Microsoft

@ChrisO_Brien

The hit list - Office 365 dev techniques you should know

Chris O'Brien – MVP
Head of Development, Content and Code

What happened recently?

Some changes in last 6 months or so..

SharePoint developments

SharePoint provisioning service, Site Designs, Page Designs, navigation, web parts etc.

New Teams capabilities

SharePoint/ Teams integration

Templated teams

PowerApps/ Flow

PowerApps “components”

Responsive apps

Licensing changes

Microsoft Graph

Teams provisioning APIs

Security APIs

More app-only support

Other

Azure Functions v2

Managed Service Identity

PowerShell v2

Maximum effectiveness in 2019

You need:

- SharePoint (e.g. Site Designs, modern web parts, header/nav etc.)
- SPFx dev – including React, SCSS, security etc.
- Graph!
- Azure (e.g. Functions, AAD app registrations, Key Vault etc.)
- PowerApps and Flow
- Teams dev – including templating, tabs and bots etc.

Office 365 dev scenarios to master (top of mind May 2019)

Chris O'Brien – www.sharepointnutsandbolts.com

SPFx/ Azure Functions:	SharePoint	Azure Functions (general)	Power platform
•- Work effectively with the Graph •- Call AAD-secured function from SPFx (using AadHttpClient) •- Call AAD-secured function from SPFx isolated web part •- SPFx formats: • Application customizer • Web parts • Field customizers • ListView command set	- Implement custom site design which calls Flow to apply PnP template (with app-only auth) - Implement JSON view/column formatting	- Use Managed Service Identity with Azure Functions/Key Vault - Use App Insights in Azure Functions	- Create custom connector to call Azure Function from PowerApps/Flow - Implement offline support in PowerApps - Implement on-premises data gateway for PowerApps/Flow/Power BI - Store image from PowerApps Camera/Pen control into SharePoint

Office 365 dev scenarios to master (top of mind May 2019)

Chris O'Brien – www.sharepointnutsandbolts.com

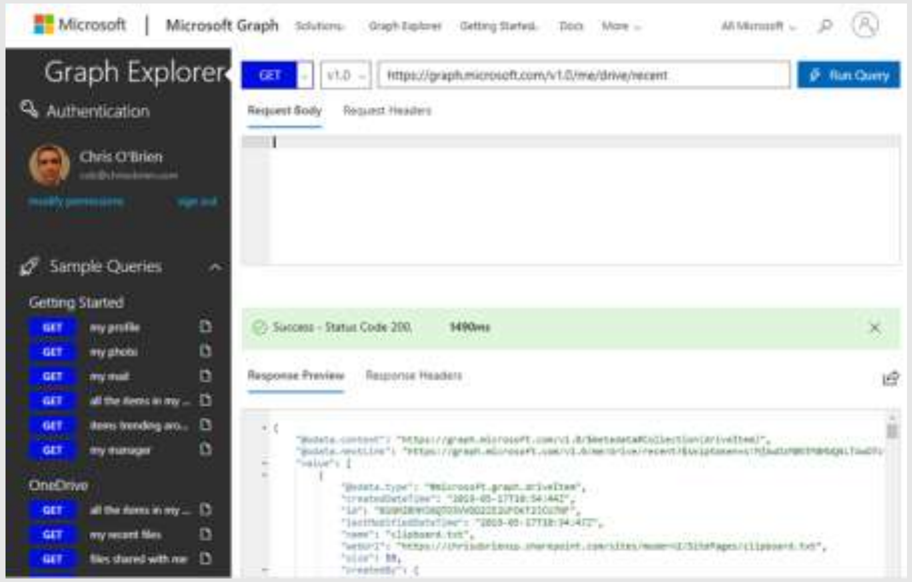
SPFx/ Azure Functions:	SharePoint	Azure Functions (general)	Power platform
•- **Work effectively with the Graph** •- Call AAD-secured function from SPFx (using AadHttpClient) •- **Call AAD-secured function from SPFx isolated web part** •- SPFx formats: • Application customizer • Web parts • Field customizers • ListView command set	**- Implement custom site design which calls Flow to apply PnP template (with app-only auth)** *- Implement JSON view/column formatting*	*- Use Managed Service Identity with Azure Functions/Key Vault* *- Use App Insights in Azure Functions*	*- Create custom connector to call Azure Function from PowerApps/Flow* *- Implement offline support in PowerApps* *- Implement on-premises data gateway for PowerApps/Flow/Power BI* **- Store image from PowerApps Camera/Pen control into SharePoint** - Flow controller pattern

Work effectively with the Graph

Technique 1

Graph Explorer is great, but…

- Can’t use your own AAD app registration and permissions
- Using GE **isn’t** doing the same thing your code is doing



```
{
    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#Collection(driveItem)",
    "@odata.nextLink": "https://graph.microsoft.com/v1.0/me/drive/recent?$skiptoken=[illegible]
    "value": [
        {
            "@odata.type": "#microsoft.graph.driveItem",
            "createdDateTime": "2019-05-17T18:54:44Z",
            "id": "[illegible]",
            "lastModifiedDateTime": "2019-05-17T18:54:47Z",
            "name": "clipboard.txt",
            "webUrl": "https://chrisobriensp.sharepoint.com/sites/[illegible]/SitePages/clipboard.txt",
            "size": 88,
            "createdBy": {
```

The answer - Use Postman with your AAD app reg!

- Microsoft provide Postman collections to import
- Ability to call use app-only or delegated/user permissions

Using Postman with the Graph

Create your AAD app reg

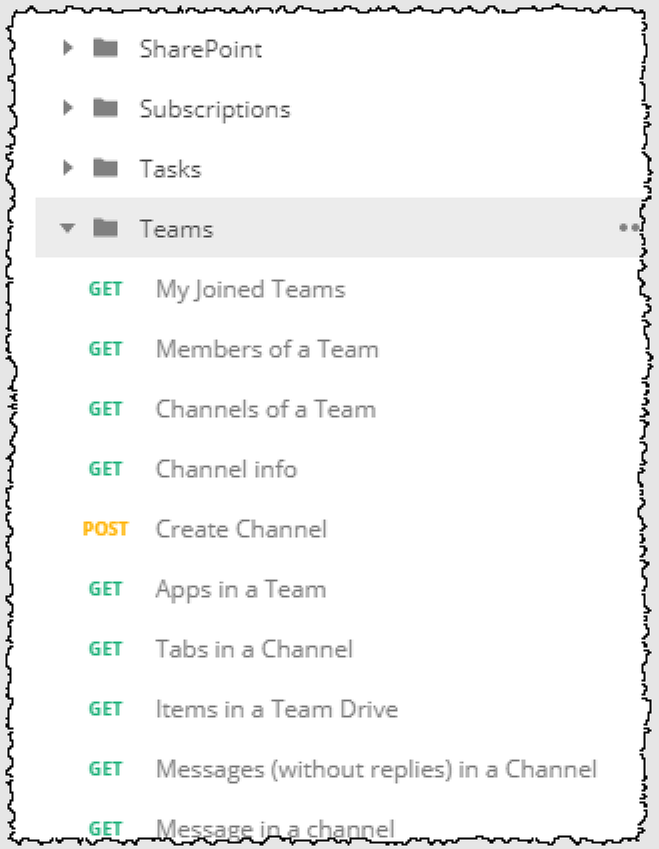
Grant appropriate app-only/delegated permission scopes and provide admin consent

IMPORTANT! Add reply URL to AAD app reg:

https://app.getpostman.com/oauth2/callback

Import Postman collections from Github

Enter Client ID/Client Secret/Tenant ID as Postman variables



SharePoint
Subscriptions
Tasks
Teams

GET My Joined Teams
GET Members of a Team
GET Channels of a Team
GET Channel info
POST Create Channel
GET Apps in a Team
GET Tabs in a Channel
GET Items in a Team Drive
GET Messages (without replies) in a Channel
GET Message in a channel

https://github.com/microsoftgraph/microsoftgraph-postman-collections

Working in TypeScript with types from Graph

Option 1 – roll your own

Option 2 – use "paste JSON as code"

https://marketplace.visualstudio.com/items?itemName=quicktype.quicktype

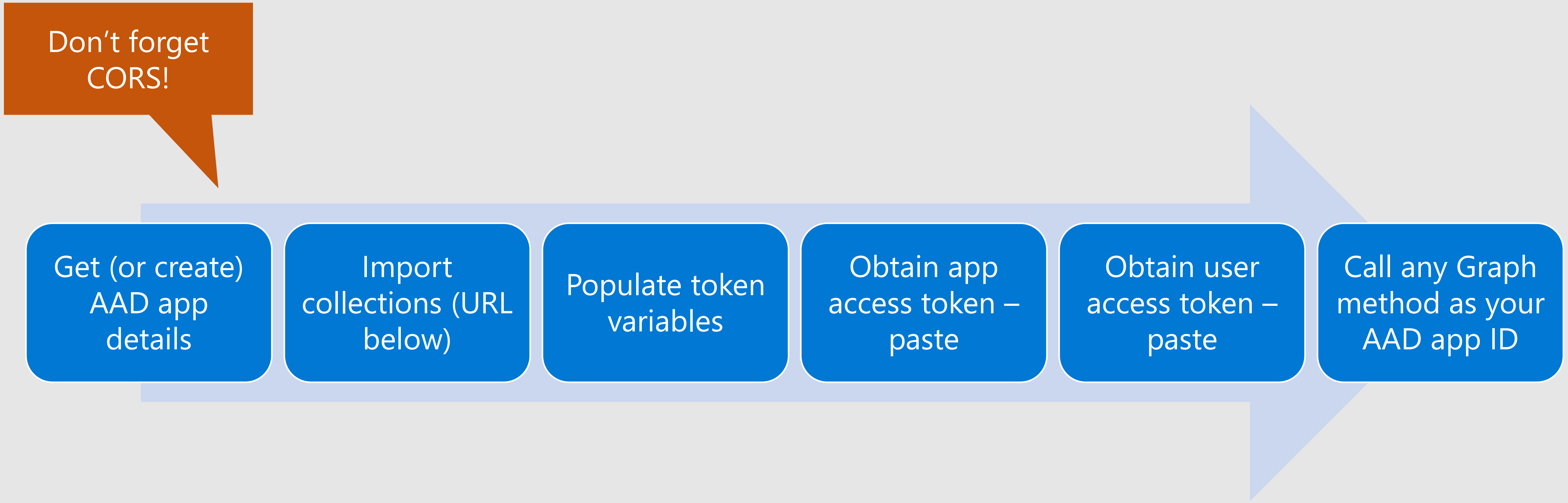
Option 3 – use Microsoft Graph TypeScript Types

https://github.com/microsoftgraph/msgraph-typescript-typings

```
npm install @microsoft/microsoft-graph-types --save-dev
npm install @microsoft/microsoft-graph-types-beta --save-dev
```

DEMO – using Postman with the Graph

Postman summary


Don't forget CORS!
Get (or create) AAD app details
Import collections (URL below)
Populate token variables
Obtain app access token – paste
Obtain user access token – paste
Call any Graph method as your AAD app ID

https://github.com/microsoftgraph/microsoftgraph-postman-collections

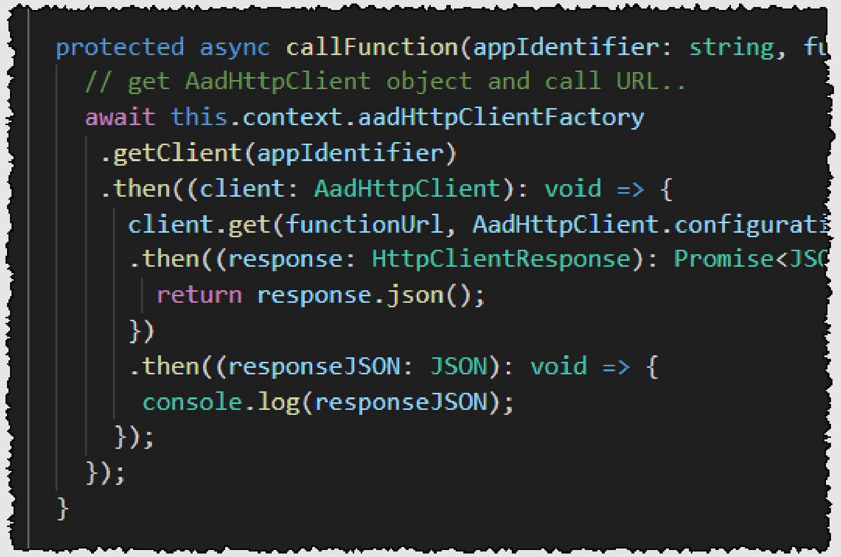
Call your secure web API (or the Graph) from SPFx

Technique 2

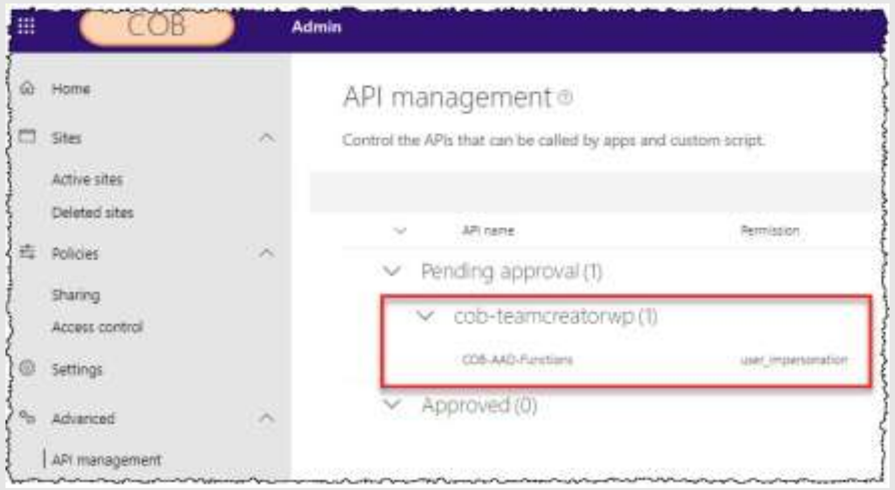
SPFx and web APIs (e.g. Azure Functions)

Fundamentals

1. Create an Azure Function, secure with AAD auth
2. Declare permissions in SPFx manifest
3. Use AadHttpClient in code to connect to your API
4. Deploy SPFx package, consent to permissions



```
protected async callFunction(appIdentifier: string, fu
  // get AadHttpClient object and call URL..
  await this.context.aadHttpClientFactory
   .getClient(appIdentifier)
   .then((client: AadHttpClient): void => {
     client.get(functionUrl, AadHttpClient.configurati
     .then((response: HttpClientResponse): Promise<JSO
       return response.json();
     })
     .then((responseJSON: JSON): void => {
      console.log(responseJSON);
    });
  });
}
```



API management

Control the APIs that can be called by apps and custom script.

API name	Permission
Pending approval (1)	
cob-teamcreatorwp (1)	
COB-AAD-Functions	user_impersonation
Approved (0)	

Calling the Graph from SPFx

Are you calling direct from TypeScript, or via an Azure Function?

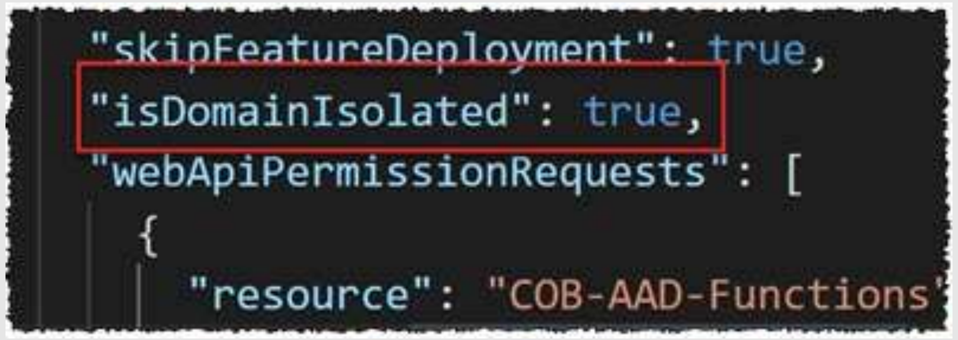
Approach	How
SPFx -> Graph	• Use AadHttpClient or MsGraphClient in code • Add WebApiPermissions to SPFx manifest – Calendar.Read, Group.ReadWrite.All etc. • **Call Graph endpoint directly from SPFx code**
SPFx -> Azure Function -> Graph	• Implement AAD auth on Function • Add WebApiPermissions to SPFx manifest – **user_impersonation** • **Call Azure Function from SPFx code** • **Inside the function - exchange passed token for one which can call Graph**

Using SPFx isolated web parts to call APIs/Graph

Beneficial because:

Permissions only apply to THIS app package – not shared across SPFx

As simple as a standard SPFx web part

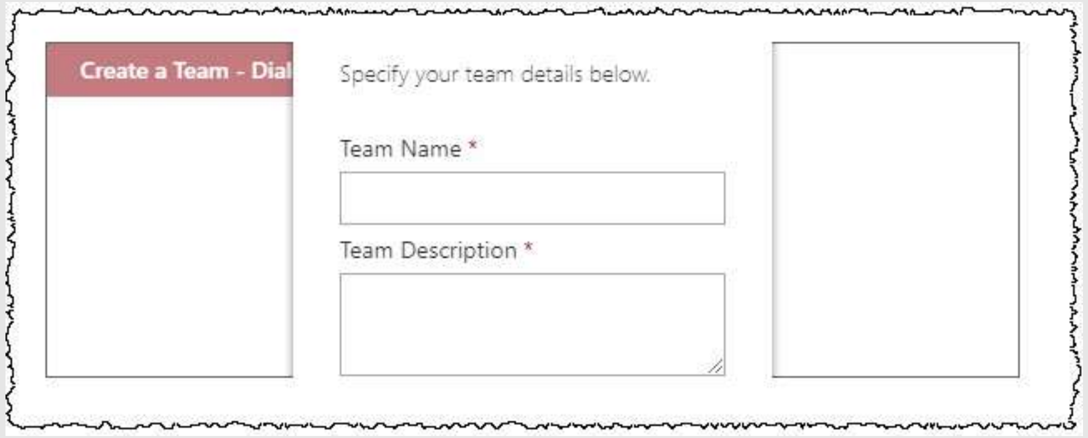


```
"skipFeatureDeployment": true,
"isDomainIsolated": true,
"webApiPermissionRequests": [
  {
    "resource": "COB-AAD-Functions
```

Considerations:

-> iFrame = some UX considerations

-> cannot use OUIF Panel, Layer, Dialog etc., unless surface area is large



DEMO – SPFx isolated web parts with Azure Function

PowerApps to collect images

Technique 3

Gathering photos and signatures with PowerApps

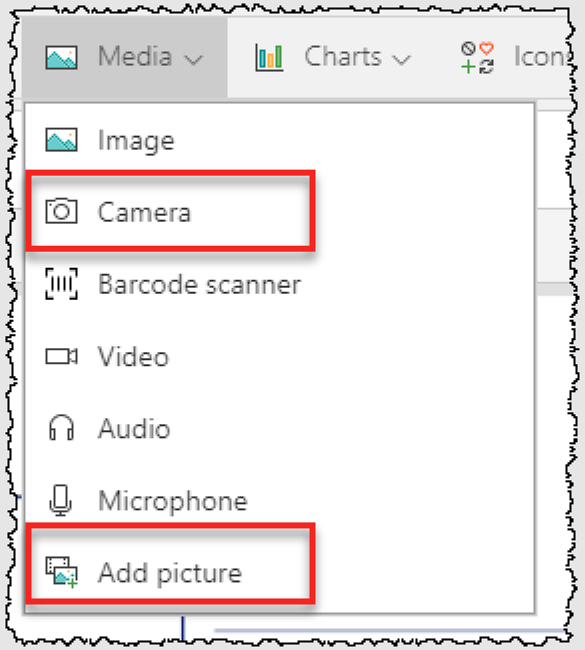
Some work required to save to SharePoint, but not much..

Use “call Flow from PowerApps” approach – no code required

The key – **Flow’s ‘dataUriToBinary’ function**

Signatures (pen input control) are harder ☹

Need an Azure Function here – different format URI

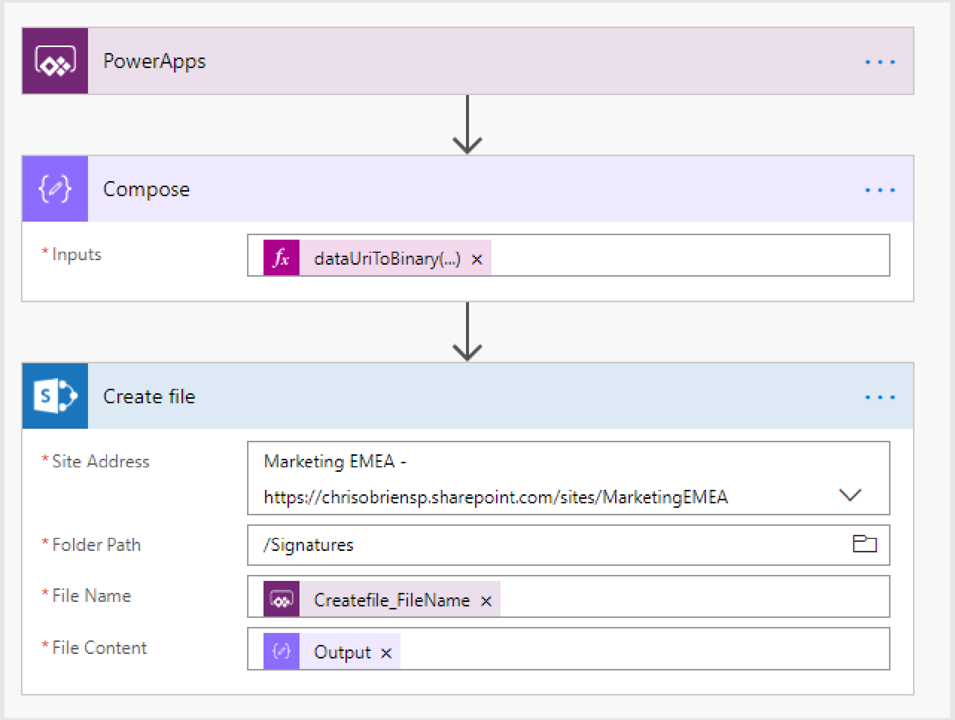


http://johnliu.net/blog/2017/5/taking-a-picture-with-powerapps-and-sending-to-sharepoint-with-just-flow

Gathering photos - detail

Fundamentals

1. Create a PowerApp which uses the Camera control. Save photo to collection.
2. Create a Flow which has PowerApps trigger – use dataUriToBinary()
3. Call Flow from your PowerApp – pass item from collection to Flow.



PowerApps

Compose

* Inputs: dataUriToBinary(...)

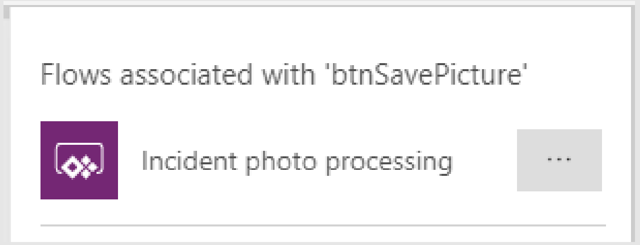
Create file

* Site Address: Marketing EMEA - https://chrisobriensp.sharepoint.com/sites/MarketingEMEA

* Folder Path: /Signatures

* File Name: Createfile_FileName

* File Content: Output



Flows associated with 'btnSavePicture'

Incident photo processing

DEMO – Gathering photos with PowerApps and Flow

Site Designs and PnP templating

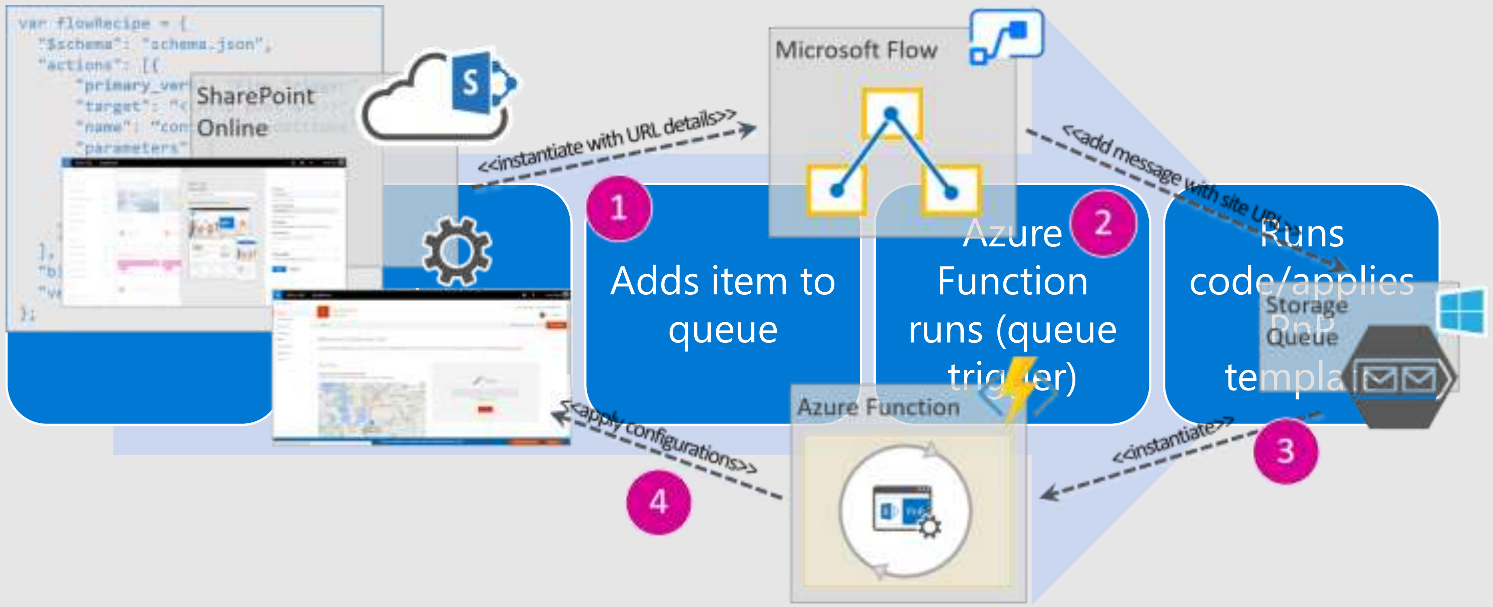
Technique 4

Site design capabilities

- Site settings – theme, logo, regional settings, permissions, external sharing
- Create lists and libraries (with columns, content types, views etc.)
- Install SPFx solution (e.g. web part)
- Register an SPFx extension (e.g. header, footer)
- **Trigger a Flow**
- Join a hub site (when site created from this template)
- Still big gaps compared to PnP!

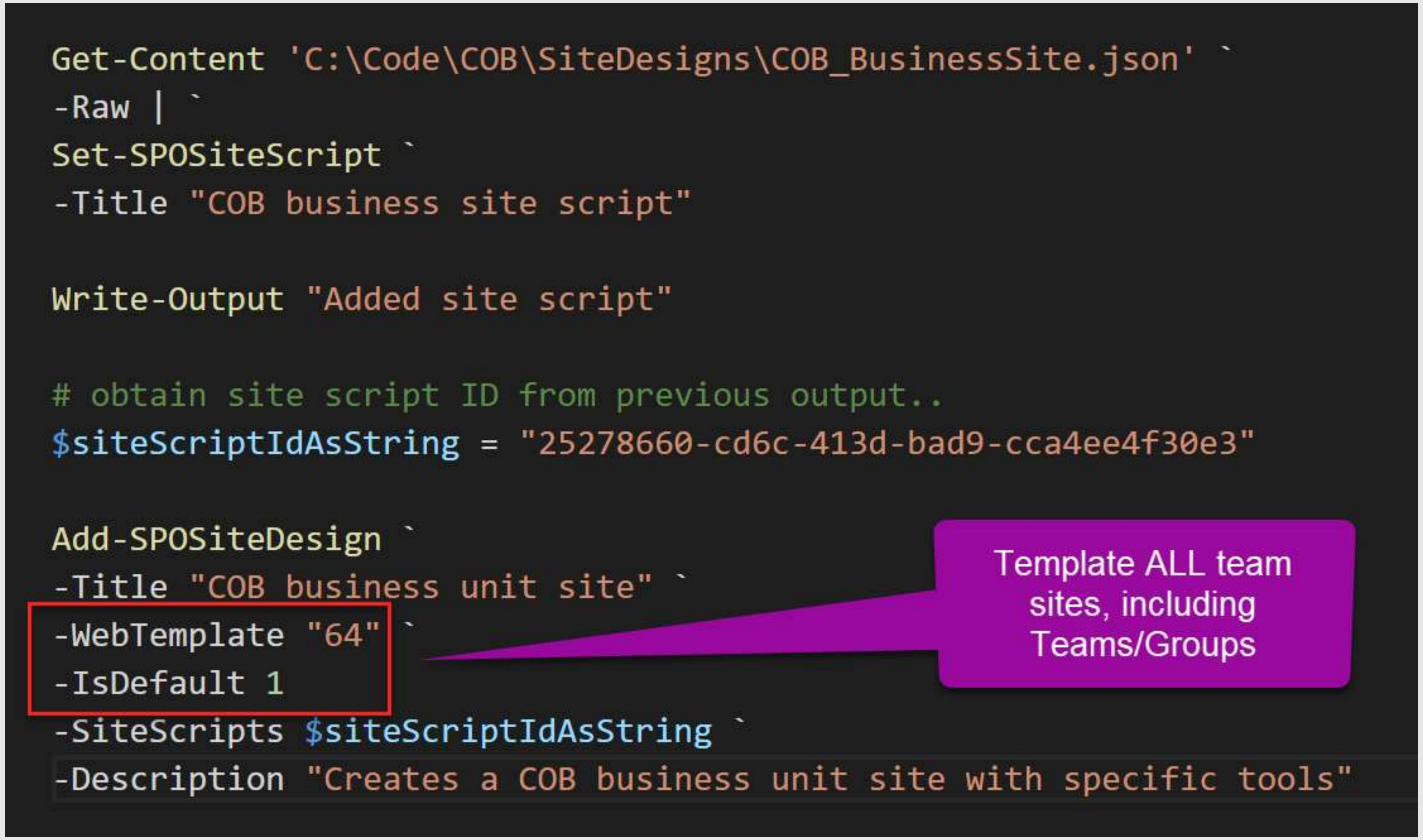
But what happens when I need more?

Answer – you plug in an Azure Function to apply a PnP template!


SharePoint Online
Microsoft Flow
<<instantiate with URL details>>
<<add message with site URL>>
1
Adds item to queue
2
Azure Function runs (queue trigger)
Runs code/applies PnP template
Storage Queue
3
<<instantiate>>
Azure Function
4
<<apply configurations>>

See https://docs.microsoft.com/en-us/sharepoint/dev/declarative-customization/site-design-trigger-flow-tutorial

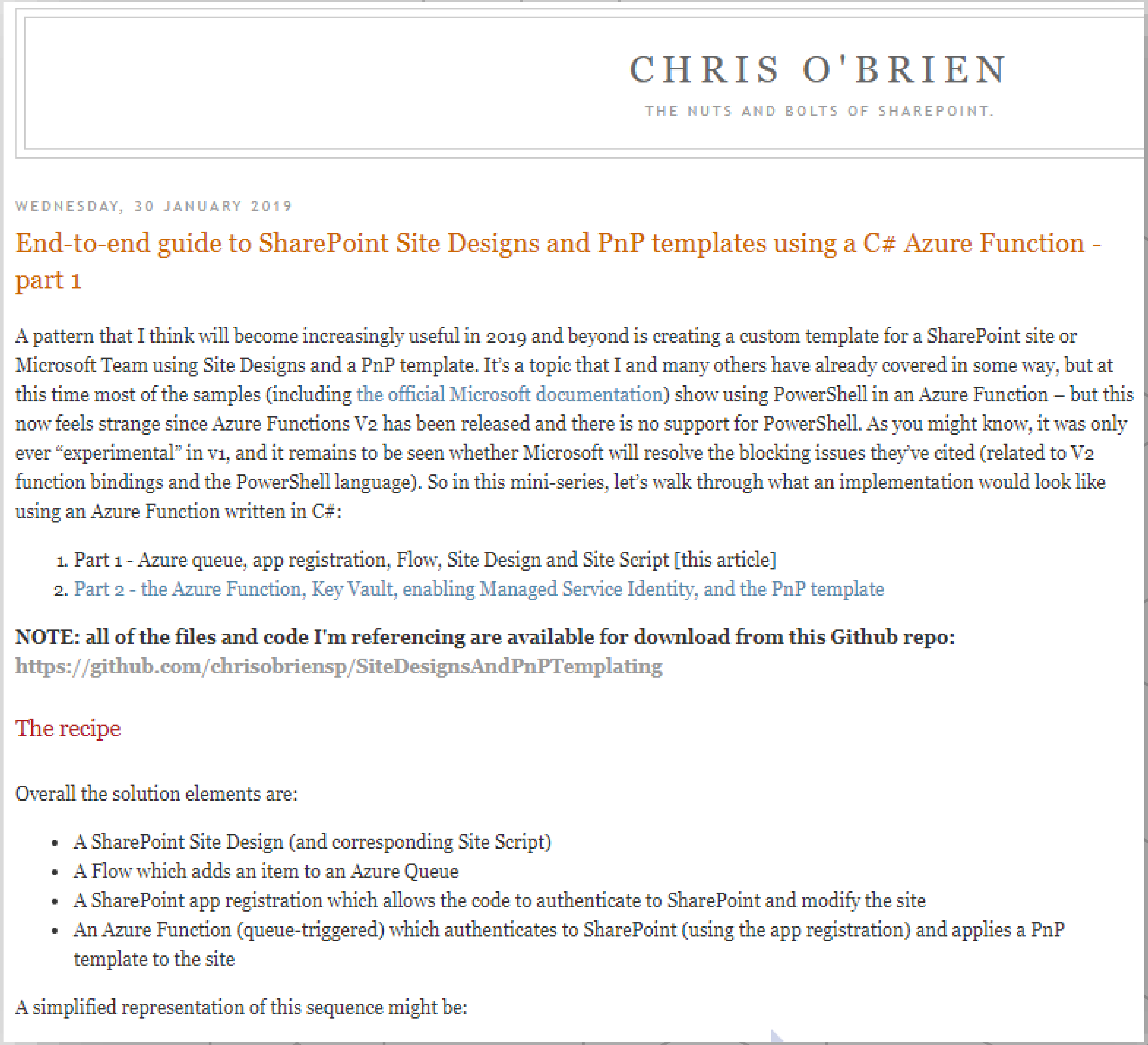
Add a Site Design/Site Script:



```
Get-Content 'C:\Code\COB\SiteDesigns\COB_BusinessSite.json' `
-Raw | `
Set-SPOSiteScript `
-Title "COB business site script"

Write-Output "Added site script"

# obtain site script ID from previous output..
$siteScriptIdAsString = "25278660-cd6c-413d-bad9-cca4ee4f30e3"

Add-SPOSiteDesign `
-Title "COB business unit site" `
-WebTemplate "64" `
-IsDefault 1
-SiteScripts $siteScriptIdAsString `
-Description "Creates a COB business unit site with specific tools"
```

Detailed blog post:

https://cob-sp.com/Site-Designs-PnP-1



CHRIS O'BRIEN

THE NUTS AND BOLTS OF SHAREPOINT.

WEDNESDAY, 30 JANUARY 2019

End-to-end guide to SharePoint Site Designs and PnP templates using a C# Azure Function - part 1

A pattern that I think will become increasingly useful in 2019 and beyond is creating a custom template for a SharePoint site or Microsoft Team using Site Designs and a PnP template. It's a topic that I and many others have already covered in some way, but at this time most of the samples (including the official Microsoft documentation) show using PowerShell in an Azure Function – but this now feels strange since Azure Functions V2 has been released and there is no support for PowerShell. As you might know, it was only ever "experimental" in v1, and it remains to be seen whether Microsoft will resolve the blocking issues they've cited (related to V2 function bindings and the PowerShell language). So in this mini-series, let's walk through what an implementation would look like using an Azure Function written in C#:

1. Part 1 - Azure queue, app registration, Flow, Site Design and Site Script [this article]
2. Part 2 - the Azure Function, Key Vault, enabling Managed Service Identity, and the PnP template

NOTE: all of the files and code I'm referencing are available for download from this Github repo: https://github.com/chrisobriensp/SiteDesignsAndPnPTemplating

The recipe

Overall the solution elements are:

- A SharePoint Site Design (and corresponding Site Script)
- A Flow which adds an item to an Azure Queue
- A SharePoint app registration which allows the code to authenticate to SharePoint and modify the site
- An Azure Function (queue-triggered) which authenticates to SharePoint (using the app registration) and applies a PnP template to the site

A simplified representation of this sequence might be:

Conclusions

A well-rounded developer has skills across:

SharePoint	Teams	PowerApps/ Flow	Microsoft Graph	Other
SPFx	SPFx/Teams integration	Solution building	From SPFx	Azure Functions
PnP – provisioning, controls, samples etc.	Templated teams	Use of camera/pen input	From .NET	Azure Dev Ops (boards, pipelines)
Site Designs	App development – tabs, bots etc.	Extending with Azure Functions	Use of SDKs	React
				Git

Action plan

Prioritise - “3 things to learn or get better at”

Build awareness – especially what you should **build**/what you should **use.** Drop in on PnP calls when you can, or watch videos..

Watch roadmaps – understand changes to the platform/functionality

Discuss! – Twitter/events/PnP calls are great places to connect!

www.sharepointnutsandbolts.com

@ChrisO_Brien

Thank you!! ☺

Any questions?

Resources

My blog – Site Designs, Azure Functions etc.

https://sharepointnutsandbolts.com

Postman Graph collections

https://github.com/microsoftgraph/microsoftgraph-postman-collections

Add photos to SharePoint with dataUriToBinary (John Liu)

https://cob-sp.com/2JeJYS4

Reference slides

Call Graph as user in Azure Function

Same set-up:

AAD-secured Function

Use of SPFx (WebApiPermissionRequest, AadHttpClient)

BUT – token passed from SPFx cannot call the Graph

-> Need to exchange for one that can, using UserAssertion

Call Graph as user in Azure Function

```
// obtain passed access token from SPFx..
var userImpersonationAccessToken = req.Headers.Authorization.Parameter;
UserAssertion userAssertion = new UserAssertion(userImpersonationAccessToken);

// use to get new token using AAD app/delegated context..
ClientCredential clientCred = new ClientCredential(clientId, clientSecret);
AuthenticationResult authenticationResult = await authenticationContext.AcquireTokenAsync
(resourceId, clientCred, userAssertion);
string token = authenticationResult.AccessToken;

var outputName = String.Empty;
var responseString = String.Empty;
var phone = String.Empty;

using (var client = new HttpClient())
{
    string requestUrl = $"https://graph.microsoft.com/v1.0/me";

    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, requestUrl);
    request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", token);
    log.Info("About to request URL " + requestUrl);

    HttpResponseMessage response = client.SendAsync(request).Result;
    responseString = response.Content.ReadAsStringAsync().Result;

    var user = JsonConvert.DeserializeObject<AADUser>(responseString);
    phone = user.mobilePhone;
    outputName = user.displayName;
}
```

Site designs = modern site provisioning

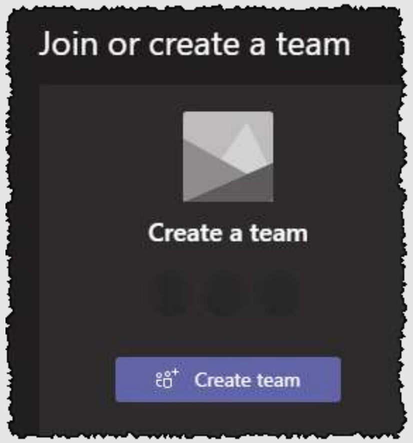
Key benefits:

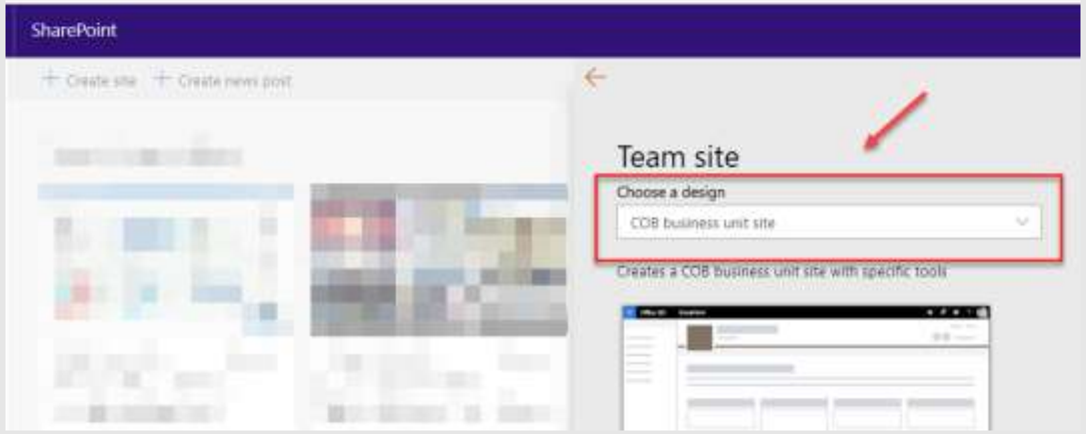
Template sites for Microsoft Teams and Office 365 Groups

and:

Team sites
Communication sites
Hub sites

Integrated into Office 365 UI





Choose a design: COB business unit site

Creates a COB business unit site with specific tools