CyberQInterface Documentation

Release 1.0.0

Bryan Kemp

September 05, 2013

CONTENTS

The CyberQInterface is a python library for communicating with the BBQ Guru CyberQ Wi-fi temperature control system. This system provides accurate monitoring of both Pit and Food temperature and tight control of the Pit temperature.

CHAPTER

ONE

TABLE OF CONTENTS

1.1 API

Class to interface with BBQ Guru's CyberQ Temperature Control System

ChangeLog:

Ver	Date	Editor	Notes
0.1	09/29/2012	Bryan Kemp	Initial Commit
0.9	11/03/2012	Bryan Kemp	Added distools functionality
1.0	03/29/2013	Bryan Kemp	First release

class `cyberqinterface.`**`CyberQInterface`**(*host=None*, *headers=None*)

Web Interface to BBQ Guru's CyberQ Temperature Controller System.

`getAll`()

Get All parameters from CyberQ

Keyword arguments: None

Returns: All Object

Example Usage: cqi.getAll()

`getAllXML`()

Get AllXML from CyberQ

Keyword arguments: None

Returns: All Xml

Example Usage: cqi.getAllXML()

`getConfig`()

Get Configuration from CyberQ

Keyword arguments: None

Returns: Config Object

Example Usage: print cqi.getConfig().FOOD1_TEMP

`getConfigXML`()

Get ConfigXML from CyberQ

Keyword arguments: None

Returns: Config Object in XML

Example Usage: cqi.getStatus()

getStatus()

Get Status from CyberQ

Keyword arguments: None

Returns: Status Object

Example Usage: print cqi.getStatus().FOOD1_TEMP

getStatusXML()

Get StatusXML from CyberQ

Keyword arguments: None

Returns: Status Object

Example Usage: cqi.getStatusXML()

rampLookup(*code*)

Provides a text representation of the food that is monitored for ramping

Value from API	String Value
0	OFF
1	FOOD1
2	FOOD2
3	FOOD3

Keyword arguments: <int> code - the code from the object or xml

Returns: <String> Which food, if any, is being monitored for ramping

Example Usage: cqi.rampLookup(cqi.getStatus().COOK_RAMP)

sendUpdate(*parameters*)

Description: sendUpdate validates new parameters and sends update to CyberQ

Possible parameters:

Parameter Name	Definition
COOK_NAME	Pit Sensor name in plain text
COOK_SET	Pit probe target temp in current units
FOOD1_NAME	Food 1 name in plain text
FOOD1_SET	Food probe 1 target temp in current units
FOOD2_NAME	Food 2 name in plain text
FOOD2_SET	Food probe 2 target temp in current units
FOOD3_NAME	Food 3 name in plain text
FOOD3_SET	Food probe 3 target temp in current units
_COOK_TIMER	Set the countdown timer HH:MM:SS (must use urlencoded colons - %3A
COOK_TIMER	Same as above - looks like you need to set both to keep changes across refresh?
COOKHOLD	Cook and hold target temp in current units if timer is set to HOLD
TIME-OUT_ACTION	What to do when timer hits 00:00:00 (0: No Action, 1: HOLD, 2: Alarm, 3:Shutdown) See 8.3.2 in manual
ALARMDEV	Alarm deviation setpoint in current units (see 8.3.3 in Manual)
COOK_RAMP	Which probe to use for Ramp mode (0: Off, 1: Food 1, 2: Food 2, 3:Food 3)
OPENDETECT	Enable/Disable Lid Open detection (0: Off, 1:On)
CYCTIME	Fan Cycle time in s (between 4 and 10 seconds)
PROPBAND	Proportional band size (between 5-100 degF)
MENU_SCROLLING	Enable/Disable LCD scrolling (0: Off, 1:On)
LCD_BACKLIGHT	Enable/Disable LCD backlight (0: Off, 1:On)
LCD_CONTRAST	Contrast percent?
DEG_UNITS	Master Switch for degC/degF (0:degC, 1:degF)
ALARM_BEEPS	Alarm beeps (0-5)
KEY_BEEPS	Enable/Disable key beeps (0: Off, 1:On)

Keyword arguments: <*dictionary*> Dictionary of values to be updated. Note: will be validated against list of known values

Returns: <*Boolean*> True if successful / False if not successful

Example Usage:

```
cqi.sendUpdate({'FOOD1_NAME' : "Tri-Tip Roast",
                'FOOD1_SET': '140',
                'COOK_SET' : '300'})
```

statusLookup(*code*)

Provides a text meaning for a given status code "status" : ["OK", "HIGH", "LOW", "DONE", "ERROR", "HOLD", "ALARM",

"SHUTDOWN"],

Value from API	String Value
0	OK
1	HIGH
2	LOW
3	DONE
4	ERROR
5	HOLD
6	ALARM
7	SHUTDOWN

Keyword arguments: <int> code - the code returned in the object or the XML.

Returns: <String> Meaning behind the code

Example Usage: cqi.statusLookup(cqi.getConfig().FOOD1_STATUS)

temperatureLookup(*code*)

Provides a text meaning for the temperature scale in use temperature" : ["CELSIUS", "FAHRENHEIT"]

Value from API	String Value
0	CELSIUS
1	FAHRENHEIT

Keyword arguments: <int> code - the code returned in the object or XML.

Returns: <String> 'Celsius' or 'Fahrenheit'

Example Usage: cqi.temperatureLookup(cqi.getStatus().DEG_UNITS)

1.1.1 Inheritance

1.2 Exceptions

Exceptions for use with libary for BBQ Guru's CyberQ Temperature Control System

ChangeLog: A00 - 11/04/2012 - Bryan Kemp - Initial release

exception cyberqinterface_exceptions.**LookupException**(*message=None*, *errors=None*)

The lookup failed

exception cyberqinterface_exceptions.**ParameterValidationException**(*message=None*, *errors=None*)

An invalid parameter was passed to the interface

exception cyberqinterface_exceptions.**ResponseHTTPException**(*message=None*, *errors=None*)

The HTTP response was invalid

exception cyberqinterface_exceptions.**ResponseValidationException**(*message=None*, *errors=None*)

The response from the CyberQ web service was invalid.

1.3 XMLs

- **Status**

```
<nutcstatus>
   <!--all temperatures are displayed in tenths F, regardless of setting of unit-->
   <!--all temperatures sent by browser to unit should be in F.  you can send tenths F with a decimal
   <OUTPUT_PERCENT>100</OUTPUT_PERCENT>
   <TIMER_CURR>00:00:00</TIMER_CURR>
   <COOK_TEMP>3343</COOK_TEMP>
   <FOOD1_TEMP>823</FOOD1_TEMP>
   <FOOD2_TEMP>OPEN</FOOD2_TEMP>
   <FOOD3_TEMP>OPEN</FOOD3_TEMP>
   <COOK_STATUS>0</COOK_STATUS>
   <FOOD1_STATUS>0</FOOD1_STATUS>
   <FOOD2_STATUS>4</FOOD2_STATUS>
   <FOOD3_STATUS>4</FOOD3_STATUS>
   <TIMER_STATUS>0</TIMER_STATUS>
   <DEG_UNITS>1</DEG_UNITS>
   <COOK_CYCTIME>6</COOK_CYCTIME>
   <COOK_PROPBAND>500</COOK_PROPBAND>
   <COOK_RAMP>0</COOK_RAMP>
</nutcstatus>
```

- **All**

```
<nutcallstatus>
    <!--this is similar to status.xml, but with more values-->
    <!--all temperatures are displayed in tenths F, regardless of setting of unit-->
    <!--all temperatures sent by browser to unit should be in F.  you can send tenths F with a decima
    <COOK>
        <COOK_NAME>Big Green Egg</COOK_NAME>
        <COOK_TEMP>3216</COOK_TEMP>
        <COOK_SET>4000</COOK_SET>
        <COOK_STATUS>0</COOK_STATUS>
    </COOK>
    <FOOD1>
        <FOOD1_NAME>Chicken Quarters</FOOD1_NAME>
        <FOOD1_TEMP>1482</FOOD1_TEMP>
        <FOOD1_SET>1750</FOOD1_SET>
        <FOOD1_STATUS>0</FOOD1_STATUS>
    </FOOD1>
    <FOOD2>
        <FOOD2_NAME>Food2</FOOD2_NAME>
        <FOOD2_TEMP>OPEN</FOOD2_TEMP>
        <FOOD2_SET>1000</FOOD2_SET>
        <FOOD2_STATUS>4</FOOD2_STATUS>
    </FOOD2>
    <FOOD3>
        <FOOD3_NAME>Food3</FOOD3_NAME>
        <FOOD3_TEMP>OPEN</FOOD3_TEMP>
        <FOOD3_SET>1000</FOOD3_SET>
        <FOOD3_STATUS>4</FOOD3_STATUS>
    </FOOD3>
    <OUTPUT_PERCENT>100</OUTPUT_PERCENT>
    <TIMER_CURR>00:00:00</TIMER_CURR>
    <TIMER_STATUS>0</TIMER_STATUS>
    <DEG_UNITS>1</DEG_UNITS>
    <COOK_CYCTIME>6</COOK_CYCTIME>
    <COOK_PROPBAND>500</COOK_PROPBAND>
    <COOK_RAMP>0</COOK_RAMP>
</nutcallstatus>
```

- **Configuration**


```
<nutcallstatus>
   <!--this is similar to all.xml, but with more values-->
   <!--all temperatures are displayed in tenths F, regardless of setting of unit-->
   <!--all temperatures sent by browser to unit should be in F.  you can send tenths F with a decimal
   <COOK>
      <COOK_NAME>Big Green Egg</COOK_NAME>
      <COOK_TEMP>3220</COOK_TEMP>
      <COOK_SET>4000</COOK_SET>
      <COOK_STATUS>0</COOK_STATUS>
   </COOK>
   <FOOD1>
      <FOOD1_NAME>Chicken Quarters</FOOD1_NAME>
      <FOOD1_TEMP>1493</FOOD1_TEMP>
      <FOOD1_SET>1750</FOOD1_SET>
      <FOOD1_STATUS>0</FOOD1_STATUS>
   </FOOD1>
   <FOOD2>
      <FOOD2_NAME>Food2</FOOD2_NAME>
      <FOOD2_TEMP>OPEN</FOOD2_TEMP>
      <FOOD2_SET>1000</FOOD2_SET>
      <FOOD2_STATUS>4</FOOD2_STATUS>
   </FOOD2>
   <FOOD3>
      <FOOD3_NAME>Food3</FOOD3_NAME>
      <FOOD3_TEMP>OPEN</FOOD3_TEMP>
      <FOOD3_SET>1000</FOOD3_SET>
      <FOOD3_STATUS>4</FOOD3_STATUS>
   </FOOD3>
   <OUTPUT_PERCENT>100</OUTPUT_PERCENT>
   <TIMER_CURR>00:00:00</TIMER_CURR>
   <TIMER_STATUS>0</TIMER_STATUS>
   <SYSTEM>
      <MENU_SCROLLING>1</MENU_SCROLLING>
      <LCD_BACKLIGHT>47</LCD_BACKLIGHT>
      <LCD_CONTRAST>10</LCD_CONTRAST>
      <DEG_UNITS>1</DEG_UNITS>
      <ALARM_BEEPS>0</ALARM_BEEPS>
      <KEY_BEEPS>0</KEY_BEEPS>
   </SYSTEM>
   <CONTROL>
      <TIMEOUT_ACTION>0</TIMEOUT_ACTION>
      <COOKHOLD>2000</COOKHOLD>
      <ALARMDEV>500</ALARMDEV>
      <COOK_RAMP>0</COOK_RAMP>
      <OPENDETECT>1</OPENDETECT>
      <CYCTIME>6</CYCTIME>
      <PROPBAND>500</PROPBAND>
   </CONTROL>
   <WIFI>
      <IP>10.0.1.30</IP>
      <NM>255.255.255.0</NM>
      <GW>10.0.1.1</GW>
      <DNS>10.0.1.1</DNS>
      <WIFIMODE>0</WIFIMODE>
      <DHCP>0</DHCP>
      <SSID>Kempville Network</SSID>
      <WIFI_ENC>6</WIFI_ENC>
```

```
      <WIFI_KEY>Nemesis69!</WIFI_KEY>
      <HTTP_PORT>80</HTTP_PORT>
   </WIFI>
   <SMTP>
      <SMTP_HOST>smtp.hostname.com</SMTP_HOST>
      <SMTP_PORT>0</SMTP_PORT>
      <SMTP_USER></SMTP_USER>
      <SMTP_PWD></SMTP_PWD>
      <SMTP_TO>destination@someplace.com</SMTP_TO>
      <SMTP_FROM>source@someplace.com</SMTP_FROM>
      <SMTP_SUBJ>Temperature Controller Status E-Mail</SMTP_SUBJ>
      <SMTP_ALERT>0</SMTP_ALERT>
   </SMTP>
</nutcallstatus>
```

CHAPTER

TWO

BACKGROUND

I wrote the CyberQInterface because I love to barbeque. I grew up near Memphis, Tn which is the home of some darn fine BBQ. The preferred style of BBQ made in my homeland is pulled pork sandwiches with coleslaw and a thin, pepper hot sauce. Pulled Pork is generally made with hickory smoke.

I have been in Texas (another area with a strong BBQ heritage) since 1999 and given that beef smoked with mesquite is the local style I learned a lot about the 'smoking' process.

I have gone through many grills and smokers over the years. In 2010 I purchased one of the most popular cooking systems on the market. I invested in a Large Big Green Egg. http://www.biggreenegg.com This is not the place to expound on the qualities of the grill/smoker, but I must admit that having the 'Egg' has made me a better cook and I could not be happier.

One of the best things about the Egg is the consistency of temperature once the Egg is in the correct configuation for a particular temperature. That coupled with the large fuel capacity means that you can cook items for 18 or more hours without touching the smoker.

However, given the amount of time it takes to smoke a brisket, the food temperature does require a bit of monitoring. The mechanical temperature guage on my BGE failed and the actual pit temperature was approximately 50 degrees warmer than indicated. Needless to say, by the time I had figured out the problem my wife had grown tired of making 'expensive' beef jerky.

She encouraged me to purchase the CyberQ Wifi from BBQ Guru. http://store.thebbqguru.com/weborderentry/CyberQ%20WiFi The CyberQ includes a web service interface which not only allows reading data from the temperature control, but the ability to change settings via the web service. Ultimately, I wrote this library to make that communication process easier and to be able to trigger events programmatically.

CHAPTER

THREE

INSTALLATION

You can install the library via easy_install. The command should look like:

```
sudo pip install CyberQInterface
```

Source is also available on github. http://github.com/TheBrilliantIdea/CyberQInterface

To install from source, clone the code from github and then execute the setup.py installer:

```
git clone git://github.com/thebrilliantidea/CyberQInterface.git
cd CyberQInterface
sudo python setup.py install
```

CHAPTER

FOUR

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

C

cyberqinterface, **??**
cyberqinterface_exceptions, **??**