OneNote to Rule Them All

Eric Selje
Salty Dog Solutions, LLC
www.SaltyDogLLC.com
Madison, WI USA
Voice:608-213-9567
Twitter: @EricSelje
Email:Eric@SaltyDogLLC.com

You may know OneNote as a pretty convenient place to keep information. But with its ability to store text, graphics, audio, video, hyperlinks, along with its sharing and replication features, OneNote can be used in ways you may not have thought of. And with the recent addition of a robust API OneNote can now be accessed from your custom apps as easily as any database.

Introduction

OneNote has been described as the best tool from Microsoft that you're probably not using. Now we Visual FoxPro developers might try to argue that, but since OneNote is being actively developed and supported I wouldn't argue. It really is an amazing program.

I'm not going to write a complete tutorial on OneNote in this whitepaper. There are plenty of resources out there already that will get you started on OneNote. If you don't have any idea what it is go ahead and take some time to read up on it. I'll wait….

…ok welcome back! Now that you know OneNote is Microsoft's app notetaking, I'm going to take you further because OneNote can do so much more. Used properly, it can keep your entire life organized, replace some of your other apps. Beyond that, the underlying sharing and replication technology allows OneNote to be used like a database, which allows for some interesting possibilities. Furthermore OneNote has an API which allows developers to do CRUD (Create, Read, Update, and Deleted) "records" from the OneNote notebook. Let's get started!

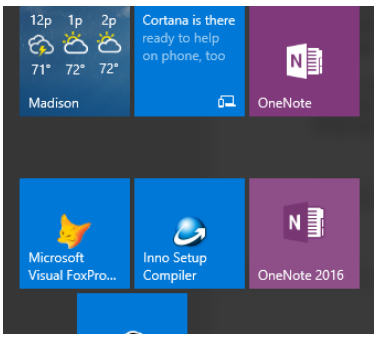
My Favorite OneNote Features

It's Free

Let's start right out by letting you know that OneNote is totally free. No subscription costs, no server costs. You can download it[i] without giving your credit card information. Free.

It's Available *Everywhere*

My favorite feature of OneNote is that the information I store in my Notebooks is available everywhere. If I'm at my desk in the office, I use the OneNote 2016 desktop version, which is the full-featured "desktop" version. There's also a Windows Store App that's useful when I've got my Surface Pro at a meeting, but mostly when I'm on my PC I stick to the desktop version because it has more features.



There's an iOS version so I can access my notes from my iPhone and my iPad. This came in handy just the other day when I was fishing in northern Wisconsin, far beyond the reach of any phone signal, and a warden wanted to see my fishing license. I had left my wallet back at the campsite, but was able to pull out my phone, open OneNote, and show him a photo that I had taken of my license. I was off the hook, unlike the northern pikes I caught that day!

There are also native Mac and Android versions, which I haven't used. Lastly, there's a OneNote web app that will get you to your notebooks from any device with a web browser, as long as you have internet access.

Wait, wait... back up. How was I able to show the warden my OneNote document if I was beyond a cellphone signal? Is the data actually sync'd to my device? If so, how did it get from my PC to my phone?

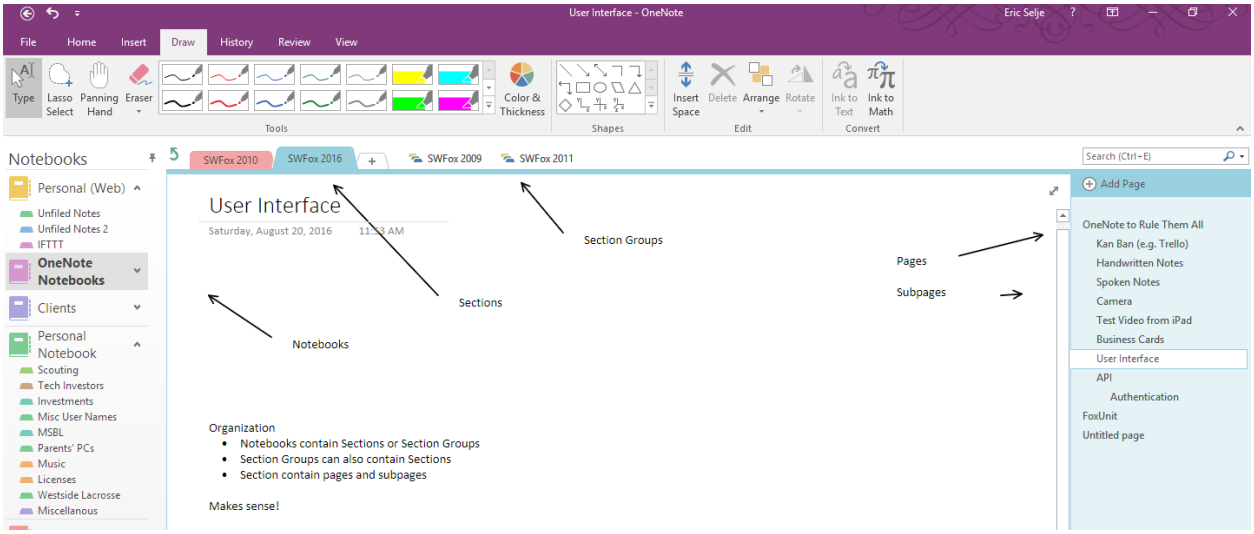
This really gets to the heart of the way OneNote stores and replicates files. You don't have to, but if you choose to store your OneNote notebooks in OneDrive, Microsoft's cloud storage service, the "master" copy of your notebooks are actually stored in OneDrive, somewhere on Microsoft's servers. This means your notebooks are always backed up, secured, and replicated in multiple Microsoft datacenters. It's this online master file that also then syncs to all of your other devices, and it's what you access when you use the web version. OneNote automatically handles the syncing and conflict resolution. You don't usually have to do anything unless there's a conflict it can't figure out what to do with, such as if you and someone else modify the same page at the same time. On your local machine, your notebooks are cached under your %UserProfile%\appdata folder, but you really don't ever have to consider that.

Storing your notebooks in OneDrive does require a "Microsoft Account". I know, *another* account, but if you've got an Outlook or Hotmail account, ever downloaded an app from the Microsoft Store, used Visual Studio for the last few years, or used OneDrive (f/k/a SkyDrive) previously you already have a Microsoft Account. It's the cost of doing business with Microsoft these days, and it's the same with Apple IDs, Google Accounts, and Amazon.

Enterprises may choose to store notebooks in SharePoint rather than OneDrive.

The User Interface

To me, OneNote's user interface makes complete sense.



Sharing

You can collaborate on OneNote notebooks with anyone you choose, but the sharing feature only works at the notebook level, which means you cannot just share one section or one particular page. This is in my opinion the biggest shortcoming of OneNote, but as long as you're aware of it you can structure your information in a way that allows you to share what you want and only what you want.

So Many Ways to Input

Typing

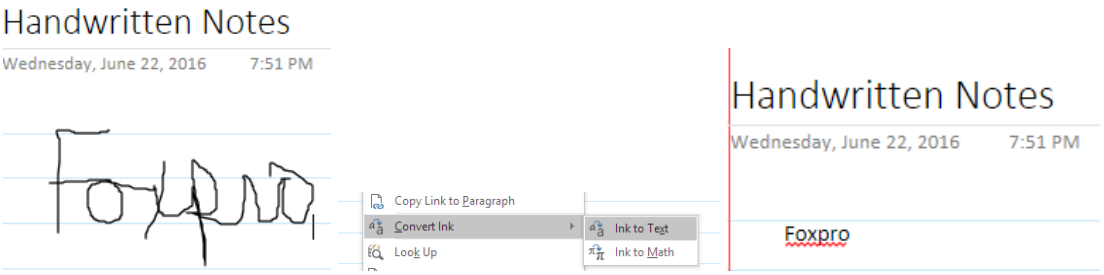
You can still use an old fashioned keyboard to get the information from your head into OneNote, and OneNote has a *ton* of shortcuts to make this method very efficient. Here is a pretty comprehensive list, many of which you'll be familiar with if you use Notes or even FoxPro's editor. A couple of my favorite shortcuts when I'm taking notes are:

Ctrl+. To start/stop bulleted lists
Ctrl+/ To start/stop numbered lists

It's easy to remember the numbered list shortcut because the slash is right next to the period on the keyboard.

Handwriting

If you're using OneNote on a device with pen input, you can draw your notes right into your notebook just like you're back in high school. Ctrl+Shift+R will even give your pages a very familiar notebook look and feel. What's surprising about OneNote is how well it can convert even sloppy handwriting to accurate text using the "Ink to Text" feature.



Voice Recording

But really who hand writes, or even types, anything? This is 2016 (or insert whichever year you're reading this year) and Siri, Alexa, and Cortana now do our bidding.

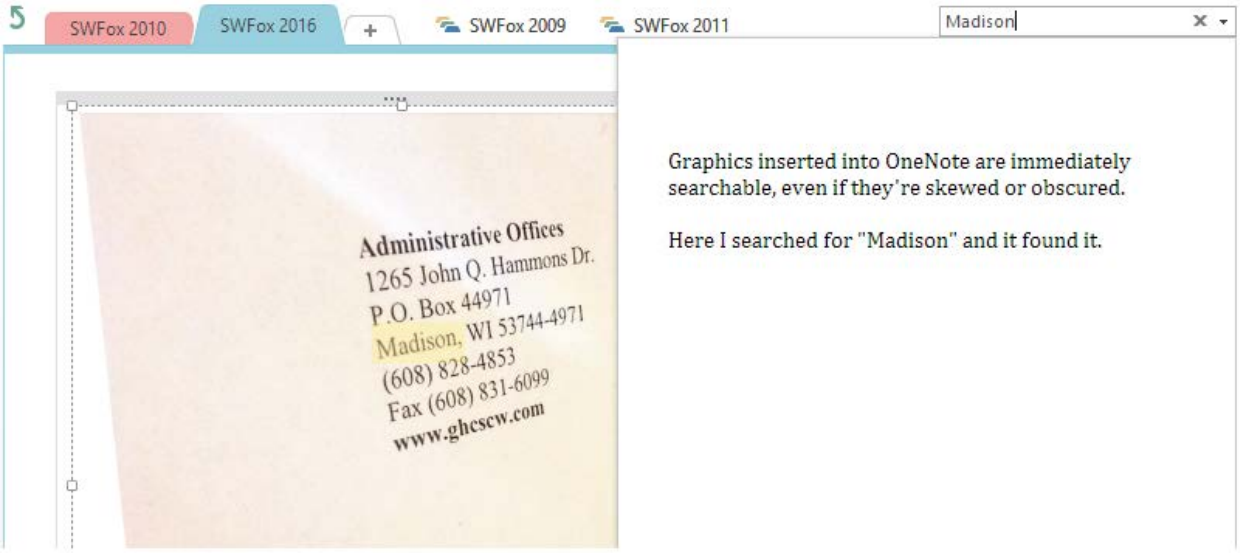
OneNote is totally with the times here. You have your choice of either recording a voice note and leaving it as a recording, or having it automatically transcribed to text. The killer feature is that as type notes while OneNote is audio recording, OneNote will actually synchronize your notes to your recording and jump directly to those notes while the

recording is playing back. It also works the opposite way, so if I click on my notes it will cue the audio up to exact place it was when I jotted that note. I wish I had this back in college!

This feature doesn't work on the iPad or Tablet versions of OneNote, which is a bit disappointing since it seems like that's the natural place for it.

Camera

You can insert snapshots from your device's camera straight into OneNote as well. Don't think selfies here though, consider taking snapshots of business cards with your phone's version of OneNote, or using it as a "copy machine" for those paper documents you want to archive. OneNote has the ability to make any text inside the photograph searchable, so we're beginning to see how you could create a "database" just by photographing your pile of business cards.

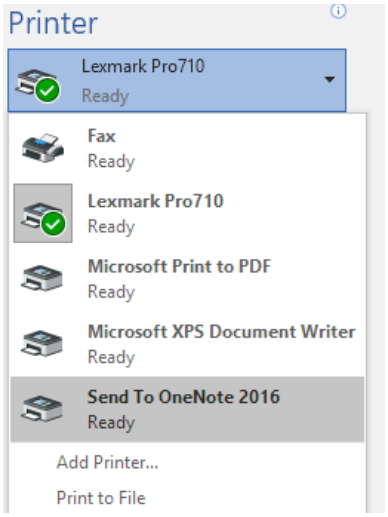


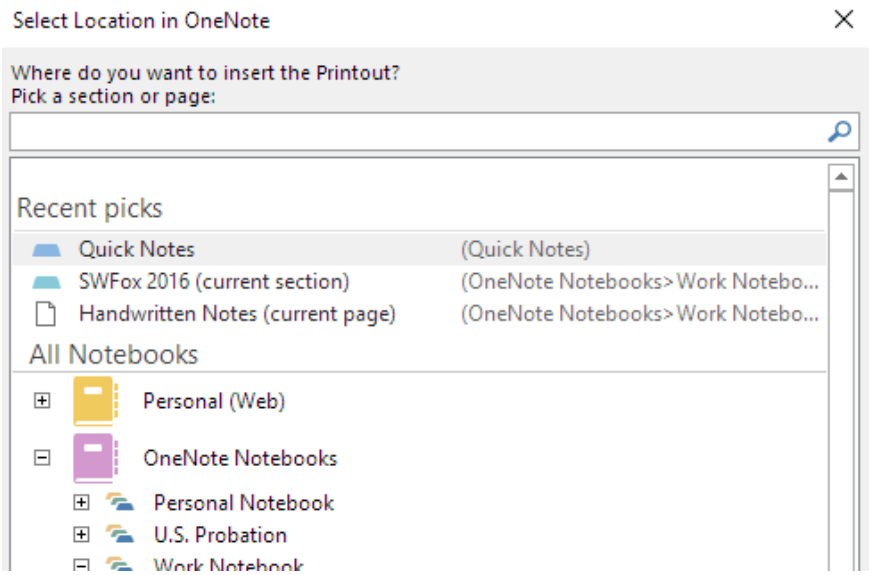
Microsoft Office Lens

Realizing the combination of your device's camera and OneNote, Microsoft created an app specifically for mobile scanning: the oddly named "Office Lens". Office Lens is better than just uploading from your camera because recognizes that you're holding something that you want to scan, and don't necessarily want the entire frame. It will automatically crop out beyond the edges of the thing you're taking a picture of (a business card, for example). It can then automatically parse the business card and add the information into your Outlook Contacts, or place it into any section of any of your OneNote notebooks. Very cool utility, though I wish it were named something like "Office Pocket Scanner".

"Send to OneNote" Print Driver

When you install the Desktop version of OneNote you also install a Windows Print Driver, so anything can be "printed" directly into OneNote. You'll be presented with a dialog asking which page of which section of which notebook to insert the printout.





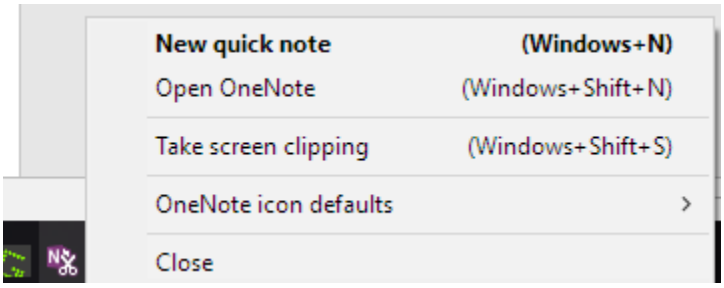
Email Directly

OneNote allows you to configure your email so that you can send directly from your email address to *me@onenote.com*. This is really handy for forwarding important information from your inbox. You select a default section when you configure this option, but you can override the section by changing the subject line of your email to *@SectionName*.

IFTT / Zapier

Along those lines you can configure the web "recipe" utilities IFTTT (If This Then That) or Zapier to automatically add to your Notebooks. This can be extremely powerful. For example, you can create a recipe that will automatically create a page in OneNote for an upcoming event in your Google Calendar, so you can get started on taking notes for that event automatically. On the sillier side, I use a recipe that will automatically add the latest XKCD comic to a section of my personal OneNote.

The OneNote Clipper



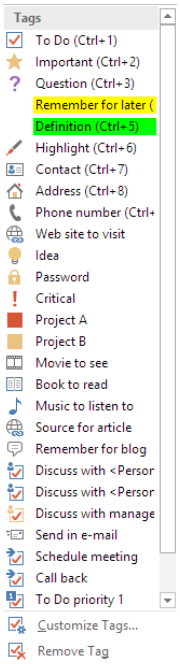
Also installed with the desktop version is the "Clipper", which gives you Windows shortcuts for taking notes and opening OneNote, but also has the very handy "screen clipping" tool that I've used for all of the screenshots in this whitepaper. It's not as powerful as SnagIt, but it's a lot more useful than the other utility that comes with Windows, the "Snipping Tool" because there's a shortcut key to initiate the screenshot. Once you grab the section you want you can either send it to the clipboard or directly into a OneNote page.

Embed Pretty Much Anything

Lastly you can drag and drop any file onto a page in OneNote and you'll get the option to either embed that entire file right there in your notebook, or insert a printout of the file.

In summary, OneNote makes it extremely easy to input information by providing a myriad of utilities and extra functionality.

Tags and Customized Tags



Once you've got content into OneNote, you can "Tag" it using any of the provided tags, or you can customize the tags to create your own. You might create a tag for each of your clients, for example, because their information is spread across multiple pages of multiple notebooks. Later you can search by that tag and see all of the pages that include that tag.

The top 9 tags in your list are automatically assigned the hotkey Ctrl+#, so if you find you're using a tag that's not hotkeyed, customize your tags and drag that tag closer to the top of the list. While you're customizing your tags you can also change the little icon as with each tag, though you do have to pick from the limited palette that OneNote offers.

Much like gmail, each item can be tagged multiple times as well.

Searching

OneNote's searching capabilities are amazing. It instantaneously displays the results of your search, regardless if the item your searching for was typed, handwritten, or embedded in a graphic. You can refine your searches to specific pages up to entire notebooks, and sort them by date modified, section, or page title.

Besides searching for text you can also search for all items that are tagged. This is extremely handy for finding all of your to-do items, all of the questions that are outstanding, or all of the people you need to call back.

You can also search for pages based on their metadata. For example you can see all the pages that were modified by a certain collaborator, or all of the pages that were changed in the previous week.

The OneNote API

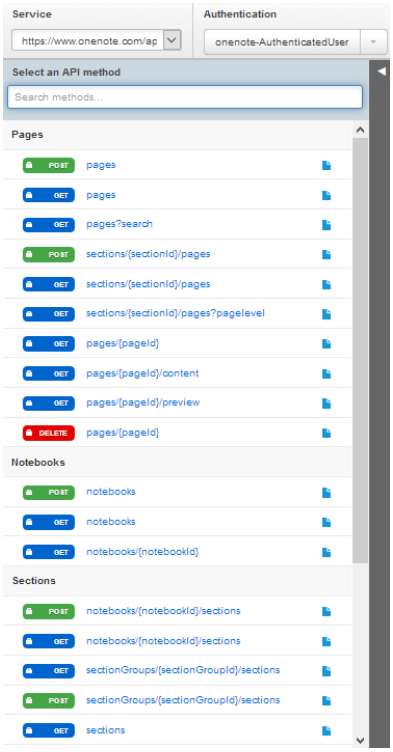
OneNote uses a REST API to allow external applications to manipulate notebooks, sections, section groups, and individual pages. Currently OneNote's API stands alone from any other Microsoft APIs, but Microsoft is working hard to create what it calls the Microsoft Graph API to have a single REST endpoint for all of its applications.

The API is laid out pretty intuitively, with calls to create, update, and delete. The content of pages is a subset of HTML, so it will seem very natural for us developers to create. You cannot have any javascript or CSS in the pages, or HTML forms. You can include <img> tags, but the image will be rendered into the page and not continue to grab the reference from somewhere else on the internet. For complete information about what HTML is allowed in OneNote pages, check http://dev.onenote.com/docs#/introduction/html-tag-support-for-pages.

APIGEE Console

The best place to learn the OneNote API is to use a customized console that Apigee and Microsoft created at https://apigee.com/onenote/embed/console/onenote/. Not every REST call is exposed from this console but you'll really get the idea of what you can do with what calls are available.

Start by authenticating yourself with your Microsoft Account, and note how easy this is to do in the Apigee console. When we develop our own apps we'll also need to authenticate ourselves, and and it's not quite as easy to accomplish as this. More on that in the next section.



Once you're authenticated, look along the right side of the console to see many of the REST calls that are available, categorized by the level of the notebook. This list is not actual complete, probably because Microsoft continues to embellish the API. You can see the up-to-date list of API calls at https://dev.onenote.com/docs. Hover over a call to get a better description of what that call will do.

To make an API call, you send either a GET, POST, DELE, or PATCH http request to the API endpoint at

https://www.onenote.com/api/v1.0/me/notes/{target}/{parameters}

where {target} is either Notebooks, Sections, SectionGroups, or Pages. The parameters will often include an ID for the target you want to get.

The Apigee console automatically sends the "Access Token" that was created in the header of the request.

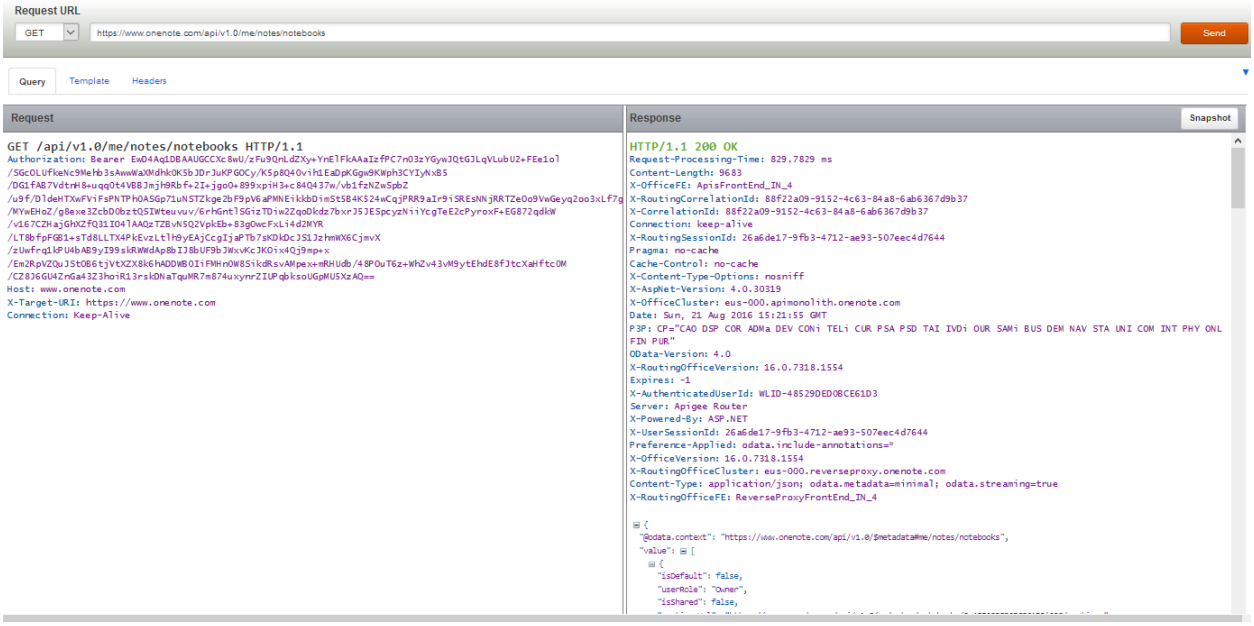
Let's look at some examples using the Apigee console, starting at the highest level and drilling our way down.

Example 1: Show Me My Notebooks

Perusing the list of API calls arranged so logically it's easy to find the call that will create a new section in an existing notebook

GET https://www.onenote.com/api/v1.0/me/notes/notebooks

There are no parameters to send here, and the results will simply enumerate all of the notebooks associated with your Microsoft account.



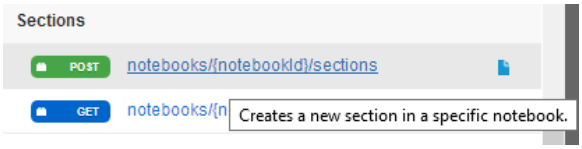
The console shows you the complete API request in the left panel, and the results in the right panel. You can see we got a 200 response code, which is a good thing. Here are the general categories of http response codes[ii]:

Code Category	Meaning
100	"I'm not done yet"
200	"Success!"
300	"I dished off your request to someone else"
400	"Your Fail!"
500	"My Fail!"

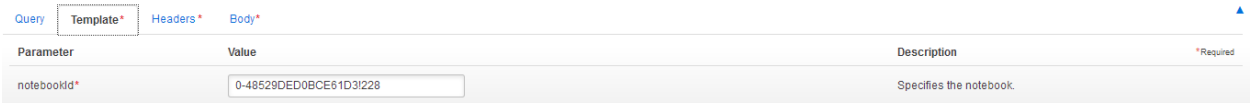
Taking a closer look at the JSON that came back, you see we got an array of notebooks, with each row containing properties and metadata pertaining to the notebook such as whether we are an owner or merely a contributor to that notebook, if it's shared, where it lives, what its name is, and most importantly, its ID which we'll use in subsequent calls when we drill down into the notebook.


```
[
    {
      "isDefault": false,
      "userRole": "Owner",
      "isShared": false,
      "sectionsUrl":
"https://www.onenote.com/api/v1.0/me/notes/notebooks/0-
48529DED0BCE61D3!230/sections",
      "sectionGroupsUrl":
"https://www.onenote.com/api/v1.0/me/notes/notebooks/0-
48529DED0BCE61D3!230/sectionGroups",
      "links": {
        "oneNoteClientUrl": {
          "href":
"onenote:https://d.docs.live.net/48529ded0bce61d3/Documents/Clients"
        },
        "oneNoteWebUrl": {
          "href":
"https://onedrive.live.com/redir.aspx?cid=48529ded0bce61d3&page=edit&resi
d=48529DED0BCE61D3!230"
        }
      },
      "name": "Clients",
      "createdBy": "Eric Selje",
      "lastModifiedBy": "Eric Selje",
      "lastModifiedTime": "2016-08-20T19:59:14.117Z",
      "createdTime": "2012-03-19T16:35:47.94Z",
      "id": "0-48529DED0BCE61D3!230",
      "self": "https://www.onenote.com/api/v1.0/me/notes/notebooks/0-
48529DED0BCE61D3!230"
    },
    {
      "isDefault": false,
      "userRole": "Contributor",
      "isShared": true,
      "sectionsUrl":
"https://www.onenote.com/api/v1.0/me/notes/notebooks/0-
F95A2753E9BBD88F!1222/sections",
      "sectionGroupsUrl":
"https://www.onenote.com/api/v1.0/me/notes/notebooks/0-
F95A2753E9BBD88F!1222/sectionGroups",
      "links": {
        "oneNoteClientUrl": {
          "href":
"onenote:https://d.docs.live.net/f95a2753e9bbd88f/Documents/DSS%20Report%
20Ideas"
        },
        "oneNoteWebUrl": {
```

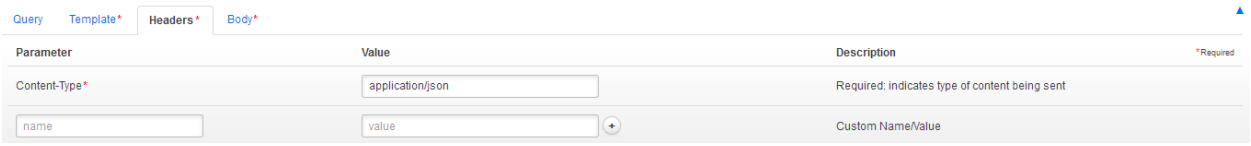
Example 2: Add a Section to an Existing Notebook



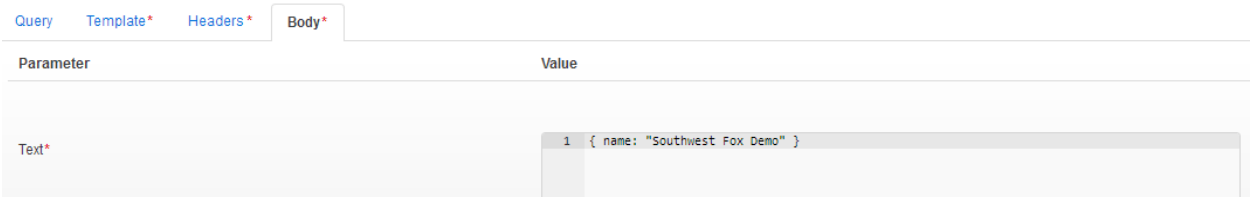
Again, looking through the list of possible API calls arranged so logically it's easy to find the call that will create a new section in an existing notebook. This one requires me to enter the ID of the notebook, which I got from the call in Example 1. In the console I could replace the {notebookID} in the URL with the ID, but it's simpler to user the Template tab and fill it in there:



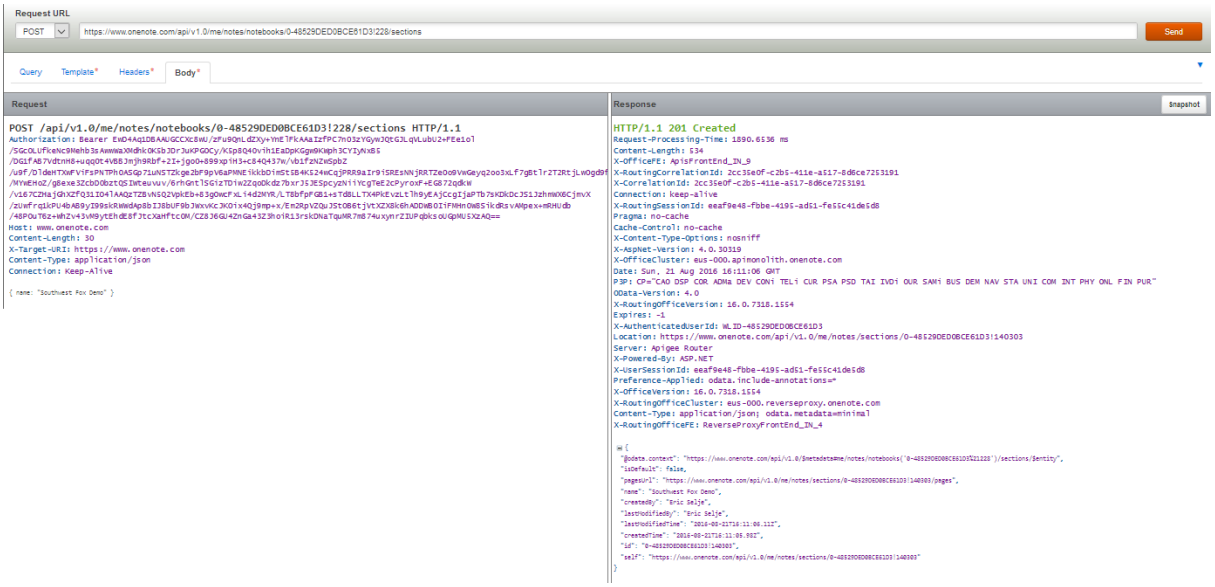
Notice how helpful the console is, giving you descriptions of each parameter required, and noting with a red asterisk when information is needed on tabs. Based on that tab it looks like there's some Header information that's required as well:



It shows that Content-Type is needed in the headers, and prefills it in with *application/json*, saving us that bit of work. Nice! On to the Body of the request:

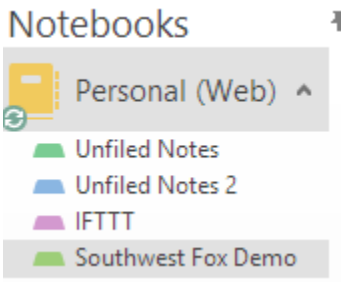


In the body it prefills a JSON template, leaving it up to you to fill in the specifics of course. Here's what I got back after sending that request:



Again you can see the complete request in the left panel and the response in the right. This time it's a 201 response, but it's still in the 200 range so we know that's a good thing. We also get the ID back of the newly created section, which we'll need to create a page in that example.

After giving OneNote enough time to synchronize the OneDrive version with my desktop version, I now see that I have this new section in my notebook:



Example 3: Adding a Page to a Section

Let's add some content to that new Southwest Fox Demo page we've created. The API allows you to send a subset of HTML[iii] in the body of the message, as well as images from a URL (which will be retrieved and inserted as a graphic). Note that all "input" tags are unsupported.

The API endpoint for creating a new page is a POST to *sections/{SectionID}/pages*. In the body tab of the Apigee console put in the HTML you want to create. It defaults to this HTML for you, which demonstrates a large number of the HTML supported by OneNote.

```
--NewPart
Content-Disposition: form-data; name="Presentation"
Content-Type: application/xhtml+xml
<?xml version="1.0" encoding="utf-8" ?>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-us">
  <head>
    <title>Page from OneNote API console</title>
   <meta name="created" content="2014-03-17T09:00:00-08:00" />
  </head>
  <body>
    <h1>HTML sample block</h1>

    <h2>The basics</h2>
    <p>For the most part, try to keep the HTML simple, and be
      sure to properly close all tags.</p>
    <p>Character encoding must be in UTF-8; the service doesn't
      accept other encodings</p>

    <h2>Overall structure</h2>
    <p>The normal &lt;html&gt;, &lt;head&gt; and &lt;body&gt; tags are
      expected, but the service is usually okay if they aren't included.</p>
    <p>Inside the &lt;head&gt; tag, we recognize only the &lt;title&gt; and
      One form of the &lt;meta&gt; tags.</p>
    <p>All includes, CSS, script, etc. inside the &lt;head&gt; and &lt;body&gt;
      blocks are ignored.</p>
    <p>As you can tell from the &lt;body&gt; tag's style attribute, in this
      HTML, the service ignores CSS styles</p>

    <h2>Block formatting</h2>
    <p>Paragraph (&lt;p&gt;) and line-break (&lt;br/&gt;) tags are handled
      properly. They get the &quot;Normal&quot; OneNote styling</p>
    <p>The &lt;div&gt; and &lt;span&gt; tags are recognized but generally
      don't have any significant effect, since CSS styling is ignored. A
      &lt;div&gt; tag acts like a &lt;p&gt; tag.</p>
    <p>The header tags &lt;h1&gt; through &lt;h6&gt; are recognized, and
      map to the OneNote Heading 1 through Heading 6 styles.</p>

    <h2>Simple character formatting</h2>
    <p>This paragraph shows how the API handles <b>HTML4+ bold
      &lt;b&gt;</b> tags</p>
    <p>...and <strong>HTML4+ strong &lt;strong&gt;</strong> tags</p>
    <p>...and <i>HTML4+ italics &lt;i&gt;</i> tags</p>
    <p>...and <em>HTML4+ emphasis &lt;em&gt;</em> tags</p>
    <p>...and <a href=".">HTML anchor &lt;a&gt;</a> tags.</p>

    <h2>Images</h2>
    <p>The &lt;img&gt; tag is supported. For example, here&apos;s a referenced
image: </p>
    <img src="http://officeimg.vo.msecnd.net/en-us/files/018/949/ZA103278226.png"/>
    <p>The OneNote API supports JPEG, GIF, TIFF and PNG images</p>
    <p>This next tag inserts a screenshot thumbnail image of the OneNote.com web
      page into the captured page, displayed as 500 pixels wide.</p>
    <img data-render-src="http://www.onenote.com" width="500"/>
    <h2>Tables</h2>
```

```
    <p>Tables are understood, but not table headers. Table headers are treated
      like normal rows. You can nest tables, but their content-layout ability
      is limited. </p>
    <p>Tables have to be &quot;regular&quot;, in that the service assumes all
      rows have same number of columns. Similarly, all columns have the same
      number of rows. More specifically, the service ignores colspan and rowspan
      attributes.</p>
    <p>You can set the table border to either "0" (no border) or "1" (with
border).</p>
    <table border="1"  >
      <tr>
        <td>First row First column</td>
        <td>First row Second column</td>
      </tr>
      <tr>
        <td>Second row First column</td>
        <td>Second row Second column</td>
      </tr>
    </table>
    <p>Lists of both types (&lt;ol&gt; and &lt;ul&gt;) are supported, and can
      be nested. But, the &quot;type=&quot; attribute is ignored.</p>
    <ul>
      <li>First unordered list item</li>
      <li>Second unordered list item<br>which contains another list
        <ol>
          <li>First (nested) ordered list item</li>
          <li>Second (nested) ordered list item</li>
        </ol>
      </li>
    </ul>
    <h2>Not (yet) supported</h2>
    <p>Nested tables are supported, but table header rows (&lt;th&gt;)
      are treated like normal table rows (&lt;tr&gt;)</p>
    <p>CSS styles are ignored at this time.</p>
    <p>Scripts are ignored.</p>
    <p>Other tags (e.g, &lt;article&gt;, &lt;header&gt;,
      &lt;footer&gt;, &lt;section&gt; and &lt;aside&gt;) are largely
      ignored, but their otherwise-valid contents are included.</p>
</body>
</html>
--NewPart--
.
```

Sending that returns a 201 (Created) result and looks like this in your OneNote:

Unfiled Notes | Unfiled Notes 2 | IFTTT | Southwest Fox Demo | +

Page from OneNote API console

Monday, March 17, 2014 9:00 AM

HTML sample block

The basics

For the most part, try to keep the HTML simple, and be sure to properly close all tags.

Character encoding must be in UTF-8; the service doesn't accept other encodings

Overall structure

The normal <html>, <head> and <body> tags are expected, but the service is usually okay if they aren't included.

Inside the <head> tag, we recognize only the <title> and One form of the <meta> tags.

All includes, CSS, script, etc. inside the <head> and <body> blocks are ignored.

As you can tell from the <body> tag's style attribute, in this HTML, the service ignores CSS styles

Block formatting

Paragraph (<p>) and line-break (
) tags are handled properly. They get the "Normal" OneNote styling

The and tags are recognized but generally don't have any significant effect, since CSS styling is ignored. A tag acts like a <p> tag.

The header tags <h1> through <h6> are recognized, and map to the OneNote Heading 1 through Heading 6 styles.

Simple character formatting

This paragraph shows how the API handles **HTML4+ bold <b>** tags

...and **HTML4+ strong ** tags

...and *HTML4+ italics <i>* tags

...and *HTML4+ emphasis * tags

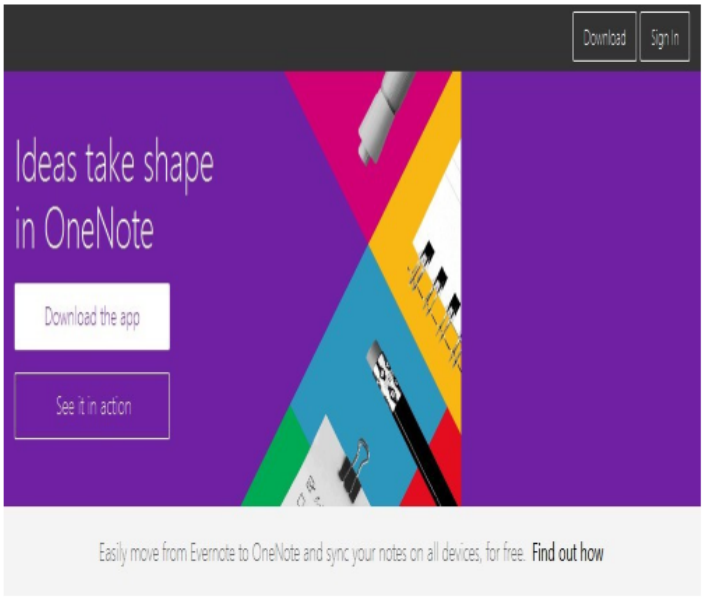
...and HTML anchor <a> tags.

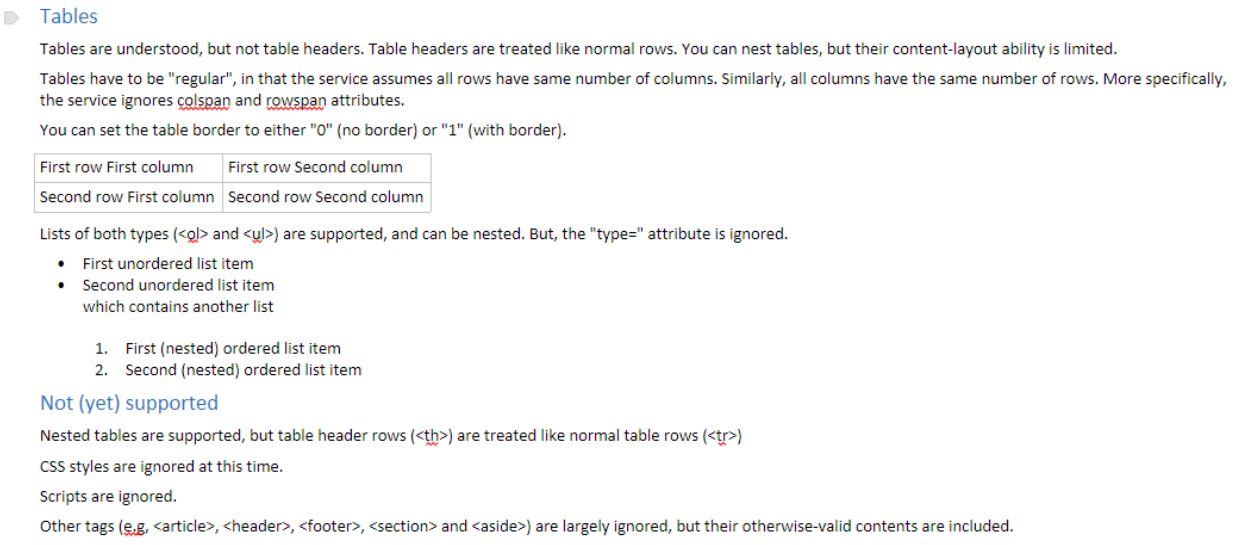
Images

The <img> tag is supported. For example, here's a referenced image:

The OneNote API supports JPEG, GIF, TIFF and PNG images

This next tag inserts a screenshot thumbnail image of the OneNote.com web page into the captured page, displayed as 500 pixels wide.





Tables

Tables are understood, but not table headers. Table headers are treated like normal rows. You can nest tables, but their content-layout ability is limited.

Tables have to be "regular", in that the service assumes all rows have same number of columns. Similarly, all columns have the same number of rows. More specifically, the service ignores colspan and rowspan attributes.

You can set the table border to either "0" (no border) or "1" (with border).

First row First column	First row Second column
Second row First column	Second row Second column

Lists of both types (<ol> and <ul>) are supported, and can be nested. But, the "type=" attribute is ignored.

- First unordered list item
- Second unordered list item
  which contains another list

  1. First (nested) ordered list item
  2. Second (nested) ordered list item

Not (yet) supported

Nested tables are supported, but table header rows (<th>) are treated like normal table rows (<tr>)

CSS styles are ignored at this time.

Scripts are ignored.

Other tags (e.g. <article>, <header>, <footer>, <section> and <aside>) are largely ignored, but their otherwise-valid contents are included.

Playing further around with the API using the Apigee Console

Deleting entire pages, sections, or notebooks is trivial, using the DELETE verb and sending in the appropriate ID.

Existing pages can theoretically be modified using the PATCH verb, but this isn't in the Apigee console and I couldn't find a good example of it anywhere. The theory behind the PATCH verb is that you'd send a *diff* of the content you'd like to add, using a special JSON format, to modify the content. In lieu of the PATCH command you could DELETE the page and POST the content again. This is more resource intensive than a PATCH would be.

I encourage you to use the Apigee console to play around with the OneNote API. It's a great tool to learn the API.

Using OneNote's API from Visual FoxPro

Finally, we get to the meat and potatoes. Using OneNote's API, we can create notebooks from pretty much any development tool, including .Net, iOS, Android, and of course our beloved Visual FoxPro.

You should note that in order to use the API the Notebooks you create *must* be stored either in OneDrive or in SharePoint. Notebooks that are only stored on your local hard drive and not synchronized to the cloud via OneDrive cannot use the API (which makes sense…how can the API running on Microsoft's servers access the files sitting in your Documents folder?)

There are the two steps[iv] needed to modify notebooks via the OneNote API:

1. Get an authentication "token" that we send with each API request.

2. Make the appropriate calls to the REST API.

Getting an Authentication Token

In order to make API calls, OneNote wants to be certain that you are someone who's *allowed* to make those calls. I can't just create notebooks on your OneDrive willy-nilly! This is done by getting an "access token" via the OAuth 2.0 protocol.

OAuth 2.0 was designed mostly for authenticating web applications, which makes the whole authentication part a fairly complex process, especially from Visual FoxPro programs. If we take this in small bites it isn't too hard.

There are two paths you can go by, depending on whether you want to access OneNote notebooks on OneDrive (what Microsoft calls "Consumer" apps) or notebooks on OneDrive Enterprise/Sharepoint/Office 365 (what Microsoft calls "Enterprise" apps).

This table should clarify the two paths:

Where is the notebook?	Microsoft's Term for This	Authentication Method
OneDrive	Consumer Apps	Microsoft Account
OneDrive Enterprise **Sharepoint** **Office 365**	Enterprise Apps	Azure AD

For the purposes of this whitepaper I'm only going to talk about "Consumer Apps", using the API to access OneNote notebooks on OneDrive using my Microsoft account. There's an exciting new authentication model called "v 2.0" [v] that Microsoft is cooking up that uses the same flow for both Consumer and Enterprise apps but as of this writing that is not completed and it does not grant access tokens for OneNote. If you're reading this you may want to check in to see if has been completed before going any farther.

Step 1: Register the Visual FoxPro application that you're writing with Microsoft.

This will generate a unique "Client ID" in Microsoft's system to identify your application, as well as a "Secret" that you send to the authorization service. This part is easy and you only need to do it once for each application that you're writing. Just go to https://account.live.com/developers/applications, sign in with your Microsoft ID (create one if you don't have one yet), click "Add an App", and give it a name.

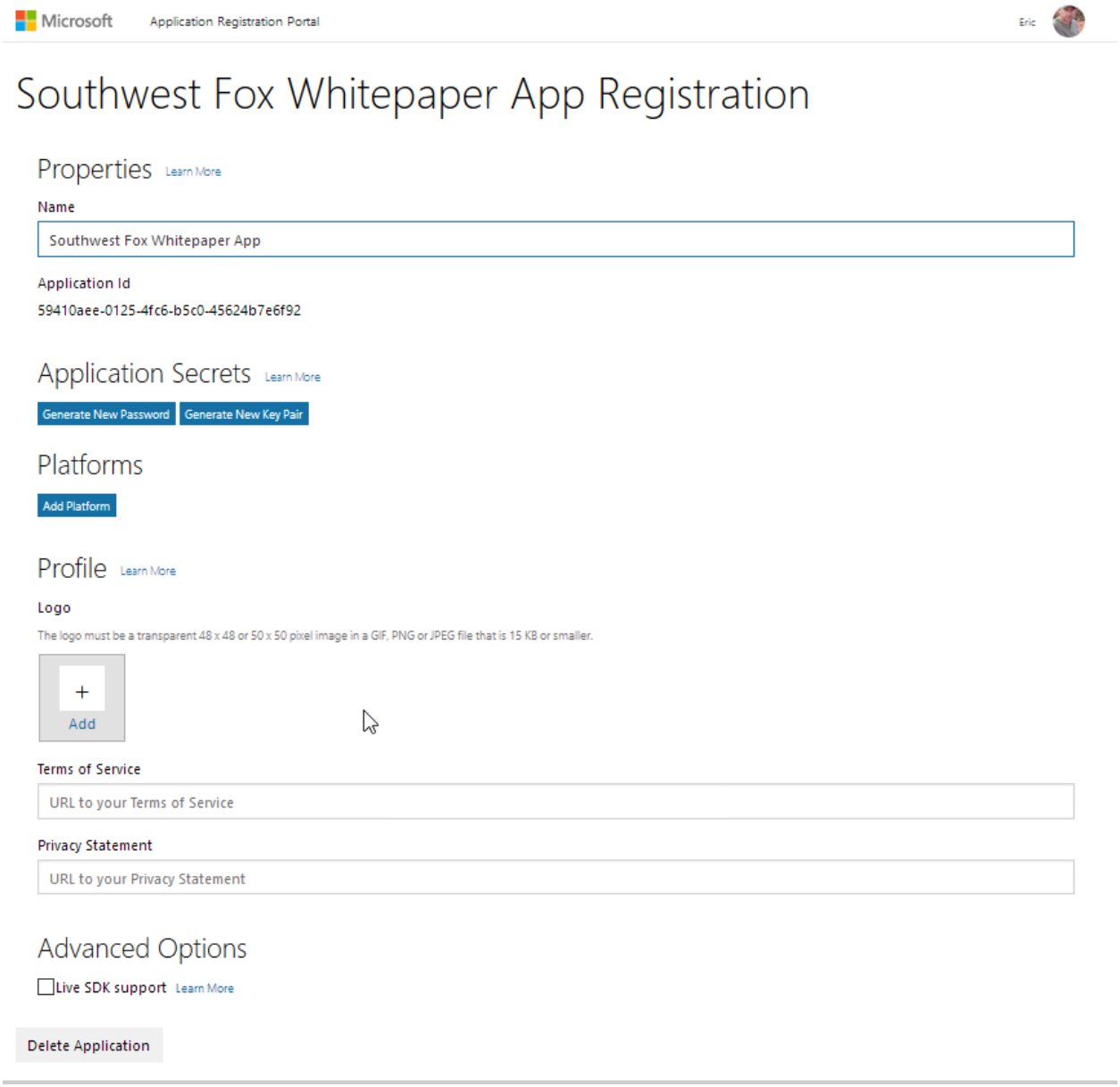
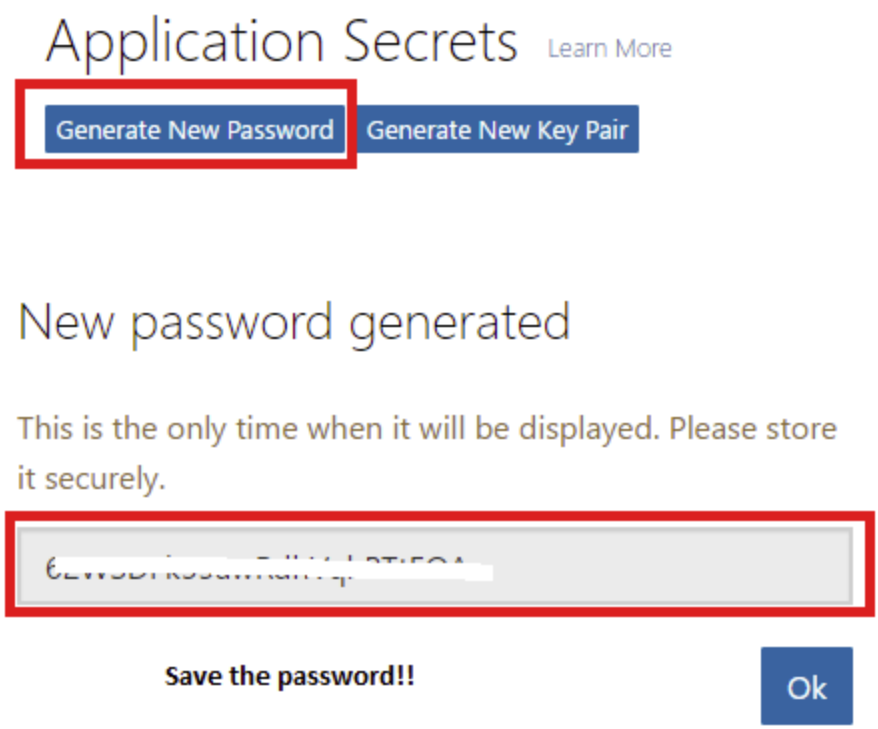


Figure 1: Registering Your App with Microsoft

The name of your app, along with the "Terms of Service" and "Privacy Statement" verbiage you enter on this screen, would be displayed to your users if they were using a web application or a Windows Store application, but because we're writing Visual FoxPro windows desktop applications so nobody will ever actually see these.

Keep track of that application ID shown below the name (see Figure 1). You'll need it when you use the API.

Next click on "Generate New Password". Keep track of that password too but do not save it anywhere someone else might get it. If you keep it in your VFP source code, be sure to use a tool like ReFox to obfuscate the code!



Step 2: Decide what permissions your app needs to function

While it might seem prudent to just ask for all permissions, you really should limit yourself to the lowest rights your app is actually going to use. Here are the permissions (a/k/a "scopes") you can request[vi]:

Scope	Description
office.onenote_create	Can view a list of the user's OneNote notebooks and create new pages, but cannot view or edit existing pages. Can enumerate the user's notebook hierarchy and create pages in any location.
office.onenote_update_by_app	Can create, view, and modify all pages created by the app.
office.onenote_update	Can create, view, and modify any content in the user's OneNote notebooks and pages.
office.onenote	Can view OneNote notebooks and pages but not modify them.
wl.signin	A Microsoft account permission scope. Allows your application to take advantage of single sign-on capabilities.
wl.offline_access	A Microsoft account permission scope. Allows your application to receive a refresh token so it can

Scope	Description
	work offline even when the user isn't active. This scope is not available for the *token* flow.

Step 3: Utilize the OneDrive API Software Development Kit (SDK) to get an access token

This gets a little confusing for a couple of reasons.

1. Note that I said the One**Drive** API SDK, not the One**Note** API. The OneDrive API handles the authentication for us. Also the SDK used to be called the "Live SDK" but in typical fashion Microsoft renamed it. If you see references to the Live SDK, know that what you're looking at is outdated.

2. The OneDrive API is not written in Visual FoxPro, nor does it have examples written in Visual FoxPro. However, it does have C# version of the SDK which means we can use Rick Strahl's most excellent wwDotNetBridge to use the SDK.

Download the C# SDK from https://dev.onedrive.com/SDKs.htm and open the project in Visual Studio. Tweak the settings to include your Secret Keys and authentication information, and compile it to a DLL. Then call wwDotNetBridge from within Visual Foxpro to get the authentication token you'll need to make REST calls.

You can make those REST calls and parse the resulting JSON with more of the wonderful tools that WestWind provides, like wwIPStuff and wwJSONSerializer. I've unfortunately run out of time to get these examples together before the papers are due but I'm going to try to get something together for the conference so check this space for that. Perhaps you're interested in helping out with this? I'd love to collaborate with you!

Conclusion

Microsoft is doing some exciting things with Office 365, OneNote, and the new Graph API. According to Vijay Sharma, Product Manager for OneNote, here's what's next for the OneNote API[vii]

- Embed live OneNote pages on a webs
- OneNote Add-Ins Sharing
- Better tools to showcase apps
- More Samples

I sincerely hope this presentation gets you excited about using OneNote and perhaps even taking advantage of the API. I would love to see a VFPX project get created

[i] https://www.onenote.com/
[ii] https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
[iii] https://msdn.microsoft.com/en-us/office/office365/howto/onenote-create-page#supported-html
[iv] https://youtu.be/BIroRqO_BpM?t=16
[v] https://msdn.microsoft.com/en-us/office/office365/howto/authenticate-office-365-apis-using-v2
[vi] https://msdn.microsoft.com/en-us/office/office365/howto/onenote-auth#choose-onenote-permission-scopes-consumer-apps
[vii] https://channel9.msdn.com/Events/Visual-Studio/Connect-event-2015/310

