Python SDK Model Gateway User Guide

OVERVIEW

This document describes the new SDK plugin layout and support for easier adding of sensors and features.

PROGRAMMERS REFERENCE SDK Plugins

Plugin structure and theory of opeartion

The gateway SDK has be re-organized to be easier to maintain and add other features. The new directory structure under ../sampe is as follows.

1) IoTConnectSDKengine.pyc – This is the main engine and is responsible for importing the onboard functions, plugins and their configuration files. It also starts the cloud connection, sends accumulated data to the cloud, operates the watchdog timer if configured, and monitors for several error conditions.
2) IoTConnectSDKonboard.pyc – This contains functions for communicating to the IoTConnect REST API. These are used for things like AddAttribute, AddCommand, AddRule, Create Template, Enroll device, DeleteAttribute, DeleteCommand, and DeleteRule.
3) Several subdirectories the start with "IoTPlugin" contain the plugin code and configuration file for that sensor or sensor family
4) SensorPlginTemplate directory is an example of a simple plugin
5) SensorPlginTemplateAdvanced directory contains an example of a complex plugin, with OnBoarding samples to AddAttribute, AddCommand and Queue data.

The following functions and descriptions are supported from the OnBoard feature that IoTConnectSDKengine.pyc imports.

1) DeleteAttribute(AttributeGuid) - This function deletes the attribute from the cloud referenced by the GUID.
2) QueueSensorValue(name, my_config_dict, value, lastvalue) - This function puts data on the queue for sending to the cloud. Returns new lastvalue .
3) GetAccessToken() – This function gets the Access Token for the cloud RestApi
4) CloudSetupObjects() – Setup the current sensors described in the plugins.
5) CloudConfigureDevice() – If the device template doesn't exist this configures the cloud and enrolls the device
6) GetAttributes() – Returns a list of current cloud attributes for the device
7) AddAttribute(my_config_dict) – Adds the attribute to the cloud
8) Enroll() – Enroll the device on the cloud
9) AddCommand(my_config_dict) – Adds the command
10) DeleteCommand(commandGuid) – Deletes the command GUID
11) GetCommands() – Returns a list of commands on the cloud for this device

The main SDK IoTConnectSDKengine.py will handle all the cloud on boarding and locating all the current plugins. After the plugins have been processed and the cloud attributes/commands/rules/etc have been update using the cloud REST API, the main engine will connect to the cloud and start transferring data.

SDK Templates

The SDK comes with four templates for creating sensors/commands/etc that are not currently available. The four templates are SemsprPluginTemplate, SensorPluginTemplateAdvanced, CommandPluginTemplate, and SensorFnPluginTemplate. Use these templates to generate new sensors/commands/rules, etc. To enable the new code just rename the directory to IoTPluginXXXX ,IOTFnPluginXXX, or IoTPluginComandXXX. In the main engine directory, then restart the SDK.

- SensorPluginTemplate is the easiest to use and most cases developers can use this template.
- SensorFnPluginTemplate is used when the sensor isn't directly supported by the gateway. This is used as a post process function that is called by the IOEXPANDER to process raw data.
- CommandPluginTemplate is used to add cloud commands to the cloud gateway instance. For example the sensor may have an output which would require a Cloud command to operate the output.
- SensroPluginTemplateAdvanced is more useful if you have a hot plug situation or where sensors come and go. It's also helpful for sensors that require adding commands/rules or attributes dynamically based on the current situation.

For the AddAttribute function you must either have settings in the IoTConnectSDK.conf file or specify the config_dict dictionary. The required config_dict/.conf file pararmeters are as follows:

```
config_dict = {}

config_dict['name'] = name

 config_dict['description'] = description

config_dict['units'] = units

config_dict['value'] = "NUMBER"

config_dict['edgeaggregatetype']  =
str(my_config_parser_dict["OmegaSensorConfiguration"]["edgeaggregatetype"])

config_dict['edgetumblingwindow'] =
str(my_config_parser_dict["OmegaSensorConfiguration"]["edgetumblingwindow"])
```

For the AddCommand function you must either have settings in the .conf file or specify the config_dict. The required config_dict/.conf file parameters are as follows:

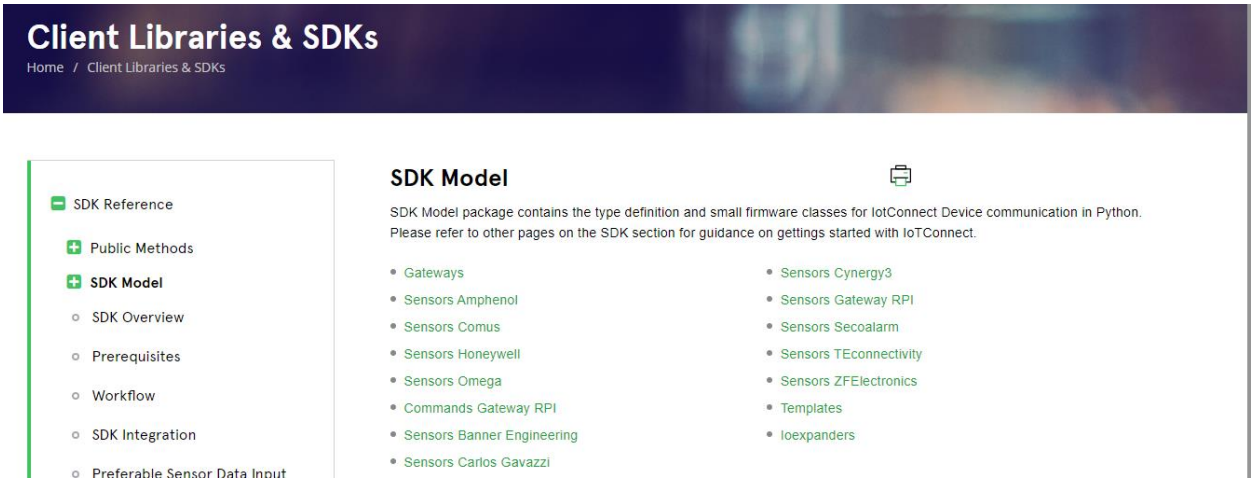
```
config_dict = {}

config_dict['commandname'] = str(name) + "Set"

config_dict['command'] = "OmegCommand " + str(name) + " 100"

config_dict['hasparameter'] = 0

config_dict['requiresack'] = 0
```

```
config_dict['isiotcommand'] = 0
```

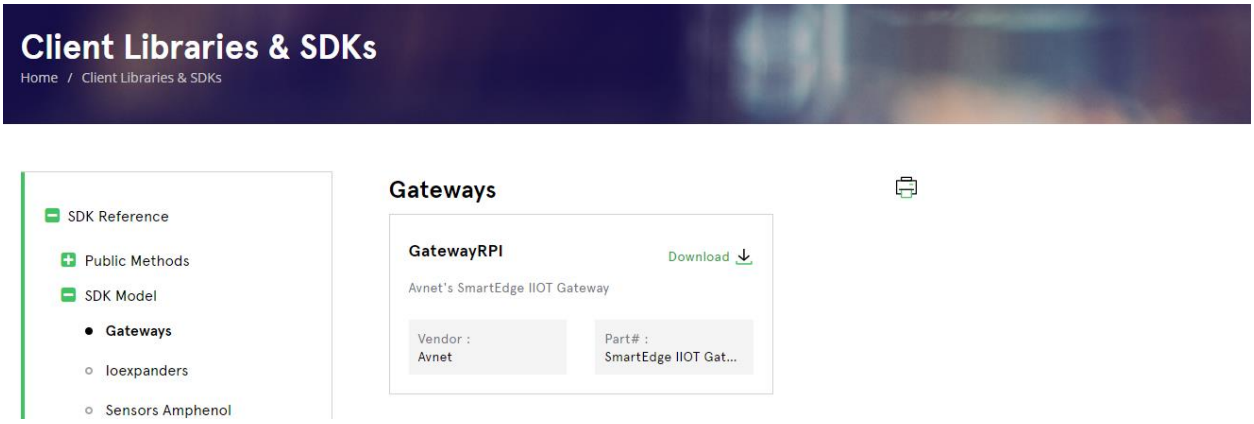
For the Advanced plugin Template, The on boarding can and should probably be handled by the plugin. The plugin can do what it needs to access the sensor device connected. Note, if you have custom imports for the plugin put them in the plugin directory. The plug adds to the path for searching imports

Selecting and Downloading SDK Models

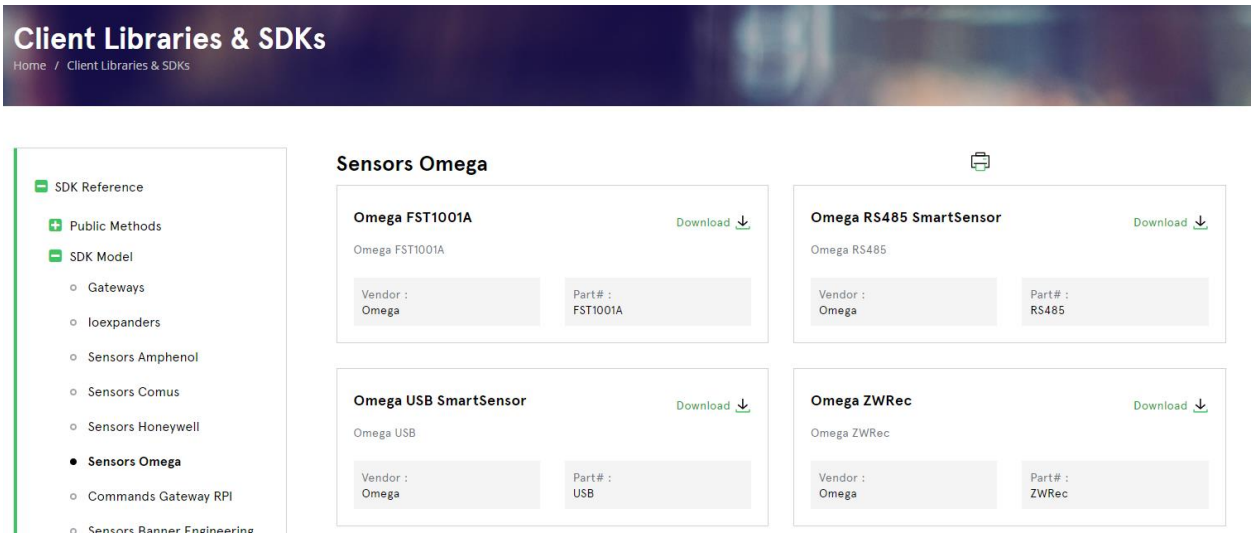
Start by logging on to the "site". The screen should look similar to this.



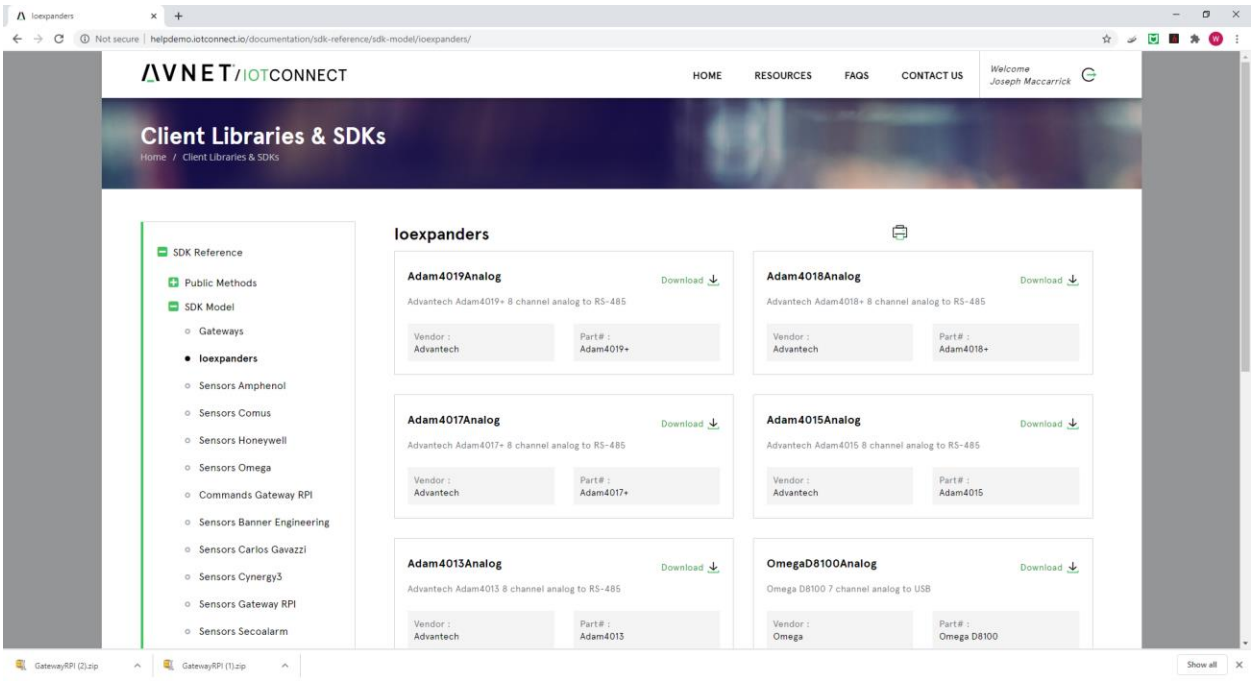
First selected from gateways. Then choose the gateway and the screen should look similar to this. Select download and the zip file for that SDK model will be downloaded to the PC downloads area. Move that zip file to a new subdirectory on the PC. Next return to the page above and start selecting other components. Repeat the procedure of moving the newly downloaded zip file to the new subdirectory on the PC.



Selecting other components for example the Omega FST100A, the “sensors omega” area will result in a screen similar to this. Select the model and download. Move the downloaded file to the new subdirectory.



To generate python code for a sensor not on the list, then select the templates area. The ioexpanders area contains several gateway addon components for use with analog sensors such as Thermocouples, Current sources(ma), voltage sources, etc. The ioexpanders screen should look similar to this.



Installing Models on the Gateway

Download the directory from your PC to the gateway. Copy the files in the PC directory to the files on the gateway in /opt/avnet-iot/IotConnect/sample. To do this follow these instructions.

DOWNLOAD A NEW IMAGE OR UPDATE FILES ON THE DEVICE

There are several options to update files on the device. Also, the micro usb port near the center of the board can be used to download a new image or update files on the device. It can also be used to save your current gateway image.

To download a new image or update files on the device, get the custom files from our github site. The rpiboot directory contains the needed files to use on your PC linux system for this feature. Note: The rpiboot must be complied for your version of linux. Run "make" to do this. There are 2 scripts that can be used after the code is compiled. The scrip rpi.sh is the non secure method to do the updates. First plug in the micro usb to the PC, then power on the gateway. The gateway will show up as a new /dev/sd* file system and you can mount either the /boot area or the /rootfs area if you intend to update files. To update a new image use the linux "dd" command. To save the current image use the linux "dd" command.

The gateway USB port can be used to update files on the device. Put your files on a usb stick , then plug it into the gateway. On a gateway terminal you will have to mount the usb device as it does not auto mount for security reasons. Then simply copy the files from the USB stick.

Here's a link that describes the procedure. Please use our version of usbboot called RPIBOOT in the github area.

The production image can be downloaded from this site.

https://docs.avnet.com/amer/smart_channel/smartedge/

https://www.raspberrypi.org/documentation/hardware/computemodule/cm-emmc-flashing.md

Now login to the gateway and run the following commands

1) "cd /opt/avnet-iot/IoTConnect/sample"
2) "unzip GatewayRPI.zip"
3) "sudo chmod 777 setup.sh"
4) "sudo ./setup.sh"

Adding and Removing Models on the Gateway

To add models to the gateway do the following.

1) Download the new models to the PC
2) Copy the .zip files to the gateway as described previously
3) Run "sudo ./setup.sh" on the gateway again.

To remove models from the gateway do the following.

1) On the gateway run "sudo rm –R IoTxxxxx/ for the model you wish to remove
2) Run "sudo ./setup.sh" on the gateway.
3) During the setup answer Yes to reset configuration. Note: You will have to re-enter the new configuration completely!

WHAT HAPPENS DURING BOOT

The boot process can be observed on a HDMI screen. A USB keyboard can be used to log into the system and run commands.

The boot process starts with some initial messages from the bootloader indicating the system is powering up. The next set of messages will be coming from the linux kernel. Just before the login prompt you will see a message indicating "My IP address is x.x.x.x" followed by the name of your device "IotGateway xxxxxxx" where xxxxxxx is the 8 digit serial number of the device. Once the kernel boots, all the operations for the gateway can be found in /etc/rc.local. These operations are required to use the mobile app and setup the device to talk to the IoTConnect cloud. Since this is a Trusted Platform using TPM 2.0 the boot up PCR list can be utilized, so that your system setup can be verified as trusted. Run the command "tpm2_pcrlist" and see SHA256 values for PCR 0 and 10. For more information on using the TPM tools installed on your system see the following web site https://github.com/tpm2-software/tpm2-tools/wiki/How-to-use-tpm2-tools . Our specific linux kernel version is "Linux raspberrypi 4.14.79-v7+" and can be verified by using the following terminal command "uname –a". The specific date stamp for the image creation can be found using the following terminal command "cat/usr/bin/ImageDate.txt".

You will also be able to use this setup as your linux terminal. When you see the following enter the text below.

raspberrypi login: avnet

Password: avnet

Now the files are properly installed on the gateway. Run the mobile app to setup and connect the gateway to the cloud.