

System Requirements

Minimum Configuration

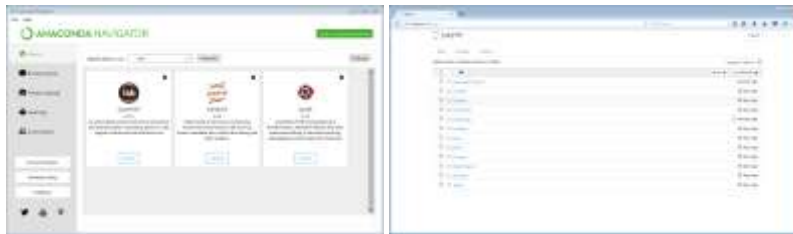
- Anaconda 4.x
- Python 3
- ODBC drivers for Microsoft SQL Server
- PowerCampus 8.7.1

Known Good Configuration

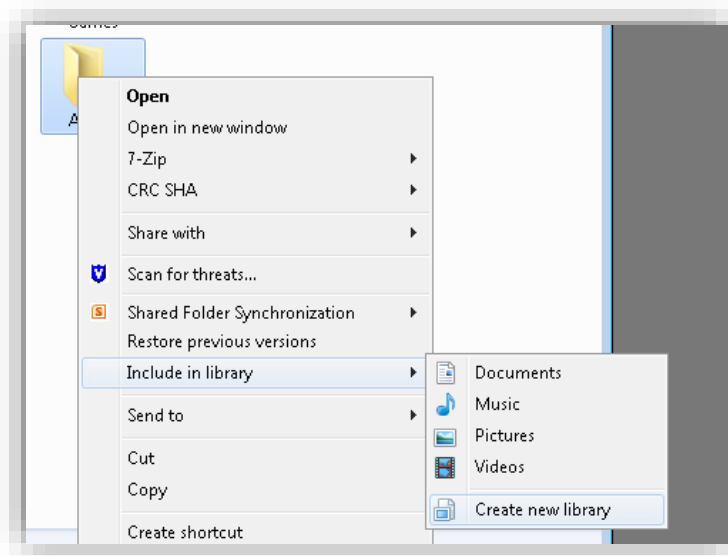
- Windows 7 Professional x64
- Anaconda 5.0.0 (64-bit)
- Python 3.6
- ODBC drivers for Microsoft SQL Server
- PowerCampus 8.7.1

Setting Up Your Python Environment

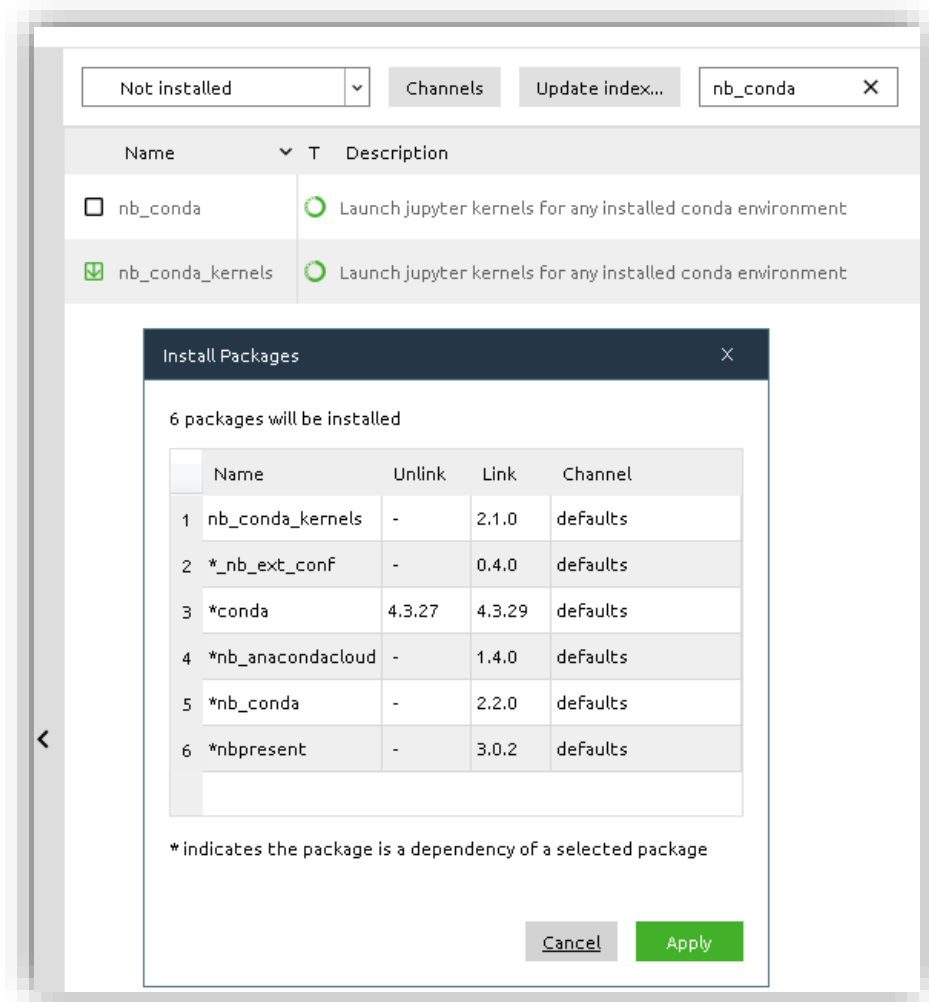
1. Install Anaconda3 from www.anaconda.com. Install for all users.
 - a. From the Start Menu, run Anaconda Navigator at least once.
 - b. From the Start menu, run Jupyter Notebook at least once.
 - c. Both of these should successfully show an interface:



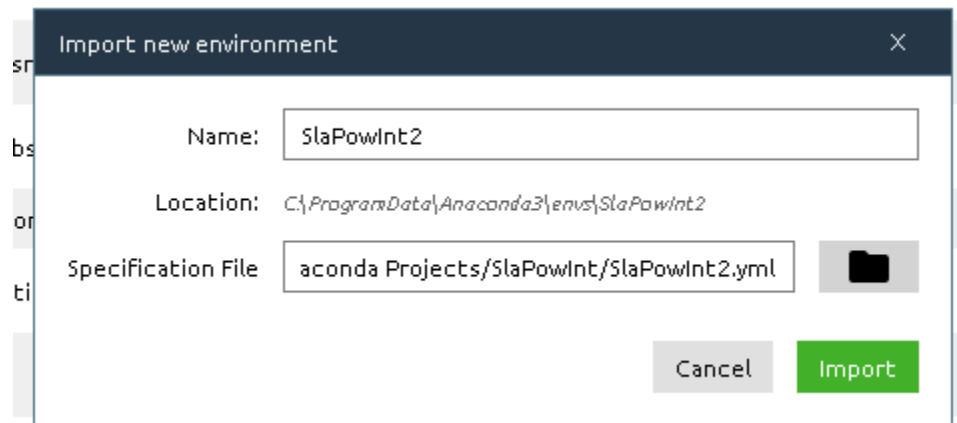
2. Add Anaconda Projects (C:\Users\<user>\Anaconda Projects) to the system Libraries for convenience:



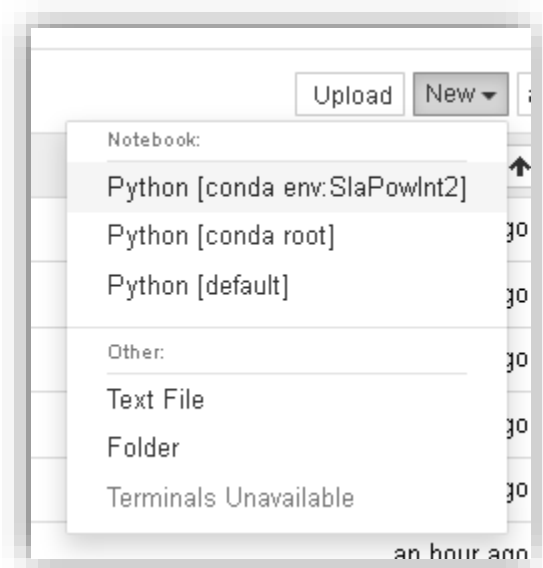
3. Unzip SlaPowInt.zip into the Anaconda Project folder.
4. Launch Anaconda Navigator as administrator.
 - a. Go to Environments > root.
 - i. Filter on *Not installed* and search for *nb_conda*.
 - ii. Mark *nb_conda_kernels* for installation and click Apply, then Apply.



- b. Click Environments > Import.
 - c. Name the environment whatever you want (e.g. SlaPowInt2) and import the SlaPowInt2.yml descriptor file.



- d. Note that the location needs to be `C:\ProgramData\Anaconda3\envs\<name>`. If you are not running Anaconda Navigator as administrator, it will attempt to create the new environment at `C:\Users\<user>\AppData\Local\conda\conda\envs\<name>`. I experienced problems with the AppData location with Anaconda 4.4.x versions and did not attempt on to use it on 5.5.0.
5. Feel free to close Anaconda Navigator when it has finished downloading and installing the necessary packages.
 - a. Either from the Start menu or from the Home tab of Anaconda Navigator (making sure to select *Applications on root*), launch Jupyter Notebook.
 - b. Make sure that if you click New, the SlaPowInt2 environment is available. If it is, congratulations! Your Python environment is ready to go. If not, you need to troubleshoot and poke around in Anaconda until you can see multiple kernels (environments) available. You may also be able to get away with using the default/root environment and installing any missing packages the SlaPowInt script complains about.



Other Setup

1. You must already have the PowerCampus Web API installed.
 - a. You must know its URL.
 - b. You must have Basic authentication enabled in IIS for the site and have a working username/password combination.
 - c. HTTPS with a valid certificate is strongly recommended.
 - d. You can test authentication and API status in your browser by visiting the version page (example <https://webapi.mcny.edu/api/version>).
 - e. You must have configured *PowerCampus WebAPI Services\Config\ApplicationSettings.config*.
 - f. You must have configured *recruiterMapping.xml* either by hand (not recommended) or via the PowerCampus Mapping Tool. For convenience, I recommend creating a dummy database in the style of Ellucian CRM Recruit, using *CodeTableInformation.xml* as a guide. Then you can fill values in the dummy tables to allow using the drop-down and automap features of the mapping tool. This is especially useful when dealing with the long list of country codes from Slate and PowerCampus.
2. On the computer/server running Python, configure a local ODBC connection to Campus6. It should also be able to access whatever database will store status log information.
3. On the server with the PowerCampus Web API and the PowerCampus Mapping Tool installed, share the folder containing *recruiterMapping.xml* on the network so that SlaPowInt can read it. This share should be restricted to read-only permissions for only the user running SlaPowInt. The other files in this folder contain sensitive database information.
4. You must have all your PowerCampus P/D/C combinations correctly configured in Self-Service under Admin > PowerCampus Setup > Program Setup, as these will be used by the Web API.

Import Slate Objects

Import any of the following objects you wish to use into Slate using the Briefcase Import tool in under Database. All of these require lots of modification or a complete rebuild to fit your data.

Please try in your Slate test environment first! MCNY maintains a test PowerCampus database, test Slate environment, and test instances of Self-Service and the PowerCampus Web API so that the entire ecosystem can be tested without any risk of impact to production databases.

1. applicants_export
 - a. Briefcase ID 96fea9db-6266-3f9e-8e59-977006848d0a
 - b. A query that returns updated applications in a flat format that SlaPowInt will transform and insert into the PowerCampus Web API.
 - c. You will need to enable Web Services for this query and specify a username and password.
2. scheduled_actions_2
 - a. Briefcase ID 9d2ae970-e3e9-1b9f-f099-08f21f50b452
 - b. A Custom SQL query used to convert Slate Checklist items into PowerCampus Scheduled Actions. The input parameter is a concatenated list of ID's; please see query's notes for details. See also <https://technolutions.zendesk.com/hc/en-us/community/posts/115001209231-Are-table-valued-or-XML-parameters-possible-for-web-services-queries->.
3. SlaPowInt Web Post

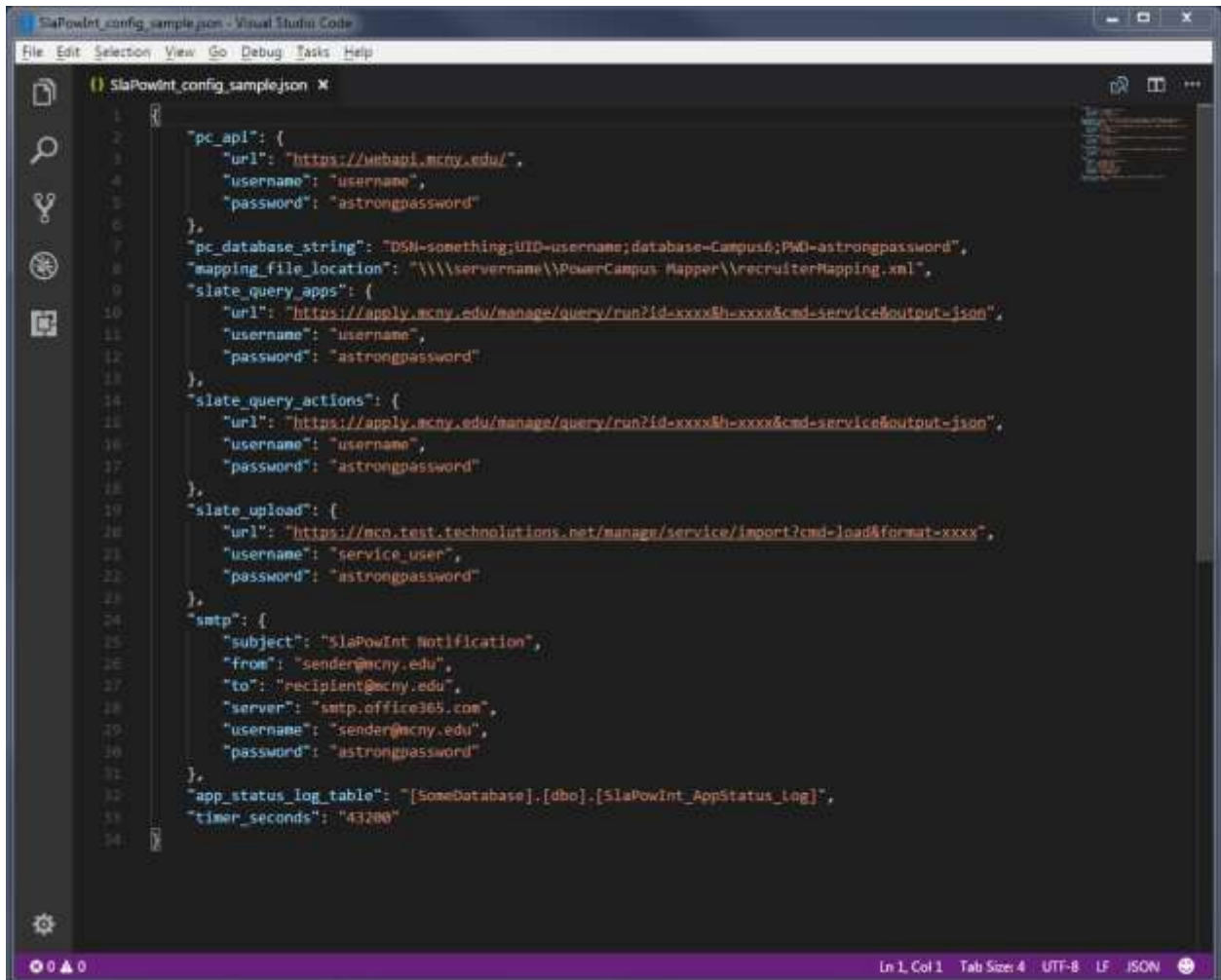
- a. Briefcase ID 59114137-8ea4-c473-4bac-06ccb0fd605a
- b. This is a Source Format used for posting JSON data back to Slate.

Creating SQL Database Objects

1. Create an SQL user for SlaPowInt to connect with. Using the SA account is not required nor recommended.
2. To create the needed stored procedures, run all the scripts with filenames like MCNY_SlaPowInt_%.sql. You are welcome to replace MCNY with the name of your school. We use the *MCNY_* prefix to indicate anything custom that we place in our Campus6 database.
3. Use *SlaPowInt GRANT EXEC.sql* to grant access to your user to the created objects.
4. The other SQL files are provided merely for informational purposes.
5. You must modify the following procedures as they contain hardcoded code table information:
 - a. MCNY_SlaPowInt_GetAcademic
 - b. MCNY_SlaPowInt_UpdContactPrimacy
 - c. MCNY_SlaPowInt_UpdAction
 - d. MCNY_SlaPowInt_UpdAcademicAppInfo

Configure JSON Configuration File

1. Open Anaconda Projects\SlaPowInt\SlaPowInt_config_sample.json in your favorite text editor (Visual Studio Code and Notepad++ are great and free!) and edit the sample configuration values with real data.



```
1 {
2   "pc_api": {
3     "url": "https://mhapi.mcnyc.edu/",
4     "username": "username",
5     "password": "astronpassword"
6   },
7   "pc_database_string": "DSN= something;UID=username;database=Campus6;PWD=astronpassword",
8   "mapping_file_location": "\\servername\\PowerCampus Mapper\\recruiterMapping.xml",
9   "slate_query_apps": {
10     "url": "https://apply.mcnyc.edu/manage/query/run?id=xxxx&h=xxxx&cmd=service&output=json",
11     "username": "username",
12     "password": "astronpassword"
13   },
14   "slate_query_actions": {
15     "url": "https://apply.mcnyc.edu/manage/query/run?id=xxxx&h=xxxx&cmd=service&output=json",
16     "username": "username",
17     "password": "astronpassword"
18   },
19   "slate_upload": {
20     "url": "https://mcn.test.technolutions.net/manage/service/import?cmd=load&format=xxxx",
21     "username": "service_user",
22     "password": "astronpassword"
23   },
24   "smtp": {
25     "subject": "SlaPowInt Notification",
26     "from": "sender@mcnyc.edu",
27     "to": "recipient@mcnyc.edu",
28     "server": "smtp.office365.com",
29     "username": "sender@mcnyc.edu",
30     "password": "astronpassword"
31   },
32   "app_status_log_table": "[SomeDatabase].[dbo].[SlaPowInt_AppStatus_Log]",
33   "timer_seconds": "43200"
34 }
```

2. The Slate query URL's come from the Web Services link in the Slate query editor.
3. Value `timer_seconds` is how long, in second, SlaPowInt should wait between each synchronization.
 - a. Idea: Create two instances of SlaPowInt, one with a 24-hr timer, and one with a 1-hr timer of only updated applications using Slate's built-in update queue on the `slate_query_apps` query.
 - b. To create a second instance of SlaPowInt with another timer, just copy the configuration file with a new name. You will later clone the .ipynb file (iPython Notebook), replace the name of the configuration file in it, and then run both notebooks in parallel.
4. SMTP values can be supplied by your email administrator.
5. `slate_upload` is used to send data back to Slate. See *Pushing Data into Slate's Web Service Endpoint to Import records using Upload Dataset* section of <https://technolutions.zendesk.com/hc/en-us/articles/115000689431-Web-Services> on how to craft this URL.
 - a. This username and password should be a Slate service user with Upload Dataset permissions.

Running SlaPowInt

SlaPowInt Trigger