Background

For programmatically controlling OneNote from the server side, it is necessary to communicate through the Microsoft Graph API for OneNote and configure the required permissions through Microsoft Azure to allow access to modify the OneNote account without the need for repetitive human-intervened authentication.

Setting up the Azure App

1. Sign in to Microsoft Azure with your Office 365 account. We won't be using any cloud computing services, so you won't need to purchase any, but you will need to sign up for an Azure account. You can choose the Pay as you go feature but it will ask to verify your credit card details.
2. Go to Azure Active Directory -> App Registrations and create a new app. Choose app type as Web App/API, give an App name and set the Reply URL to https://login.live.com/oauth20_desktop.srf
3. Go to Keys and create a new Client Secret. Copy the key, you can't view it anymore after this. Also copy the Application ID
4. Go to Required Permissions and add the following Delegated Permissions:
   a. Microsoft Graph –
      i. Read and write all OneNote notebooks that user can access
      ii. Read and write user OneNote notebooks
      iii. Create user OneNote notebooks
   b. OneNote –
      All delegated permissions
   c. Windows Azure Active Directory –
      i. Sign in and read User Profile
5. Press the Grant Permissions button to ensure the app has received all the permissions
6. Go back to the Azure Active Directory tab and choose Properties. Copy the Directory ID, this is required later to identify this OneNote account.
7. In the Python programs, the above information gets mapped as follows:
   a. Application ID = _clientID_
   b. Directory ID = _tenantID_
   c. Client Secret = _clientSecret_
   d. Email = Office 365 email ID

Setting up the Environment Dependencies

Python – Not required if Pre-compiled

1. Python (v3.6 or higher)
2. Requests: HTTP for humans (v2.19 or higher)

LabVIEW 2010+

NGROK

1. ngrok (needs to be a registered user, free service) – used for tunnelling server

Notebook Structure

The code assumes that a Notebook for a machine has ALREADY been created and the name of the Notebook must be supplied while setting up the python code. Subsequently, all section groups, sections and notes are automatically generated in chronological order when creating a page, following the existing directory structure for OneNote 2010. In OneNote Online, a single notebook appears as one single file.

Setting up the VIs

There are five VIs in the LLB for Creating a Page, Adding an Image, Adding a Header and Adding text and Adding a Table. Each comes with its own associated Python executable.

Careful consideration must be given to configure all the file & folder paths and the Notebook name in the VI. The tempImg should get saved in the same folder as the LLB/AddImage VI to allow web tunneling. Also, the necessary username paths, IDs and tokens earlier mentioned should be updated for all Python modules and executables recompiled.

Setting up the servers

A simple server has been written in Python that can handle static content stored in its local directory. This has been compiled and can be run by executing Python_server.exe. The default port launched is 7800, but any port can be passed as a command line argument. Server can be accessed at localhost:7800. The NGROK is a tool that lets you create a web tunnel to a local server. The supplied script configures NGROK with account info and then runs it to access the tunnelled port at 7800. Both the server and the NGROK script must initially be run once before using the VIs for first time, to provide Firewall access to the applications. The applications can be closed by Ctrl+C after granting Firewall access and this completes their initialisation

Data Flow in Code backend

Workflow between Python & LabVIEW

1. Menu action in LabVIEW
2. Launch a Sub-VI
3. Decide inputs to Sub-VI (Map from existing code)
4. Use SystemExec VI to launch executable
5. Pass data through interchange files - CSV, PNG, JPG
6. Read data from file to Python modules
7. Python code communicates with the Microsoft Graph API

Background server for image uploading

1. Check if a server is running,
2. Start a Python server if not running, tunnel with NGROK
3. Use Python to access NGROK REST API and return the URL
4. Use this URL for adding images to pages in OneNote
5. Kill both server and ngrok executables once image has been added

Description of the VIs in the LLB:

VI	Inputs	Outputs	Description
Create Page	• Notebook name • Path to Python executable • Error in	Error Out	Creates a new page in chronological fashion in an existing notebook. Will create section group, section & note if not present.
Header	• Notebook name • Path to Python executable • Font size • Error in	Error Out	Creates in a new heading in today's page.
Image	• Notebook name • Path to Python executable • Port Name • NGROK Auth-token • Error in	Error out	Launches a local server, tunnels to web, reads the **PNG** image, writes a local copy of the **PNG**, and sends the image to be appended to existing day's page. It then kills the server and tunnel applications.
Text	• Notebook name • Path to Python executable • Font size • Input Text • Error in	Error out	Takes the input text and appends to todays page
Table	• Notebook Name • Path to Python exe • Table • Column Widths array • Error in	Error out	Creates CSV from table and from width values and adds it to today's page

Creating compiled executables from Python scripts

Update all credential variables before compiling into executables

For compiling the scripts:

1. Make sure to have Python 3.6+ on the right archi
2. Install requests using -> pip install requests
3. Install pywin32 from https://github.com/mhammond/pywin32/releases
4. Install pyinstaller using -> pip install pyinstaller
5. Go to the folder containing the python scripts and run the following
6. pyinstaller --onefile <your_script_name>.py
7. You can find the exe under the dist folder