName": "Regulatory Compliance",
  "description": "This channel is for Regulatory Compliance"
}
```

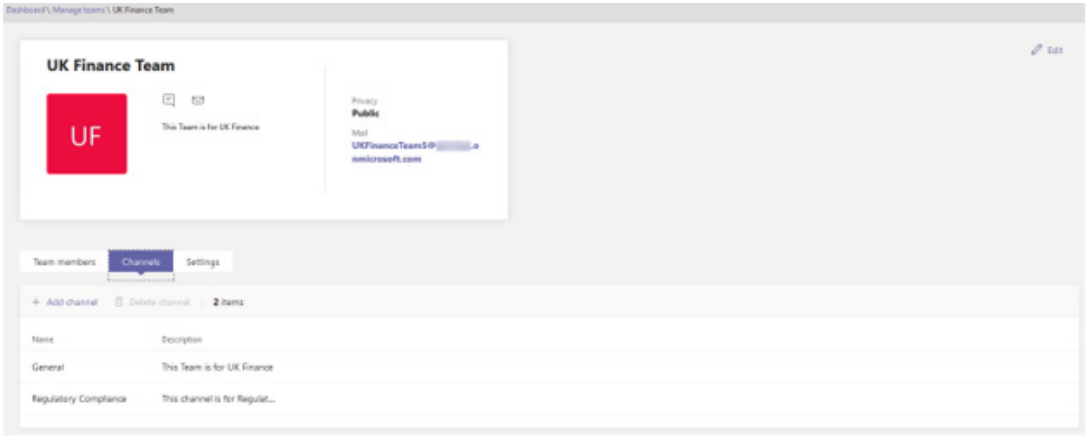


A Single Call?

As you can see, there are a few steps involved to get what we want. So, the next question is, how can we make this simpler? Can we create a Team with the Graph API in a single process - like was done with the scripting in PowerShell? Well, we can now in Beta endpoints using templates. But as I mentioned before, we'll be covering templates later.

Conclusion

You can see that whether you use PowerShell or the Graph, the Team is created with the settings, members, and channels you have defined. I think it is also interesting to do the creation this way to understand the underlying relationship between workloads to create Teams.



Quadrotech

The next article in this series on Teams management will look at the creation of Teams templates, which is currently in Beta. This method delivers a more programmatic approach to Teams creation. Make sure you check back on the blog to see the next installment.

Teams Templates

In a previous section, I walked through the creation of Microsoft Teams using PowerShell and the Graph API. Both of those involve multiple steps to fully create a Team. In that section I mentioned that Microsoft now has Teams Templates in beta and that these templates can be used for a more streamlined process to programmatically create Teams.

What are Teams Templates?

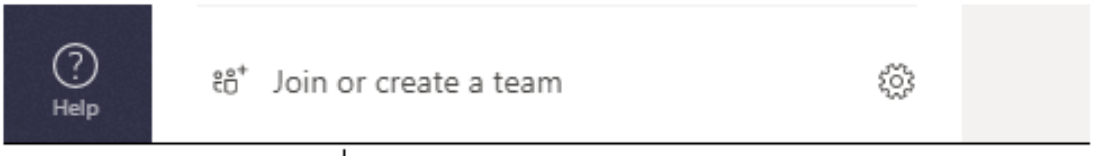
With the introduction of templates, we can now quickly create Teams based on pre-defined templates yet still modify those templates to suit the customer needs (at least via Graph). This is super helpful if you need to create similar Teams with defined channels, members, settings, and apps. It avoids having to start from scratch each time – saving a lot of time. It could also help avoid user error during configuration, because as long as your base template is correctly set up, and you select precisely what you want, you can't go too far wrong!

Important note: *Like everything in Microsoft 365, this feature is changing quickly. While the information here is correct at the time of writing, it is likely to change as Microsoft develops and adds to Teams templates. For the most up to date information, and more details about this feature – check here.*

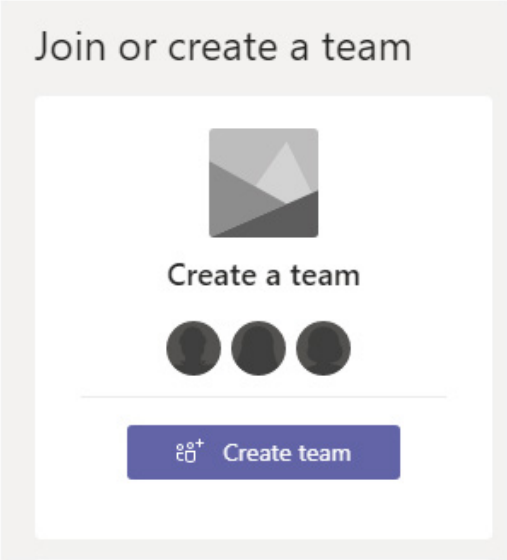
Before Teams Templates: using existing Teams as a template

If you are looking to create a team with a similar style and configuration to an existing team in your tenant, you can use the existing team as a template for the new one. It's a straightforward, but manual process. Here's how to do it.

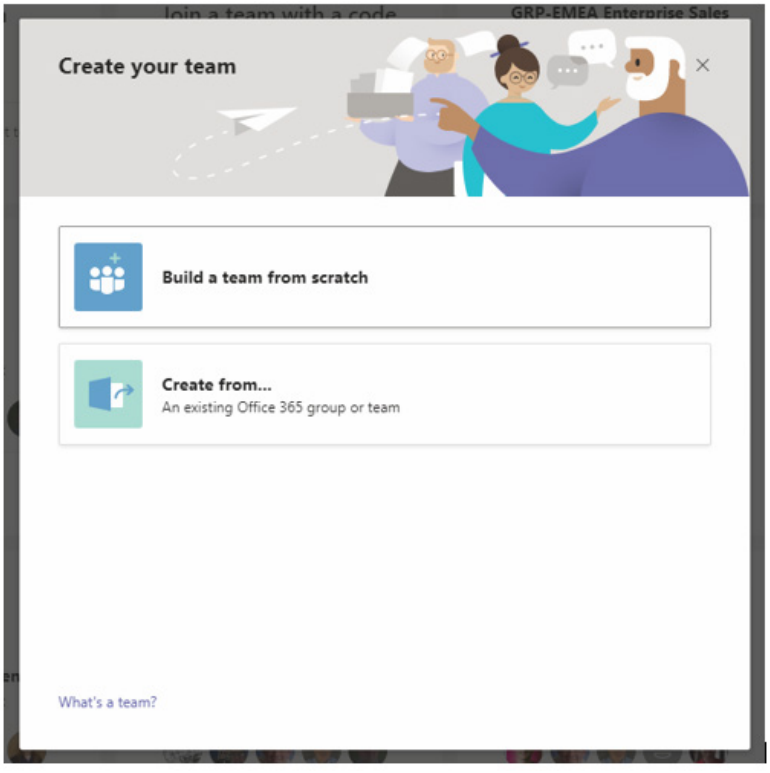
- In the 'Teams' section of the application, click on the 'Join or create a team' option.



- Next, click ‘Create Team’

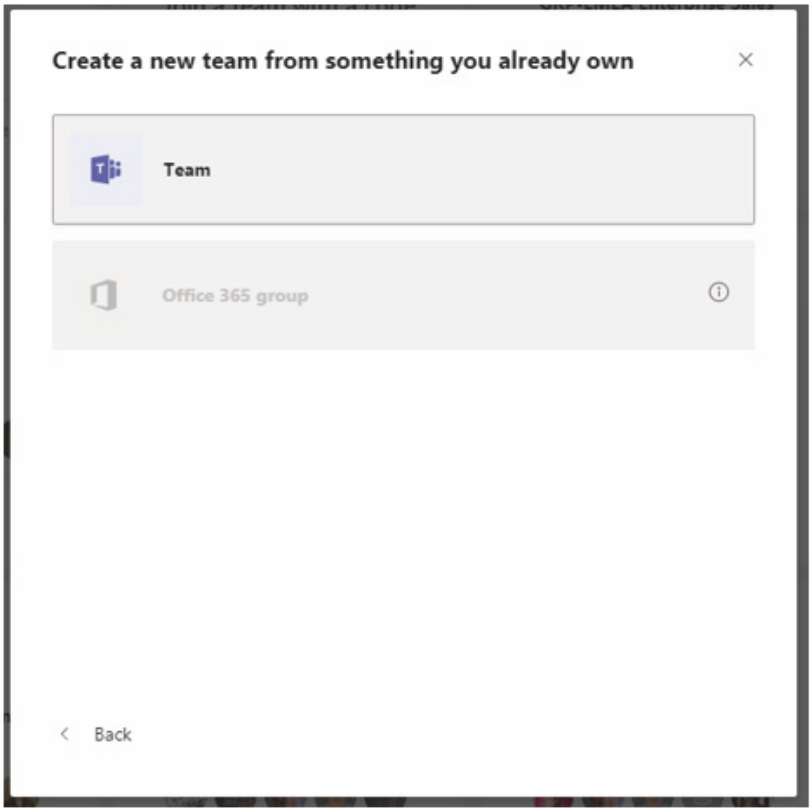


- You will be asked whether you want to build a team from scratch or use an existing Office 365 Group or team. Choose ‘Create from...An existing Office 365 group or team’.



Quadrotech

- Then, you will be asked whether you want to create your new team from a team, or an Office 365 group. Note: You will only be able to replicate an existing team if you are an owner, or belong to that team.



- Whether you choose 'Team' or 'Group' you will be presented with your existing options:

- From there, you can configure your new team with all of the pre-existing settings, to save time, and ensure the correct people, apps, and channels are moved over.

As you can see, this is an extremely simple process, but it is multi-step and highly manual. It also requires time to configure any differing settings you need to add from your existing team. So if you need to create teams regularly, this is not an ideal, nor streamlined process. Enter Teams Templates.

This section explores how to use templates via PowerShell and the Graph. There is a more limited selection of templates available for this method in PowerShell, which we will highlight. As this feature is only in beta currently, we expect to see the capabilities of both approaches increase for GA and beyond.

Base Templates

There are currently 8 base templates, with specific apps and properties for the industries they relate to: Education, Retail, and Healthcare. There is also a standard one, with no additional apps and properties. It is possible to build upon these templates; however, some have properties that cannot be removed/changed. For more information on these, you can take a look at the documentation.

Properties supported by Teams Templates

Here is a list of properties included by templates currently:

- Base template type
- Name
- Description
- Visibility (whether it is a private or public team)
- Team settings
- Auto-favorite channel
- Installed apps
- Pinned tabs

And here is a list of properties that aren't supported yet:

- Team membership
- Team picture
- Channel settings
- Connectors
- Files and content

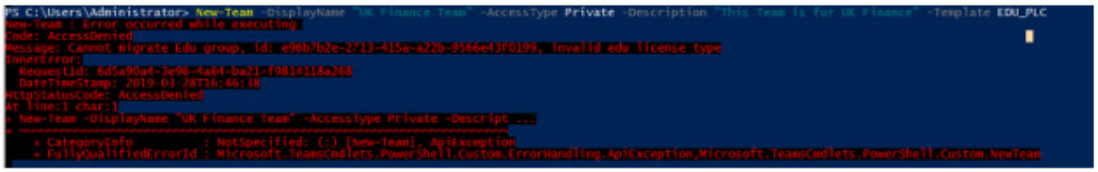
PowerShell and Templates

While it is possible to use all of these templates with the Microsoft Graph, at the time of writing this, only 2 templates are available via PowerShell. Those are Education – Class (EDU_Class) and Education – PLC (EDU_PLC).

As outlined in the documentation, "If you have an EDU license, you can use this parameter to specify which template you'd like to use for creating your group."

Unfortunately, I do not have an EDU license, so I am unable to provide any guidance on those other than to say that they will fail if you do not have an EDU license...

We will return to this topic when we are able to provide first-hand experience.



Graph API and Templates

The usage of Teams Templates via Graph API is supported with either user delegated or application permissions. Either way, you will need access to Group. ReadWrite.All. The only difference in user vs application is when using application-level permissions you will need to specify a user as the owner.

Application Permissions

Application permissions are easily handled by adding in an owner. This is pretty straightforward by using:

```
“owners@odata.bind”: [
   “https://graph.microsoft.com/beta/users/UserID”
 ]
```

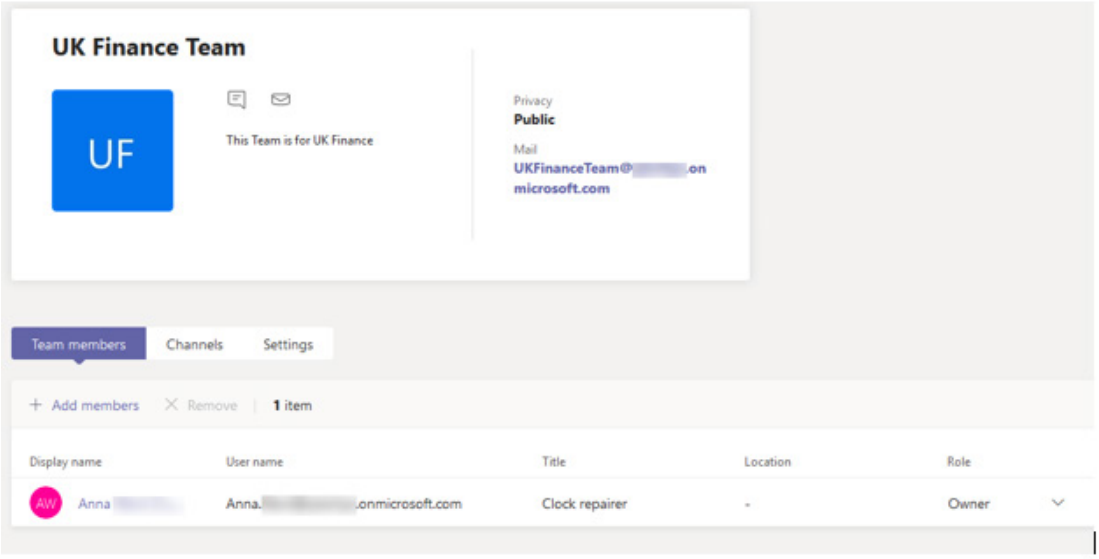
The following standard template request with application permissions results in the team creation.

Request Method POST

Request Url https://graph.microsoft.com/beta/teams

Request Body

```
{
  "template@odata.bind": "https://graph.microsoft.com/beta/teamsTemplates('standard')",
  "displayName": "UK Finance Team",
  "description": "This Team is for UK Finance",
  "owners@odata.bind": [
    "https://graph.microsoft.com/beta/users/a103604b-0441-4e96-a599-9edd9bd1b271"
  ]
}
```



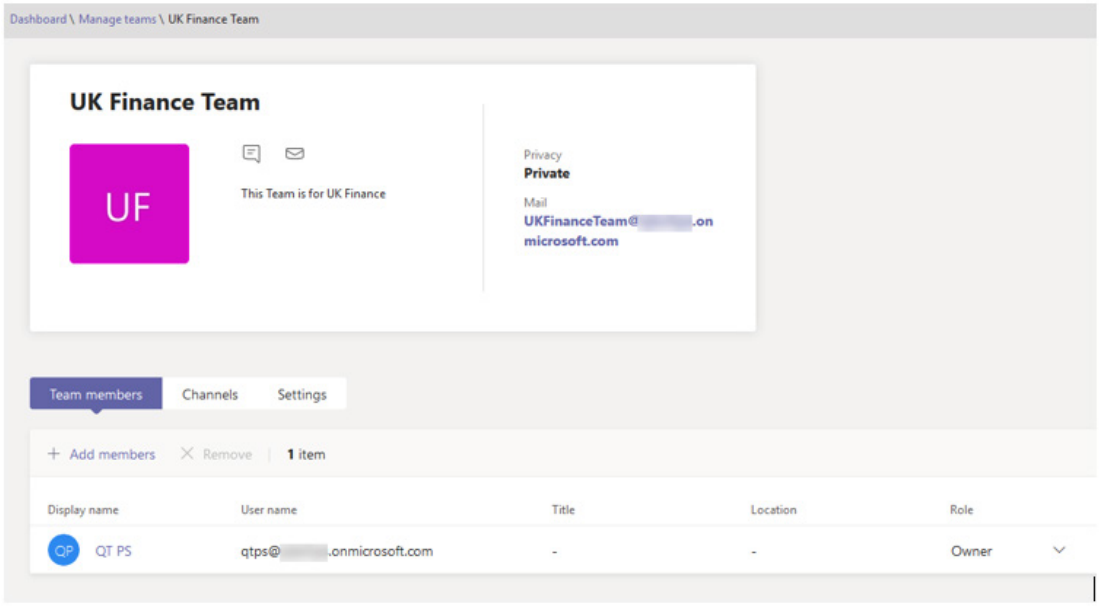
User Permissions with Advanced Requests Using the Standard Template

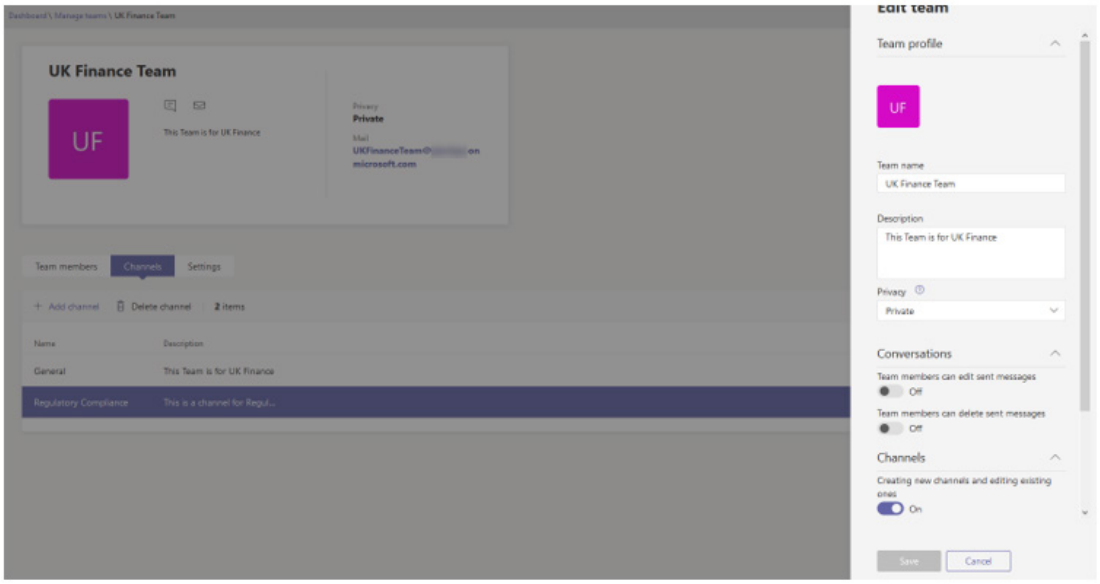
As mentioned before, the payload is basically the same for user permissions except for defining a user as an owner. In the following example, I will create my UK Finance Team with the Regulatory Compliance Channel to set as a Favorite by default. I will also create a UK Payroll Channel with a pinned link to the Payroll company. As in the previous section on Native Management, we will be allowing members to create and update channels, but not remove them. We are also not going to allow removal of apps. Guests will not be able to create or delete channels. Moderate content gifs will be allowed but stickers and memes will not be allowed. Only owners will be allowed to delete messages and team and channel mentions will be allowed.

To do this I simply post the following JSON to the beta teams endpoint.

```
{
  “template@odata.bind”: “https://graph.microsoft.com/beta/teamsTemplates(‘standard’)”,
  “visibility”: “Private”,
  “displayName”: “UK Finance Team”,
  “description”: “This Team is for UK Finance”,
  “channels”: [
    {
      “displayName”: “Regulatory Compliance”,
      “isFavoriteByDefault”: true,
      “description”: “This is a channel for Regulatory Compliance.”,
      “tabs”:  [
        {
          “teamsApp@odata.bind”: “https://graph.microsoft.com/v1.0/appCatalogs/
teamsApps(‘com.microsoft.teamspace.tab.web’)”,
          “name”: “Regulatory Compliance Website”,
          “configuration”: {
            “contentUrl”: “https://en.wikipedia.org/wiki/Regulatory_compliance”
          }
        }
      ]
    }
  ],
  “memberSettings”: {
    “allowCreateUpdateChannels”: true,
    “allowDeleteChannels”: false,
    “allowAddRemoveApps”: false
  },
  “guestSettings”: {
    “allowCreateUpdateChannels”: false,
    “allowDeleteChannels”: false
  },
  “funSettings”: {
    “allowGiphy”: true,
    “giphyContentRating”: “Moderate”,
    “allowStickersAndMemes”: false,
    “allowCustomMemes”: false
  },
  “messagingSettings”: {
    “allowUserEditMessages”: false,
    “allowUserDeleteMessages”: false,
    “allowOwnerDeleteMessages”: true,
    “allowTeamMentions”: true,
    “allowChannelMentions”: true
  },
  “installedApps”: [
    {
      “teamsApp@odata.bind”: “https://graph.microsoft.com/v1.0/appCatalogs/
teamsApps(‘com.microsoft.teamspace.tab.vsts’)”
    }
  ]
}
```

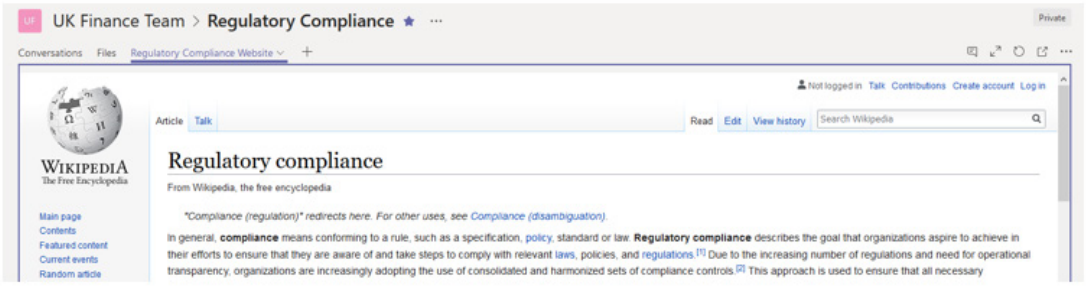
In the Teams admin center, it looks like this:





Then in the Teams client we can see the pinned website and that the channel is a favorite by default.

Quadrotech



Takeaways from API Testing

As a positive, it is a huge time saving to be able to accomplish the creation of a team with settings, channels, apps, etc...in a single payload. This eliminates most of the need for a multi-step process and with the business specific templates it only makes it easier.

I have noticed a few things from testing Teams creation based on Templates. The first is I can only add a single owner during creation. This isn't a huge deal, but it would be a nice option to have multiple owners. The second is that I cannot add owners at the time of the creation of the Team which still makes this a multi-step process albeit much more streamlined than without the templates. Finally, it does take some time for Teams to be created this way. The typical time between submitting the payload and getting a successful response is typically between 3-8 seconds. As this is all still in beta I would anticipate a lot of these items to be addressed before production or as a feature for post v1.

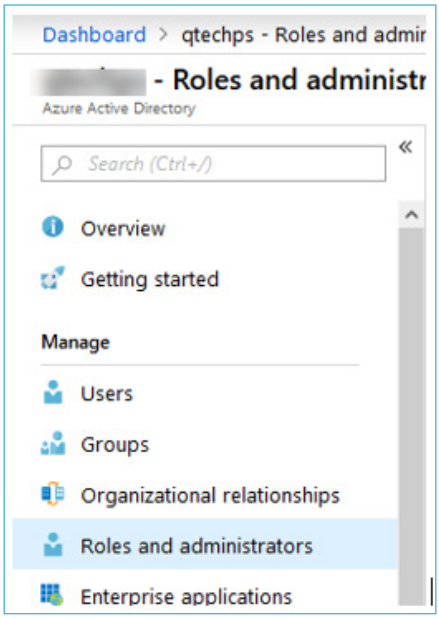
Next up, we will look at how to delegate teams creation.

Delegating Teams Creation

Earlier in this guide, I covered standard creation of teams via PowerShell and Graph API. Next, I tested out a new method for creating Teams that is still in beta, known as Teams Templates. Finally, we're going to wrap up with the most fundamental aspect: delegating teams creation. Because no matter which approach you use for creating or replicating Teams, you need to have the ability to do so. This is achieved with either the permissions granted by the Global Admin Role, Graph API user or application permissions, or delegation through Azure AD roles - which is what this section will cover.

Azure AD Roles for Delegation

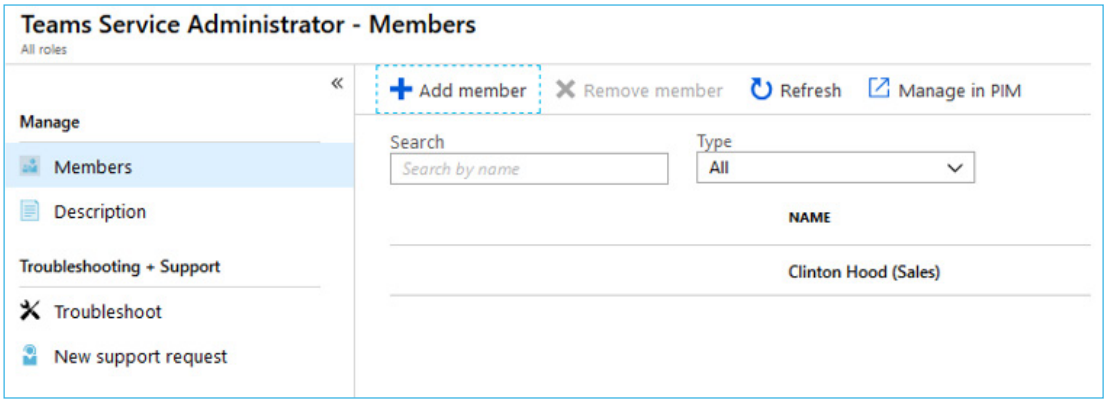
Just to briefly cover delegation with pre-defined AAD roles, simply navigate to Azure AD and then under "Manage" you will see "Roles and administrators".



This is where you will find your various admin roles: License administrator, Billing Administrator, Global Admin, etc... Towards the bottom of that list you will see 4 roles specifically for Teams.
Those roles are:

Role name	Function
Teams Communications Administrator	Manage calling and meetings features
Teams Communications Support Engineer	Troubleshoot communications issues with advanced tools
Teams Communications Support Specialist	Troubleshoot communications issues with basic tools
Teams Service Administrator	Manage the Microsoft Teams service, and manage and create Office 365 Groups

The use case for today is delegation of management, so we will be using the Teams Service Administrator Role. To delegate rights, I can simply add the user I want to manage Teams into this role.



The Teams Service Administrator Role

The first thing I look at when trying to understand the rights granted with a role is to look at the description of that role. This will give me a basic description as well as the role permissions (read/write), and any additional read permissions.

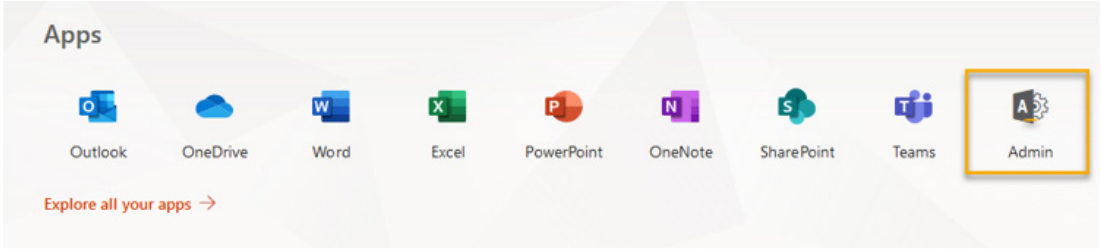
The text for the Teams Service Admin role reads as follows.

"Users in this role can manage all aspects of the Microsoft Teams workload via the Microsoft Teams & Skype for Business admin center and the respective PowerShell modules. This includes, among other areas, all management tools related to telephony, messaging, meetings, and the teams themselves. This role also grants the ability to manage O365 groups."

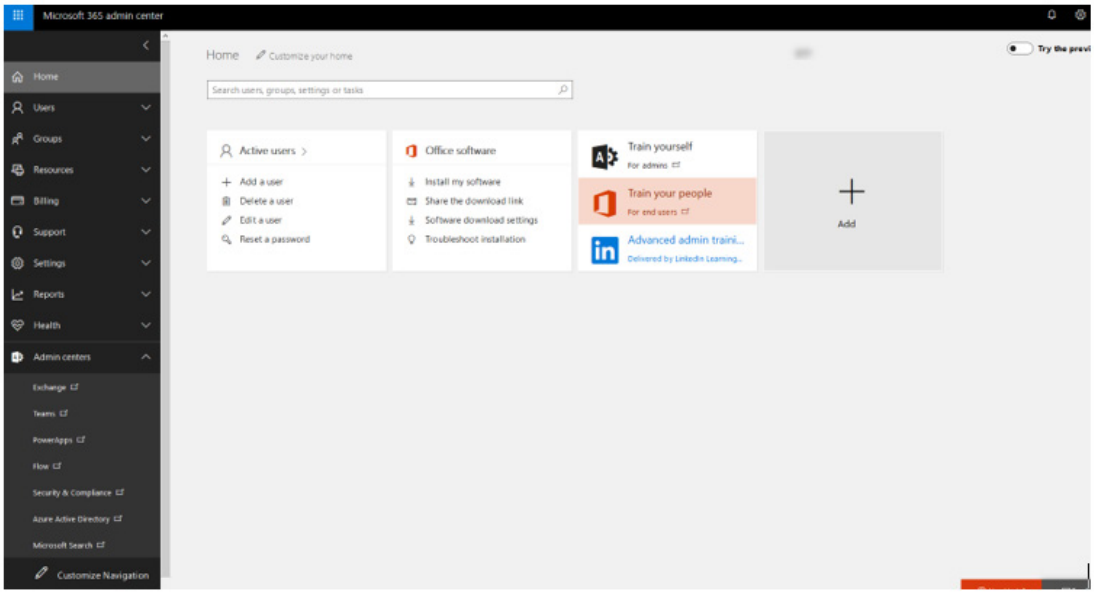
This role is helpful for a lot of reasons. If I have a dedicated Skype for Business person who is transitioning to Teams, this is a great role. It is also great for a dedicated admin/engineer for Teams. I can limit their actions to all things Teams, increasing my overall service security.

How does this work?

Once I add the member to the role, they will now see the admin icon in their apps.

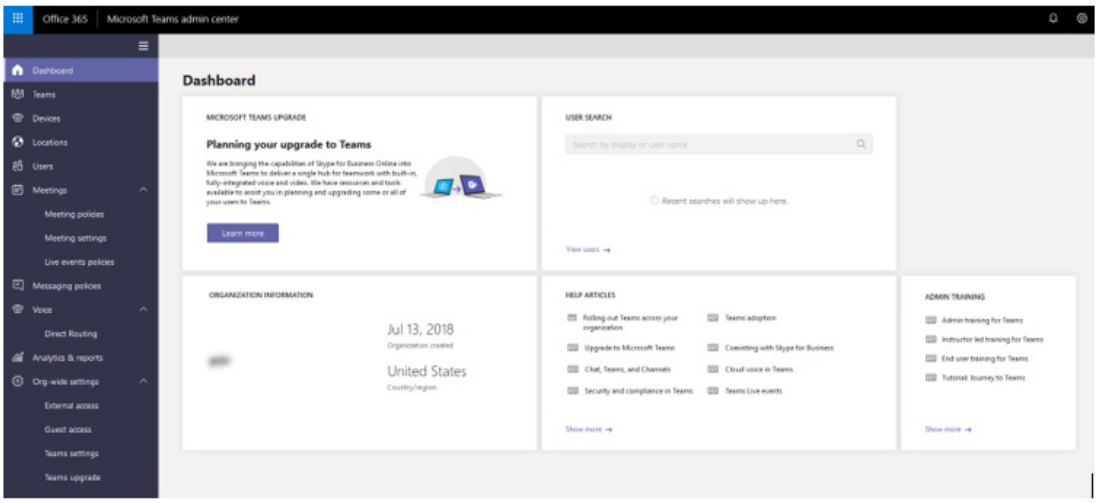


In the Microsoft 365 admin center they will have the following view even though they have just the Teams Service Admin role.



This role will have a full view of the admin center but will only have access to view/modify Users and Groups. This is full access to users and groups. My delegated admin will have the ability now to add Office 365 licenses to users, reset passwords, and do a range of admin tasks for Teams. As Groups and Teams are highly integrated, this role also has full access to create groups and modify members/owners. This includes security groups which really don't have a relation to Teams administration. In the admin section it also gives me access to the Teams admin center which I can also just access at https://admin.teams.microsoft.com

Within the Teams admin center, you have full access to manage Teams as one would expect. This includes access to all teams that have been created and creation of new teams. You can see/update devices, locations, meeting policies, assigned meeting policies, voice settings, and org-wide settings. As stated in the description of the role 'Everything in the Microsoft Teams admin center and associated PowerShell controls'... aka full Teams management.



Granular Delegated Administration

The delegation model above is as far as you can go with native roles for Teams. It provides administration segregation from other Microsoft workloads but doesn't provide the granularity that role-based access controls in a workload like Exchange Online provide. To be fair, Exchange RBAC has had a long time to evolve and Teams is still relatively new. Teams also spans workloads, so it makes reasonable sense that granular native delegations will be difficult to implement.

The Use Case

CEO: "My North America sales team needs to be able to create and modify teams for their department. It is crucial that there is not a delay for new Team creation and Team membership as this has a potential impact to revenue."

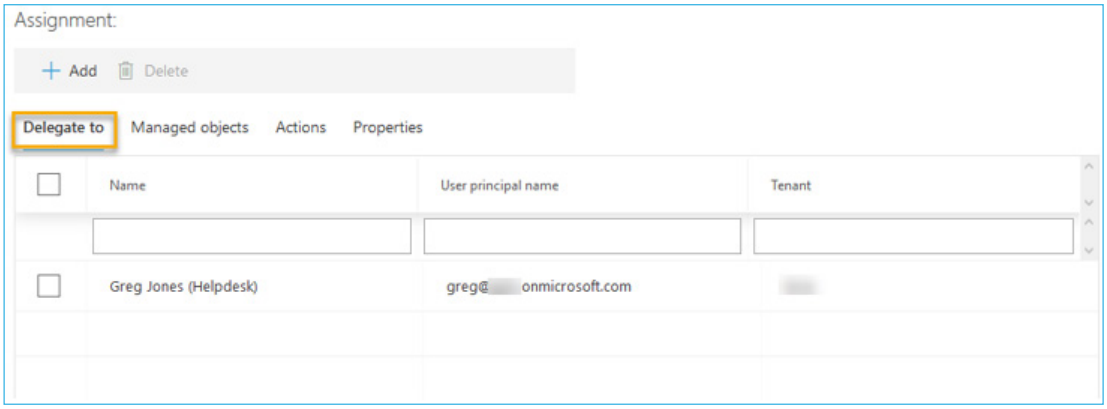
The Solution

I *could* grant my VP of NA Sales membership in the native Teams Service Administrator Role. As we have discussed, this would fulfil the use case the CEO has defined. However, if your VP on NA Sales is like mine, you probably don't want them to have access to telephony, org level settings, or access to ALL the company's Teams. This is contrary to a policy of least-privileged access.

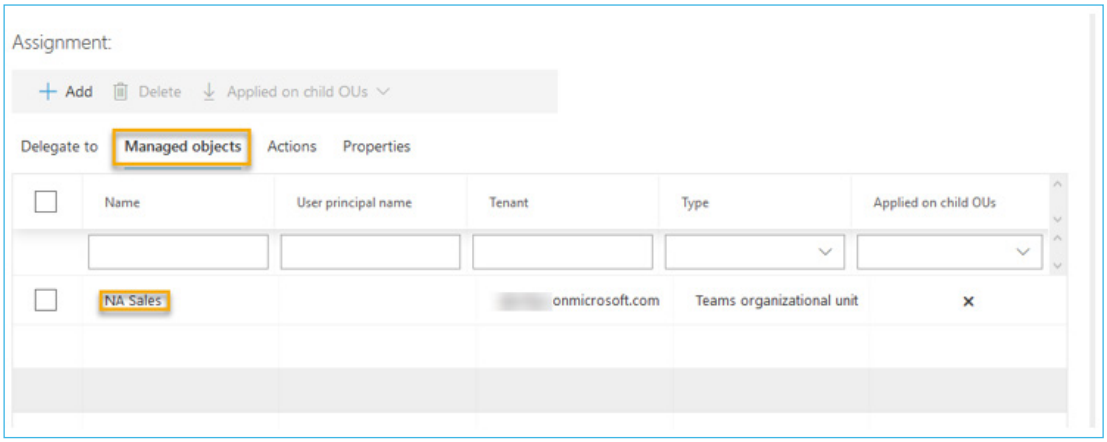
At Quadrotech, we have created a solution to provide granular delegation to Office 365 workloads, Teams being one of those. Let me explain how we can meet the CEO's use case outside of native delegation.

The first thing we do is gather information about what teams exist in the tenant, their settings, along with owners and members. We then allow an administrator to "group" teams together for management. In this example, all NA Sales Teams would be added into a "Teams Group" that will be delegated to the VP on NA Sales. In this example I will create a new Teams Group for NA Sales.

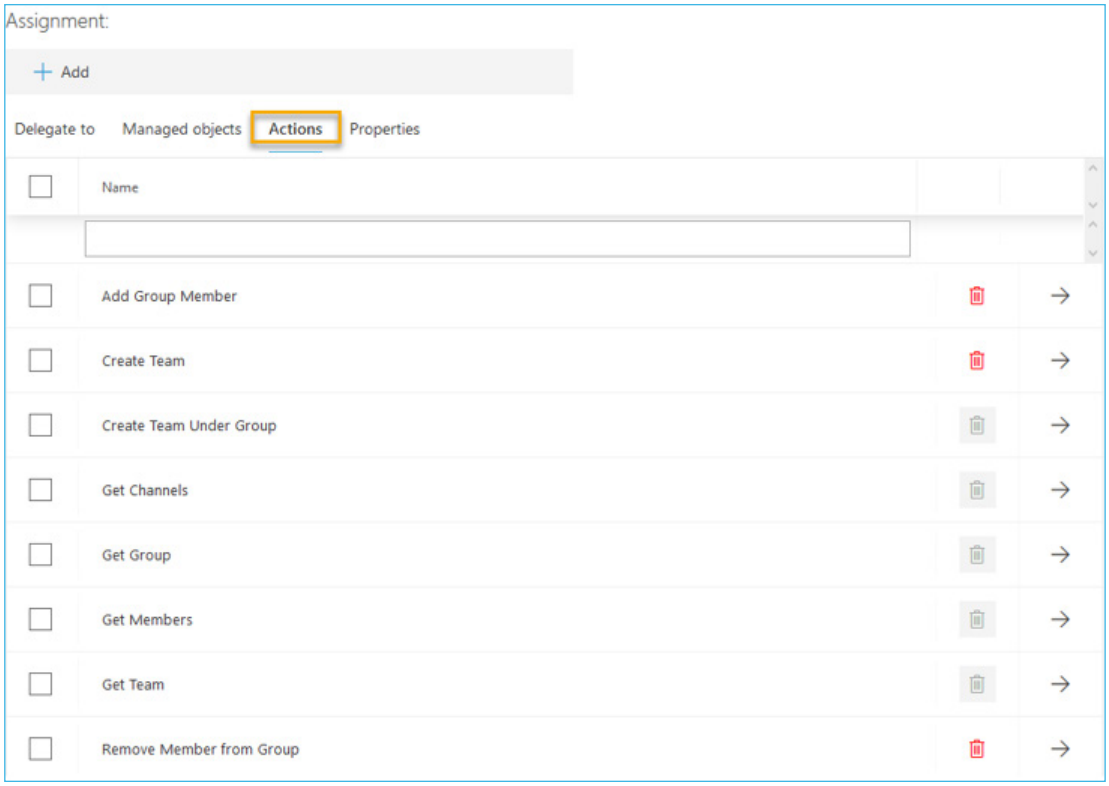
Once the grouping of the Teams is created. We can then delegate just the management of those Teams to the VP of NA Sales. This way they can manage their own Teams and add new Teams within their scope. This is done by creating what we call "Authorization Policies" or "auth policies". The auth policy in this case is defined with who will be given access, also known as "Delegate to" (see image below).



You'll then define what they will have access to. In this case, the NA Teams grouping. This is the "Managed Object."

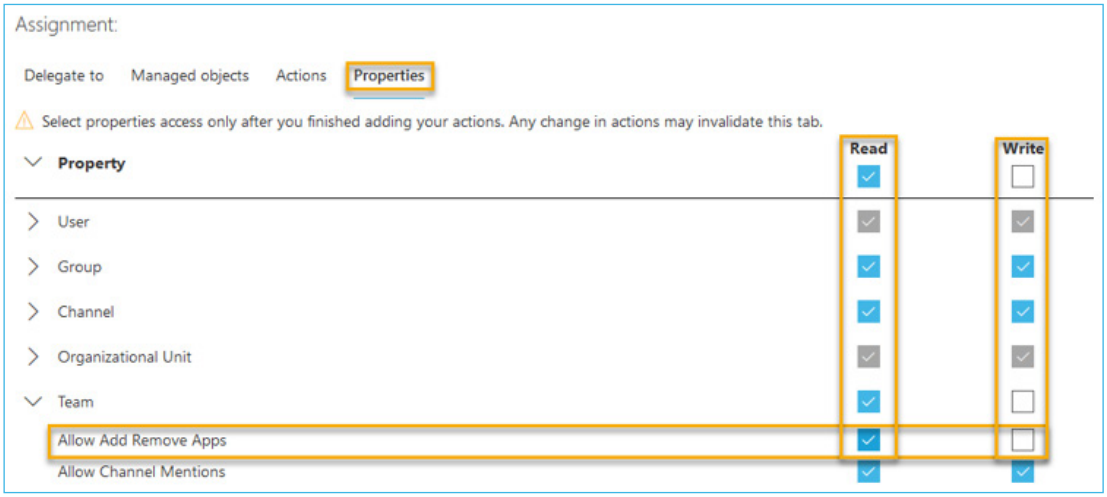


Then we provide the "Actions" or the things we are allowing our delegated admin the rights to do.



Quadrotech

And then finally we can get really granular by defining precisely what the delegated admin can do with users, groups, channels, and teams. In the example below, I've allowed my delegated admin full read/write access to the user, group, and channels. I then allowed the ability for that delegated admin to see the 'Allow Add/Remove Apps' setting but not the ability to change it.



And here is the kicker. This doesn't have to be all within a single tenant. It can be, but in this example, I granted my NA Sales VP to Teams that existed in a second tenant. The solution is not only extremely granular, but also very flexible.

The result

With this delegation, we enable the NA Sales team to manage their Teams, but have restricted them to just their Teams. They do not have the ability to create security groups, change user passwords, modify Teams that don't belong to them, nor the ability to modify any Teams workload settings. It is truly the ability to have a least privilege access model when it comes to Teams as well as other Microsoft Office 365 workloads. We have met the needs of our CEO use case outlined earlier and not granted any additional access that would introduce risk to the organization.

Conclusion

I hope this series has shed some light on a few areas of Teams management that can become challenging. As we've seen and highlighted throughout, Teams is still a relatively new service, with deep integrations between workloads. As a result, management features are developing and improving alongside the tool itself. New capabilities for both the end user and the admin are constantly appearing in the service, and we expect to see further improvements made to features like Teams Templates (currently in beta), and other features as time goes on.

Delegation is an area where there is still a lot of room for improvement. While it might be sufficient for a smaller, or less complex environment, larger organizations with more complex compliance or governance needs may need something that's more robust, granular, while being flexible to specific needs.

As we mentioned, Quadrotech has a powerful solution for delegated administration and simple automation of regular Office 365 tasks, which can be used across multiple tenants. If you would like to find out more about how you can maximise efficiency while keeping your environment secure and compliant, contact us now for a demo and personalised walkthrough.

Alternatively, why not take a look at what industry expert and Microsoft MVP Dominik Hoefling thought when he took this tool for a spin. Download his detailed technical review here.