

Using SAS with Microsoft 365: A Programming Approach

Chris Hemedinger, SAS Institute Inc.

ABSTRACT

In today's cloud-connected world, traditional methods of accessing Excel data from SAS are becoming obsolete. With more content stored in SharePoint Online and OneDrive (hosted in Microsoft 365), it can be challenging to get SAS to read these files and publish new ones to these locations. This paper guides you through the steps of connecting your SAS programs to Microsoft 365, enabling you to read and write files to SharePoint folders, OneDrive folders, and Microsoft Teams. You will learn how to use SAS to connect to Microsoft 365 using the Microsoft Graph APIs. Additionally, we will introduce SAS macros developed to simplify common tasks: listing your files, reading files into SAS, and publishing new files from SAS. By the end of this session, you will have the tools and knowledge to seamlessly integrate SAS with Microsoft 365, enhancing your data management and collaboration capabilities.

INTRODUCTION

If your work environment is like ours here at SAS, you're seeing more of your data and applications move to the cloud. It's not yet a complete replacement for having local files on your desktop machine, but with cloud storage and apps -- like Microsoft OneDrive and SharePoint Online -- we can now access work documents from any browser and any device, including smartphones.

For those of us who use SAS to read and create Microsoft Excel documents, cloud-based files can add an extra wrinkle when we automate the process. It also adds some exciting possibilities! The Microsoft 365 suite offers APIs to discover, fetch, and update our documents using code. This paper shows you how to use SAS programs to reach into your Microsoft OneDrive (or SharePoint Online) cloud to read and update your files.

Note: All of this assumes that you already have a Microsoft 365 account -- perhaps provisioned by your IT support team -- and that you're using it to manage your documents.

USING SAS WITH MICROSOFT 365: AN OVERVIEW

Microsoft 365 uses an OAuth2-style authentication flow to grant access and permissions to third-party apps. If you're accustomed to the simpler style of just user/password authentication (ah, those were the days), OAuth2 can be intimidating.

When we're writing SAS programs to access Microsoft OneDrive or SharePoint, we're actually writing a third-party app. This requires several setup steps, a few of which cannot be automated. Fortunately, these need to be done just once, or at least infrequently. Here's an outline of the steps:

1. Register a new client application at [the Microsoft Azure Portal](#). (You will need to sign in with your Microsoft 365 credentials, which might be your primary organization credentials if you have single-signon.) **You can do this just once in your organization**; multiple users can connect using the same application!
2. Using your browser while you are signed into Microsoft 365, navigate to a special web address to obtain an authorization code for your application.
3. With your authorization code in hand, plug this into a SAS program (PROC HTTP step) to retrieve an OAuth2 access token (and a refresh token).
4. With the access token, you can now use PROC HTTP and the Microsoft 365 APIs to retrieve your OneDrive folders and files, download files, upload files, and replace files.

You'll have to complete Step 1 just once for your application or project. Steps 2 and 3 can be done just once, or at least just occasionally. The access token is valid for a limited time (usually 1 hour), but you can always exchange the refresh token for a new valid access token. This refresh token step can be automated in your program, usually run just once per session. Occasionally that refresh token can be

revoked (and thus made invalid) when certain events occur (such as you changing your account password). When that happens, you'll need to repeat steps 2 and 3 to get a new set of access/refresh tokens.

A SAS MACRO LIBRARY FOR THE BASIC TASKS

For most SAS users, the main goal is to be able to list files and folders in Microsoft 365, download any file to SAS, and upload files from SAS to SharePoint or OneDrive. If that's your situation, then we have some SAS macros that hide much of the complexity.

I've [created this project on GitHub with SAS macros to automate the most common tasks](#). You will still need to complete Step 1 and Step 2 as described in this paper, but after that the macros help with the most common tasks. The macro routines cover these tasks:

- List the OneDrive root folders or SharePoint root folders
- List all files within a folder
- Download a file from a folder
- Upload a file to a folder

I've recorded a [special video that shows how to get started with these macros](#). (The video features SAS Viya Workbench, but you do not need SAS Viya Workbench or any other specialized SAS products to use this technique.)

These macros include logic to manage some of the trickier aspects of the Microsoft Graph API:

- They provide a way to manage your access token and credentials safely, whether you store them in a secure file (SAS 9 or SAS Viya) or in SAS Content folders (SAS Viya only).
- The routines to list folders and files will include **all items** in the collections. The Microsoft Graph APIs return a maximum of 200 items with each call, but these macros detect this situation and will make multiple calls to retrieve the full set.
- When publishing content from SAS to a OneDrive or SharePoint folder, the method for **uploading** files is complex. For large files, the API requires that you create an "upload session" and split the file into chunks. Each chunk is uploaded via an API call and then reassembled on the other side. These SAS macros manage the file splitting, upload session, and iterative upload requests until the file publish is complete. (Learn more about [the file splitting technique in this article](#).)

You can study the code in these macros to implement in your own way, or you can simply use them "as-is" for these common operations.

DOWNLOAD AND INCLUDE MS-GRAPH-MACROS.SAS CODE

The GitHub repository contains a SAS program (named [ms-graph-macros.sas](#)) with all of the macro routines you need for the remaining tasks. Download this file to a local folder and use %INCLUDE to submit in SAS:

```
%let src=<my-local-project>\sas-microsoft-graph-api;  
%include "&src./ms-graph-macros.sas";
```

You can also include directly from GitHub:

```
/* Run just once in your session */  
options dlcreatedir;  
%let repopath=%sysfunc(getoption(WORK))/sas-microsoft-graph-api;  
libname repo "&repopath.";  
data _null_;  
    rc = git_clone(  
        "https://github.com/sascommunities/sas-microsoft-graph-api",
```

```

        "&repoPath."
    );
    put 'Git repo cloned ' rc=;
run;
%include "&repopath./ms-graph-macros.sas";

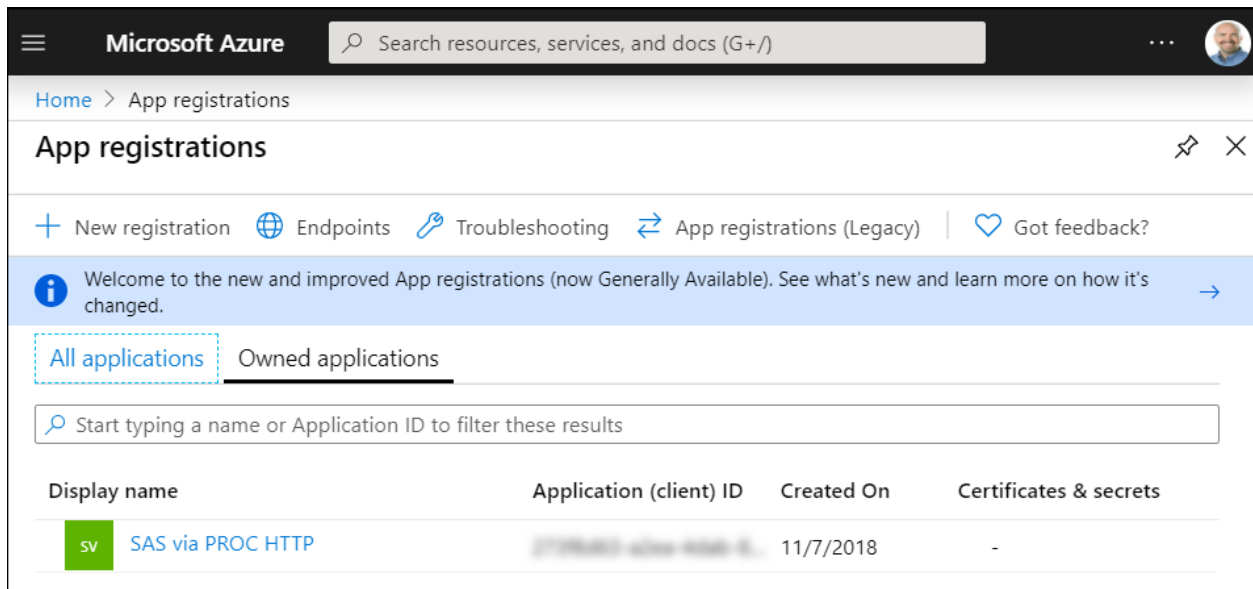
```

STEP 1: REGISTER YOUR APPLICATION

Before you can use the Microsoft Graph API, you need a way to identify your application and grant it the necessary permissions to do its work. At the end of this step, you will have the following information:

- Your **Tenant ID**, which is specific to your organization and would be the same for any application that is registered within your company.
- Your application's **Client ID**, which is unique to your application.
- Optionally, a client "secret" phrase, needed only for applications that do their work without an authenticated user. The examples in this paper assume that you are obtaining your authorizations by logging in to your account to get the authorizations, but it's possible to set up a service account to do the work instead.

To register your application, begin by visiting the Microsoft Azure portal (<https://portal.azure.com>). Sign in with your Microsoft Office 365 credentials. Within the **Azure services** section, select **App registrations**.



The screenshot shows the Microsoft Azure portal interface for App Registrations. At the top, there's a search bar and a user profile. Below the navigation bar, the 'App registrations' section is active. A message banner indicates that App Registrations are now Generally Available. Below the banner, there are tabs for 'All applications' (selected) and 'Owned applications'. A search bar is provided to filter results. A table lists the registered applications with columns: Display name, Application (client) ID, Created On, and Certificates & secrets. One application is listed: 'SAS via PROC HTTP' with a client ID and a creation date of 11/7/2018.

Display name	Application (client) ID	Created On	Certificates & secrets
sv SAS via PROC HTTP	27794523-400a-400a-8000-000000000000	11/7/2018	-

Display 1. App Registrations in the Azure Portal

Click **New Registration** to get started. The "Register an application page" presents you with a form where you can complete the details that define your app. Mainly, you are giving it a name and defining its scope. You will probably want to limit its use to just your organization (your company) unless you are collaborating with colleagues who work elsewhere.

Microsoft Azure Search resources, services, and docs (G+)

Home > App registrations > Register an application

Register an application

*** Name**
The user-facing display name for this application (this can be changed later).

SAS via PROC HTTP ✓

Supported account types
Who can use this application or access this API?

- ☒ Accounts in this organizational directory only (SAS only - Single tenant)
- ☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- ☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

Display 2. Register Your Application

As you register your application, you also need to provide a redirect URL for the authorization flow. In our example, our app is considered "Public client/native (mobile/desktop)." The standard URL to indicate this is:

```
https://login.microsoftonline.com/common/oauth2/nativeclient
```

In the **Redirect URI** section, select this option and specify this URL value.

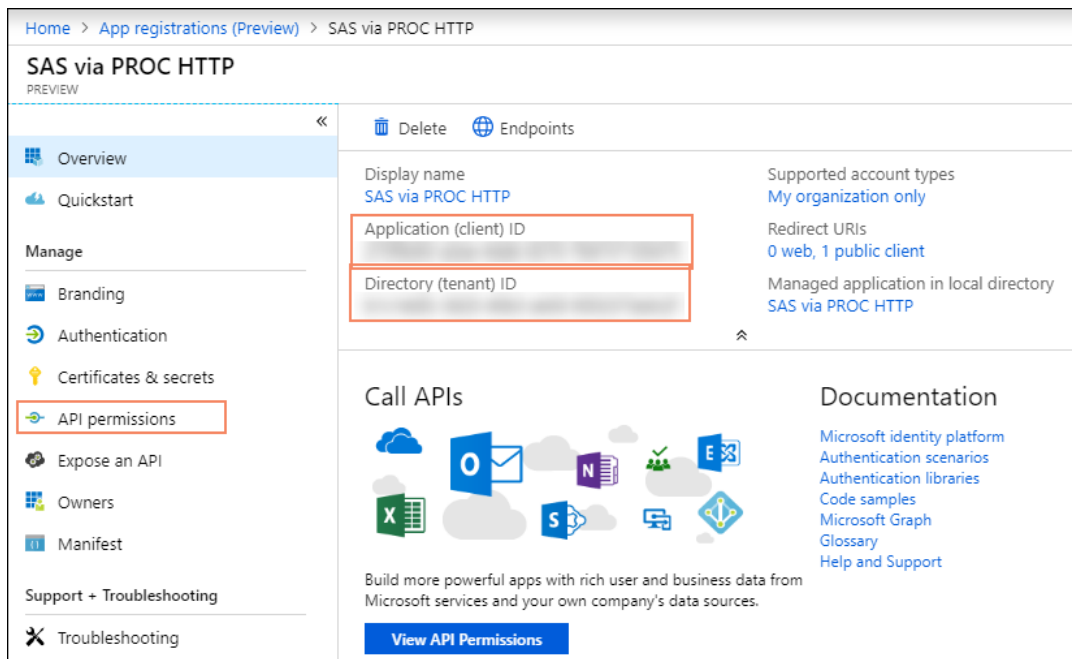
Redirect URI (optional)
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Public client/native (mobile ... ▼ https://login.microsoftonline.com/common/oauth2/nativeclient ✓

Display 3. Redirect URI Selections

Finish by clicking **Register** at the bottom of the page.

When you create an app, you receive a client ID (unique to your app) and tenant ID (unique to your organization). You need these values to obtain your authorization code and tokens later. The application portal provides a sort of control center for all aspects of your app. (Note: I masked out my client ID and tenant ID in this screenshot.)



Display 4. Sample Application, Registered

SPECIFYING YOUR APP PERMISSIONS

Your app needs specific permissions in order to function. In my example, I want my SAS program to read documents from my OneDrive, add new docs, and update existing docs. Here are the permissions that I need:

- **Files.ReadWrite.All:** allows the app to read, create, update and delete all OneDrive files that you can access.
- **User.Read:** allows you to sign into the app with your organizational account and lets the app read your profile.
- **Sites.ReadWrite.All** (if using SharePoint): allows the app to read, create, update, and delete SharePoint Online files for sites that you can access.

To add these permissions to your app, open the **API Permissions** tab in the control center. To be clear, these are not permissions that your app will automatically *have*. These are the permissions that will be *requested* when you "sign in to" the app for the first time, and that you will have to agree to before the app can call these APIs on your behalf.

Home > App registrations > SAS via PROC HTTP - API permissions

SAS via PROC HTTP - API permissions

Refresh

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)

+ Add a permission Grant admin consent for SAS

API / Permissions name	Type	Description	Admin Consent Required	Status
Microsoft Graph (3)				***
Files.ReadWrite.All	Delegated	Have full access to all files user can access	-	***
Sites.ReadWrite.All	Delegated	Edit or delete items in all site collections	-	***
User.Read	Delegated	Sign in and read user profile	-	***

Display 5. Required Permissions for My Sample App

Permission types have their own terminology that is important to understand:

- **Delegated versus Application Permissions:** In our example, we are sticking to **Delegated** permissions, which allow the application to take actions on behalf of the signed-in user and provides access to the user's data. However, some use cases require use of **Application** permissions, which allow the application to take actions without a signed-in user and potentially access data across the system and different users.
- **Admin Consent Required:** Some permissions cannot be delegated or granted without the approval of an administrator. This restriction permits the organization to maintain oversight of the important resources that might be accessed by the application and to prevent unauthorized uses. The Microsoft Azure Portal provides an easy way for you to submit a request to an admin, so you can get the permissions that you need. However, I recommend that you follow up (or better yet, precede this) with a formal request to your IT support staff to state what you need and your business case. In my experience, this helps to expedite the process. A good working relationship with IT is important for any SAS user!

The [documentation for the Microsoft Graph API](#) provides a comprehensive list of the permission names, whether they are Delegated or Application level, and whether Admin Consent is required. This documentation also includes a helpful 4-minute video on the topic.

INITIALIZE THE CONFIG FOLDER

These macros use a file named config.json to reference your client app details, including the app ID and your Azure tenant ID. The file has this format:

```
{
  "tenant_id": "your-azure-tenant",
  "client_id": "your-app-client-id",
  "redirect_uri": "https://login.microsoftonline.com/common/oauth2/nativeclient",
  "resource" : "https://graph.microsoft.com"
}
```

Designate a secure location for this file and for your token.json file (to be created in a later step). The information within these files is sensitive and specific to you and should be protected. See [How to protect your REST API credentials in SAS programs for guidance](#).

If you are using SAS Viya, you can optionally create this folder and file in your SAS Content area that is private to you. For example, create a ".creds" folder within "/Users/*your.account*/My Folder". By default, only your account will be able to read content you place there.

The macro routines need to know where your config.json and token.json file are located. The initConfig macro initializes this.

```
/* This path must contain your config.json, and will also */  
/* be the location of your token.json */  
%initConfig(configPath=/u/yourId/Projects/ms365);
```

If you are using SAS Viya and you would like to store your config and token files in the SAS Content folders (instead of the file system), this is supported with a boolean flag on initConfig. For example, if you store config.json in a folder named .creds within your SAS Content user home, this tells the macro to look in that folder:

```
%initConfig(configPath=/Users/your.account/My Folder/.creds, sascontent=1);
```

Note: This sascontent flag is needed to tell the macro to use the FILENAME FILE SVC method to access the SAS Content area. It requires a different file access method than traditional file systems.

Working with an HTTP proxy

The code in this project uses PROC HTTP without proxy options. If your organization requires a proxy gateway to access the internet, specify the proxy value in the special PROCHTTP_PROXY macro variable:

```
%let PROCHTTP_PROXY=proxyhost.company.com:889;
```

Add this line before calling any other actions.

STEP 2. OBTAIN AN AUTH CODE

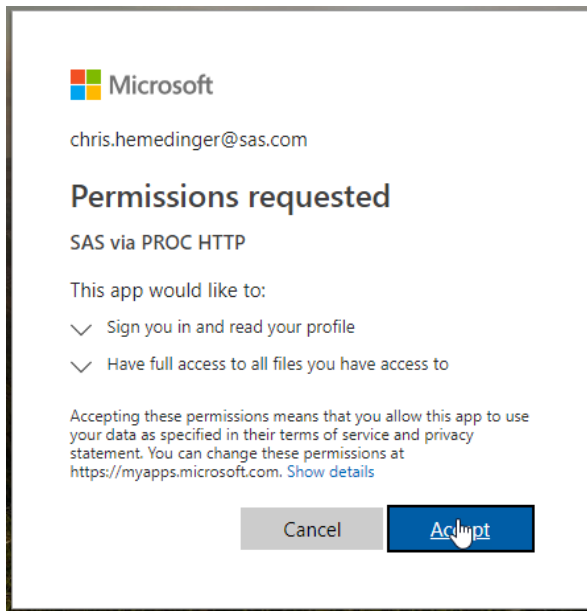
This helper macro will generate the URL you can use to generate an auth code.

```
%generateAuthUrl();
```

The SAS log will contain a URL that you should copy and paste into your browser:

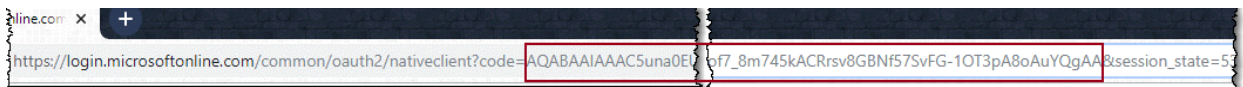
```
-- START -----  
https://login.microsoftonline.com/206db638-6adb-41b9-b20c-  
95d8d04abcbe/oauth2/authorize?client_id=8fb7804a-8dfd-40d8-bf5b-d02c2cbc56  
f3&response_type=code&redirect_uri=https://login.microsoftonline.com/common  
/oauth2/nativeclient&resource=https://graph.microsoft.com  
---END -----
```

Copy and paste the URL (**all on one line, no spaces**) into the address bar of your web browser. When you press Enter, you'll be prompted to grant the required permissions:



Display 6. Consent screen for custom app

After authenticating to Microsoft 365 and granting permissions, the URL address bar will change to include a code= value that you need for the next step. **Copy only the code= value, not any other values that follow in the URL.**



Display 7. URL with auth code as part of the address

STEP 3. GENERATE THE FIRST ACCESS TOKEN

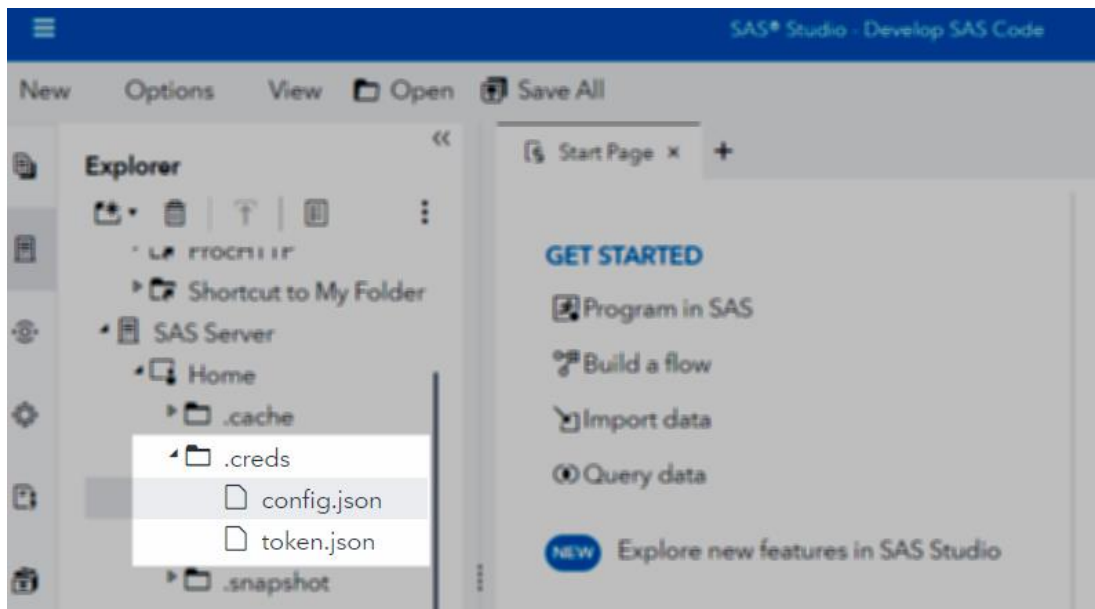
If you just generated your auth code for the first time or needed to get a new one because the old one was revoked or expired, then you need to use the auth code to get an initial access token.

```
/* Note: this code can be quite long -- 700+ characters. */
%let auth_code=PASTE-YOUR-AUTH-CODE-HERE;

/*
  Now that we have an authorization code we can get the access token
  This step will write the token.json file that we can use in our
  production programs.
*/
%get_access_token(&auth_code.);
```

When successful, token.json will be created/updated in the config directory you specified.

You should now have both config.json and token.json in your designated config folder. This screenshot shows an example of these files in a hidden folder named "~/.creds".



Display 8. Example of storing and securing token data

STEP 4. REFRESH THE ACCESS TOKEN AND CONNECT TO MICROSOFT 365

Use the %initSessionMS365 macro routine to exchange the refresh-token stored in token.json for an active non-expired access token.

```
%initSessionMS365;
```

When this is successful, you will see notes similar to these in the SAS log:

```
M365: Reading token info from token.json
M365: Token expires on 26JUL2024:10:04:22
```

The Microsoft Graph API session token is stored in the macro variable &access_token, which is referenced implicitly in the other macro routines in this package.

METHODS TO LIST CONTENT, DOWNLOAD FILES, AND UPLOAD FILES

With a valid access token to connect to Microsoft 365, we can now use various methods to discover and list content within OneDrive and SharePoint (including Teams), and also copy files from these sources into your SAS session, and copy files from SAS into Microsoft 365.

The flow for file discovery is iterative. Each method creates an output data set that can be queried/filtered to a selection of interest, and that will result in an identifier for a folder or file that feeds into the next method.

EXAMPLE: LIST ONEDRIVE CONTENTS

This sequence lists your OneDrive "root" drives (you may have more than one) and then lists the contents of the "Documents" drive.

```
%listMyDrives(out=work.drives);

/* store the ID value for the drive in a macro variable, where "Documents"
is at root */
proc sql noprint;
```

```

select id into: driveId from work.drives where
driveDisplayName="Documents";
quit;

%listFolderItems(driveId=&driveId, folderId=root, out=work.folderItems);

```

Here's some example output:

	name	size	webUrl	lastModifiedDate...	createdDateTime
1	AppData	0	https://sasoffice365-my.sharepoint.com/personal/	/...	2022-02-22T20:23:5...
2	Apps	0	https://sasoffice365-my.sharepoint.com/personal/	/...	2020-11-05T20:16:0...
3	Archive	82546741	https://sasoffice365-my.sharepoint.com/personal/	/...	2018-03-06T12:58:5...
4	Attachments	0	https://sasoffice365-my.sharepoint.com/personal/	/...	2017-01-04T17:46:3...
5	bits&bobs	20348	https://sasoffice365-my.sharepoint.com/personal/	/...	2022-11-22T17:31:1...
6	Communities	7516886...	https://sasoffice365-my.sharepoint.com/personal/	/...	2017-06-28T14:16:4...
7	CustomerPresentations	8478884...	https://sasoffice365-my.sharepoint.com/personal/	/...	2018-08-16T16:54:0...
8	Desktop	18827	https://sasoffice365-my.sharepoint.com/personal/	/...	2022-01-19T20:09:0...
9	Documents	7651279...	https://sasoffice365-my.sharepoint.com/personal/	/...	2018-05-30T15:27:1...
10	EG	107677478	https://sasoffice365-my.sharepoint.com/personal/	/...	2017-07-21T17:40:0...
11	Microsoft Teams Chat...	189682639	https://sasoffice365-my.sharepoint.com/personal/	/...	2018-12-19T19:08:3...
12	Notebooks	0	https://sasoffice365-my.sharepoint.com/personal/	/...	2017-06-29T11:36:2...
13	personal	9939024	https://sasoffice365-my.sharepoint.com/personal/	/...	2018-08-27T19:59:1...
14	Photos	846997761	https://sasoffice365-my.sharepoint.com/personal/	/...	2022-01-18T17:14:2...
15	Pictures	134057754	https://sasoffice365-my.sharepoint.com/personal/	/...	2022-01-19T20:09:0...

Display 9. Sample listing of folders and files in OneDrive

EXAMPLE: LIST SHAREPOINT FOLDERS FILES

The %listSiteLibraries macro helps you to explore a SharePoint site or Teams folder. Here's an example code flow:

```

/* this macro fetches the root IDs for document libraries in your site */
%listSiteLibraries(
  siteHost=mysite.sharepoint.com,
  sitePath=/sites/Department,
  out=libraries);

/* store the ID value for the library in a macro variable, where
"Documents" is at root */
proc sql noprint;
  select id into: libraryId from libraries where name="Documents";
quit;

/* LIST TOP LEVEL FOLDERS/FILES */

/* special macro to pull ALL items from root folder */
%listFolderItems(driveId=&libraryId., folderId=root, out=work.paths);

```

And here's an example output from SAS Studio:

LIBRARIES Library: WORK	Enter expression			
		_odata_context	createdDateTime	description
PATHS Library: WORK	1	https://graph.microsoft.com/v1.0/\$met...	2017-07-12T07:35:37Z	

Filter	<< PATHS Table rows: 9 Columns: 12 of 12 Rows 1 to 9			
LIBRARIES Library: WORK	Enter expression			
		name	size	webUrl
	1			
	2	Blogs	3934472	sharepoint.com/sit...
	3	General	32718671...	sharepoint.com/sit...
	4	Responsive	806449	sharepoint.com/sit...
	5	Spam	8776078	sharepoint.com/sit...
	6	Experience 2030.pdf	1714735	sharepoint.com/sit...
	7	header.png	187750	sharepoint.com/sit...
	8	How to Use Influencers Within Commu...	161462	sharepoint.com/sit...
	9	PitneyBowesAcceleratingEngagement...	292612	sharepoint.com/sit...

Display 10. Sample listing of folders and files in SharePoint Online (or Teams)

With these folders from the root path, you can select one to explore further, drilling into folders as deep as needed.

```
/* Find the ID of the folder I want */
proc sql noprint;
  select id into: folder from paths
    where name="General";
quit;

/* Pull ALL items from a folder */
%listFolderItems(driveId=&libraryId., folderId=&folder.,
  out=work.folderItems);
```

Here's an example of the output with more detail at one level lower:

	Enter expression			
	name	size	webUrl	lastModifiedDateTime
201	SAS communities thread (2)...	118272	.sharepoint.com/sit...	2019-04-19T14:59:52Z
202	SAS communities thread.msg	117248	.sharepoint.com/sit...	2019-04-19T14:59:52Z
203	if	3715768	.sharepoint.com/sit...	2020-09-25T16:46:01Z
204	.png	260319	.sharepoint.com/sit...	2019-03-12T13:24:17Z
205	.png	11448	.sharepoint.com/sit...	2022-08-12T19:26:46Z
206	F International Festival Ma...	98562	.sharepoint.com/sit...	2021-03-26T15:11:48Z
207	activity March 2021.png	157296	.sharepoint.com/sit...	2021-03-26T15:12:05Z
208	.png	4441515	.sharepoint.com/sit...	2017-10-04T12:36:17Z
209	Watch Phrases.pdf	433434	.sharepoint.com/sit...	2021-05-14T11:49:31Z
210	Enterprise Guide Tips and Tr...	50120	.sharepoint.com/sit...	2020-08-12T19:14:01Z
211	ODA.mp4	301405364	.sharepoint.com/sit...	2021-03-16T14:45:29Z
212	GA Edition.mp4	15970801...	.sharepoint.com/sit...	2021-06-03T17:01:55Z
213	ModelOps.mp4	335708334	.sharepoint.com/sit...	2021-08-10T19:40:49Z
214	I, Graphically Speaking blo...	290869160	.sharepoint.com/sit...	2021-10-12T20:59:50Z

Display 11. Example listing of lower-level folder in SharePoint Online

EXAMPLE: DOWNLOAD A FILE FROM SHAREPOINT TO YOUR SAS SESSION

After using the list-style macros to gather the unique IDs of folders and files, we can use the Microsoft Graph APIs to “download” a selected file into the SAS session. This creates a local copy of the file that SAS can read using code, such as with DATA step or PROC IMPORT. A classic use case is to download an Excel file from SharePoint, then read its content into SAS by using PROC IMPORT. This example code illustrates these two operations:

```
/*
  With a valid source folderId and knowledge of the items in this folder,
  we can download any file of interest.

  This example downloads a file named "ScoreCard2022.xlsx" from a known
  folder on SharePoint (obtained in previous steps) and places it in a
  file location on the SAS session.
*/
%downloadFile(driveId=&driveId.,
  folderId=&folder.,
  sourceFilename=ScoreCard2022.xlsx,
  destinationPath=/tmp);

/* Downloaded an Excel file into SAS? Now we can PROC IMPORT if we want */
proc import file="/tmp/ScoreCard2022.xlsx"
  out=xldata
  dbms=xlsx replace;
run;
```

EXAMPLE: UPLOAD A FILE FROM SAS TO SHAREPOINT

Since SAS users often produce results that they need to share with the organization, there is a need to be able to upload one or more result files from SAS to SharePoint Online or a Teams folder. The Microsoft

Graph API supports uploads through its `createUploadSession` API. The package of macro routines shared here simplify the process with the `uploadFile` macro. Here is an example use:

```
/* Create a sample file to upload */
%let targetFile=iris.xlsx;
filename tosave "%sysfunc(getoption(WORK))&targetFile.";
ods excel(id=upload) file=tosave;
proc print data=sashelp.iris;
run;
ods excel(id=upload) close;

/* Upload to the "General" folder, the folder ID from previous step */
%uploadFile(driveId=&libraryId.,
  folderId=&folder.,
  sourcePath=%sysfunc(getoption(WORK)),
  sourceFilename=&targetFile.);
```

When you upload a file to a folder and file name that already exists in SharePoint, Microsoft 365 creates a new version of the file that is visible in SharePoint's Version History feature.

The macros for listing files and uploading files do special additional work to overcome some of the designed limitations of the APIs:

- The "list" methods (such as `listFolderItems`) have special handling to use multiple API calls to gather a complete list of results. The Microsoft Graph API methods return a max of 200 items in a response with an indicator if there are more. These SAS macros will follow through and gather the complete list.
- The `uploadFile` method uses the special "large file upload" handling to create an upload session that can accommodate files larger than the 4MB size that is the default size limit.

USE ANY MICROSOFT GRAPH API ENDPOINT

With the authenticated session established, you can use PROC HTTP to execute any API endpoint that your app permissions allow. For example, with `User.Read` (most apps have this), you can download your own account profile photo:

```
filename img "c:/temp/profile.jpg";
proc http url="%msgraphApiBase./me/photo/$value"
  method='GET'
  oauth_bearer="%access_token"
  out = img;
run;
```

The `msgraphApiBase` and `access_token` macro variables are set during `%initSessionMS365` macro routine.

This example shows how to retrieve the SharePoint Lists that are defined at the site root. Note that the `/sites/root/lists` endpoint requires `Sites.Read.All` permission.

```
filename resp temp;
proc http url="%msgraphApiBase./sites/root/lists"
  method='GET'
  oauth_bearer="%access_token"
  out = resp;
run;

libname lists JSON fileref=resp;
proc sql;
  create table work.list_names as
```

```
select t1.name,  
       t1.displayname,  
       t1.weburl,  
       t2.template  
from lists.value t1  
     inner join lists.value_list t2 on  
       (t1.ordinal_value = t2.ordinal_list);  
quit;
```

All APIs are documented in the [Microsoft Graph API reference](#).

CONCLUSION

Integrating SAS with Microsoft 365, including OneDrive and SharePoint, presents exciting possibilities for data management and collaboration. By leveraging SAS macros and the Microsoft Graph API, users can seamlessly access, update, and share their files across cloud platforms, enhancing productivity and flexibility. This integration not only simplifies the automation of routine tasks but also ensures secure and efficient handling of large files. As organizations continue to embrace cloud-based solutions, mastering these techniques will be crucial for maximizing the potential of both SAS and Microsoft 365 in a dynamic work environment.

REFERENCES

Hemedinger, Chris. "SAS Macros for Microsoft Graph API", *GitHub.com*. Available at <https://github.com/sascommunities/sas-microsoft-graph-api>

Hemedinger, Chris. "Demo: SAS Viya Workbench and SAS code to access Microsoft 365", *communities.sas.com*. Available at <https://communities.sas.com/t5/SAS-Viya-Workbench-Getting/Demo-SAS-Viya-Workbench-and-SAS-code-to-access-Microsoft-365/ta-p/952476>

Hemedinger, Chris. "Using SAS with Microsoft 365 (OneDrive, Teams, and SharePoint)", *blogs.sas.com*. Available at <https://blogs.sas.com/content/sasdummy/sas-programming-office-365-onedrive/>

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Chris Hemedinger
SAS Institute Inc.
Chris.Hemedinger@sas.com
<https://blogs.sas.com/sasdummy>

Any brand and product names are trademarks of their respective companies.