# *i3 Application Wrapper*

The following is an example of an i3 application wrapper.  This simple application displays any messages the application has been approved to receive is displayed on the system console.  Such an application serves to demonstrate the construction of an application wrapper and in on operational environment, it is also useful for testing purposes.  When bringing up a device for the first time, the messages from that device can be directed to such an test application to verify the messages are properly being emitted from the device and passed through the i3 message fabric.

Note: this application wrapper assumes it is receiving asci/text messages. If there is a possibility that the device may be transmitting binary messages, some additional logic will be needed to ensure that only the printable characters are displayed.

```
import time

from sdk.i3_broker_application import I3BrokerApplication

BROKER_HOST = '18.117.224.165'  #IP of the i3 core information fabric
BROKER_PORT = 1883              # port number on the i3 core node
APPLICATION_NAME=’hello_world’  # the wrapper name
APPLICATION_PASSWORD = '123456' # password used to authenticate this wrapper
DEVICE_NAME = 'device'          # the name of the device this app wants data from
DEVICE_TOPIC = 'temperature'    # the data type the wrapper is interested in

def on_message(topic, message): # when the wrapper gets a msg, print it out
    print('New message received', topic, message)

i3_broker_application = I3BrokerApplication(BROKER_HOST, BROKER_PORT,
APPLICATION_NAME, APPLICATION_PASSWORD)

i3_broker_application.connect()

i3_broker_application.on_message(on_message)
i3_broker_application.subscribe(DEVICE_NAME, DEVICE_TOPIC)

time.sleep(60 * 60 * 24)        # keeps the app alive for one day

i3_broker_application.disconnect()
```

- The *import* commands make sure the python code has access to the python system time and to the i3 data structures and class software.
- The next six (6) parameters are the key parameters used by the i3 system to establish a connection between this application and the core i3 software.
    - BROKER_HOST is the IP address of the machine that hosts the i3 software.
    - BROKER_PORT is the port number this application will monitor for new information coming from authorized data streams.
    - APPLICATION_NAME is the logical name of this wrapper and it must correspond to the application name as defined in the i3's user menus. Note the file name and the wrapper names are independent of each other and can be named differently.
    - APPLICATION_PASSWORD: is the password used to authenticate this wrapper with the i3 software. This parameter much match the password associated with the application password as defined in the user menus.
    - DEVICE_NAME is the name of the device that this application is seeking information from. This should match the device name as listed in the Device Submenu of the Spaces Menu and as displayed on i3's Application Desktop (or the Data Streams submenu if the connection has not yet been authorized).
    - DEVICE_TOPIC is the name of the data type as displayed on the Application Desktop (or the Data Streams submenu if the connection has not yet been authorized) and links to a data type associated with a device in the Devices submenu of the Spaces Desktop.

The *def on_message* sequence is used to define a function that will be called when a new information message is detected for which that application has been approved to receive. For this example wrapper, the software will print the information topic type and message to the console. The process of posting the message to the console will be executed for every message received by this wrapper until the wrapper process is terminated.

The *i3_broker_application = I3BrokerApplication(BROKER_HOST, BROKER_PORT, APPLICATION_NAME, APPLICATION_PASSWORD)* line creates the object needed to allow this wrapper to communicate with the i3 core software. The *i3_broker_application.connect* command uses the created object to communicate with the i3 core software on the BROKER_HOST over port BROKER_PORT and authenticate this wrapper as being a valid i3 related process with the APPLICATION_NAME and APPLICATION_PASSWORD.

The *i3_broker.application.on_message* tells the i3 software that whenever a new data type is received, the *on_message* function is called.

*The i3_broker_application.subscribe* is used by the application to subscribe to a specific topic that is associated with a defined data type.

In order for a wrapper to receive a data type, the wrapper must be approved by the device owner to receive these data types from the requested device AND the application has to subscribe to the data topic as a part of the wrapper logic.

A information message is defined by its source device name and data type prefaced by the ‘device’ keyword. For example, if an application wants to receive the data type “location” from device “SAN-truck-1”, the application wrapper would subscriber to the data topic “devices/SAN-truck-1/location”

A single application can be subscribed to many different device topics at the same time. For example, if the application subscribes to “devices/SAN-truck-1/location” and “devices/SAN-truck-2/location” they would receive location messages from truck-1 and truck-2 (assuming the device owners have authorized the wrapper to receive the location data type from both trucks).

The *time.sleep* command keeps this application running for 24 hours and then the wrapper will terminate. In effect, each message sent to this wrapper over a 24 hour period will be posted to the console screen. If the *time.sleep* command is deleted, then the wrapper will run continuously and show all messages that come through the I3 software. If the duration is changed to 10 minutes, then the console window will show the messages that come in over the next 10 minutes and then the wrapper will terminate.

The "*i3_broker_application.disconnect()*" gracefully terminates the connection between the wrapper and the i3 core before the wrapper shuts down.

When the wrapper development is complete, if the wrapper code is to be installed on the machine that is running the i3 core software, it should be sent to the I3 administrator so they can install it in an appropriate I3 application library. If the wrapper code is to be run on an edge processor (real or virtual) that is independent from the i3 core node (e.g. a front end processor), the developer can install the software on that computer directly.