HWControl Dummy

Was ist der HWControl Dummy?

Der HWControl Dummy ist für zum Testen des DBDaemons enstanden. Er imitiert die Funktionalität die die HWControl und deren angesprochenen Wrappern erfüllt. Der Dummy liefert, bei Anfragen, dabei simple Beispiel- bzw. selbsterstellte Werte zurück. Dabei wird der Dummy wie die richtige HWControl über die Funktion hwcontrol(dict_, *args) angesprochen.

Als Beispielparameter wird ein Currencost Subdevice angesprochen, welches den aktuellen Wert in einem Dictionary liefern soll.

```
hwcontrol(True, {‘device’: CurrentCost, ‘subdev’:’1_ch9’, ‘action’:’get_state’}) # 1_ch9 ist dabei ein Beispiel Gerät
```

Rückgabewert:

```
[[{‘device’:’CurrentCost’, ‘subdev’:’1_ch9’, ‘power’:’1337’, ‘timestamp’:’1234567890’}]]
```

Code

```
# -*- coding: utf-8 -*-



import logging

from sys import stderr

from time import gmtime, mktime



from lxml import etree



import config

import utils









logger = logging.getLogger(__name__)



# Devices

Arduino = { 'device' : 'Arduino' , 'subdev' : 'Arduino' , 'temp' : '24' }

Twitter = { 'device' : 'Twitter' , 'subdev' : 'Twitter' , 'message' : 'foo'}
```

```
# GoogleWeather
 GoogleWeather = {'device':'GoogleWeather' , 'subdev' : 'GoogleWeather',
'condition': 'sonnig', 'temp': '23.4','humidity':'27', 'wind_condition':'ruhig'}
# CurrentCost
CurrentCost = {
# Waschmaschine
'3_ch7' : {'device' : 'CurrentCost' , 'subdevice' : '3_ch7' ,
          'power': '2500'} ,
# Computer
'4_ch2' : {'device' : 'CurrentCost' , 'subdevice' : '4_ch2' ,
          'power': '42'} ,
# Kuehlschrank
'1_ch9' : {'device' : 'CurrentCost' , 'subdevice' : '1_ch9' ,
          'power': '1337'}}

CCToken = { '3_ch7' : False , '4_ch2': False , '1_ch9' : False}

# HomeMatic
HomeMatic = {
# Wetterstationen
'IEQ0007536:0': [{'unreach': 'False', 'subdev': 'IEQ0007536:0',
'timestamp': '1304749563.0', 'config_pending': 'False', 'lowbat': 'False',
'sticky_unreach': 'False', 'aes_key': '0', 'device': 'HomeMatic',
'dutycycle': 'False'}] ,
'IEQ0007536:1': [{'inhibit': 'False', 'working': 'False',
'subdev':'IEQ0007536:1', 'on_time': '0.0', 'timestamp': '1304749563.0',
'state': 'False','device': 'HomeMatic'}],
# Heizungssteuerungen
'HEQ0083602:0' : [{'unreach': 'False', 'subdev': 'HEQ0083602:0',
'lowbat': 'False','timestamp': '1304749561.0', 'config_pending': 'False',
'sticky_unreach': 'False', 'device': 'HomeMatic'}] ,
'HEQ0083602:1':[{'device': 'HomeMatic', 'timestamp': '1304749561.0',
'valve_state': '0', 'subdev': 'HEQ0083602:1', 'error': '0'}],
```

```
# Steckdoseen
'HEQ0105569:0' : [{'unreach': 'False', 'subdev': 'HEQ0105569:0',
'lowbat': 'False', 'timestamp': '1304749562.0', 'config_pending': 'False',
'sticky_unreach': 'False', 'device': 'HomeMatic'}],
'HEQ0105569:1':[{'device': 'HomeMatic', 'timestamp': '1304749562.0',
'humidity': '38' , 'subdev': 'HEQ0105569:1', 'temperature': '22.4'}] }


def hwcontrol(dict_, *args):
   """
    Kontrolle der Uebergabeparameter
    Dictionary oder XML
    """
   logger.debug('CALL: hwcontrol(%s, %s)', dict_, args)
   response = []
   for request in args:
       ret = None
       try:
           # dictionary
           device = request.pop('device')
           action = request.pop('action')
       except AttributeError as e:
           # vermutlich XML
           try:
               tree = etree.fromstring(data)
               if tree.tag != 'request':
                   raise Exception('missing request tag')
           except Exception as e:
               print >> stderr, e
               logger.error('Eingabe konnte nicht verarbeitet werden: %s', e)
           else:
               response_tree = etree.Element('response')
               for elem in tree.findall('device/action'):
                   device = elem.getparent().get('name')
                   action = elem.get('name')
```

```
                    args = {}
                    for param in elem.iterchildren(tag='parameter'):
                        if param.get('name') != None:
                            args[param.get('name')] = param.text
                    ret = process(device, action, dict_=False, **args)
                    logger.info('call ausgefuehrt')
                    response_tree.append(ret)
            finally:
                ret = response_tree
        except Exception as e:
            print >> stderr, e
            logger.error('Unerwarteter Fehler: %s', e)
        else:
            ret = process(device, action, dict_, **request)
            logger.info('call ausgefuehrt')
        finally:
            if ret != None:
                response.append(ret)
    return response



def process(device, action, dict_=True, **kwargs):
    """
     Prueft die gewuenschte Aktion nach und uebergibt an die Hilfsfunktionen
     """
    logger.debug('CALL: process(%s, %s, %s, %s)', device, action, dict_,
            kwargs.items())
    ret = None
    try:
        if action not in ['get_state', 'set_state', 'get_devices']:
            raise Exception('unknown action')
    except Exception as e:
        print >> stderr, e
```

```
        logger.error('action konnte nicht ausgefuehrt werden: %s', e)
    else:
        if action == 'get_state':
            logger.info('action = get_state')
            ret = _get_state(device, **kwargs)
        elif action == 'get_devices':
            logger.info('action = get_device')
            ret = _get_devices(device, **kwargs)
        elif action == 'set_state':
            logger.info('action = set_state')
            ret = _set_state(device, **kwargs)
    finally:
        return ret



def _get_state(device, **kwargs):
    """
     Hilfsfunktion, welche den Wert eines devices,subdevices liefert.
     """
    logger.debug('CALL: _get_state(%s, %s)', device, kwargs.items())

    subdev = kwargs.values()[0]
    value = None
    if kwargs.keys() == ['subdev']:
        logger.info('subdev vorhanden')
        # subdev = Arduino
        if kwargs.values() == ['Arduino']:
            Arduino.update({'timestamp':mktime(gmtime())})
            value = Arduino
            logger.info('Arduino, value = %s', value)
        # subdev = GoogleWeather
        elif kwargs.values() == ['GoogleWeather']:
            GoogleWeather.update({'timestamp' : mktime(gmtime())})
            value = GoogleWeather
```

```
            logger.info('GoogleWeather, value = %s', value)
        # subdev = Twitter
        elif kwargs.values() == ['Twitter']:
            Twitter.update({'timestamp' : mktime(gmtime())})
            value = Twitter
            logger.info('Twitter, value = %s', value)
        # subdev = HomeMatic
        elif device == 'HomeMatic':
            if subdev in HomeMatic.keys():
                HomeMatic[subdev][0].update({'timestamp':mktime(gmtime())})
                value = HomeMatic[subdev][0]
                logger.info('HomeMatic, value = %s', value)
        # subdev = CurrentCost
        elif device == 'CurrentCost':
            value = []
            if CCToken['3_ch7'] == True:
                CurrentCost['3_ch7'].update({'timestamp':mktime(gmtime())})
                value.append(CurrentCost['3_ch7'])
                CCToken.update({'3_ch7':False})
            if CCToken['4_ch2'] == True:
                CurrentCost['4_ch2'].update({'timestamp':mktime(gmtime())})
                value.append(CurrentCost['4_ch2'])
                CCToken.update({'4_ch2' : False})
            if CCToken['1_ch9'] == True:
                CurrentCost['1_ch9'].update({'timestamp':mktime(gmtime())})
                value.append(CurrentCost['1_ch9'])
                CCToken.update({'1_ch9' : False})
            logger.info('CurrentCost, value = %s', value)
    else:
        logger.info('Kein subdev angegeben')
        value = None
    return value
```

```
def _get_devices(device, **kwargs):
    """
     Hilfsfunktion, welche eine Liste von Subdevices liefert.
     """
    logger.debug('CALL: _get_devices(%s, %s)', device, kwargs.items())
    # device = HomeMatic
    if device == 'HomeMatic':
        logger.info('HomeMatic Liste erstellt')
        return HomeMatic.keys()
    # device = CurrentCost
    elif device == 'CurrentCost':
        logger.info('CurrentCost Liste erstellt')
        return CurrentCost.keys()
    # device = Arduino
    elif device == 'Arduino':
        logger.info('Arduino Liste erstellt')
        return Arduino['subdev']
    # device = GoogleWeather
    elif device == 'GoogleWeather':
        logger.info('GoogleWeather Liste erstellt')
        return GoogleWeather['subdev']
    # device = Twitter
    elif device == 'Twitter':
        logger.info('Twitter Liste erstellt')
        return Twitter['subdev']
    else:
        return 'Falsche Eingabe'



def _set_state(device, **kwargs):
    """
     Hilfsfunktion, Geraeteeigenschaften werden veraendert
     """
```

```
logger.debug('CALL: _set_state(%s, %s)', device, kwargs.items())

try:
    subdevice = kwargs['subdev']
    del kwargs['subdev']
    logger.debug('subdev = %s', subdevice)

    key = kwargs.keys()[0]
    value = kwargs.values()[0]

    # CurrentCost
    if device == 'CurrentCost' and key == 'power':
        CurrentCost[subdevice].update({key:value})
        CCToken.update({subdevice: True})
        logger.info('CC geaendert: %s ist nun %s', key, value)
    # HomeMatic
    elif device == 'HomeMatic':
        if key in HomeMatic[subdevice][0].keys():
            HomeMatic[subdevice][0].update({key:value})
        elif key in HomeMatic[subdevice][1].keys():
            HomeMatic[subdevice][1].update({key:value})
        logger.info('HomeMatic geandert: %s, value %s', key, value)
    # Falsche Eingabe
    else:
        print 'Falsche Eingabe'
        logger.info('Keine Aenderung')
except:
    subdevice = None

    key = kwargs.keys()[0]
    value = kwargs.values()[0]

    # Arduino
```

```
        if device == 'Arduino' and key == 'temp':
            Arduino.update({key:value})
            logger.info('Arduino: %s ist nun %s', key, value)
        # Twitter
        elif device == 'Twitter' and key == 'message':
            Twitter.update({key:value})
            logger.info('Twitter: %s ist nun %s', key, value)
        # GoogleWeather
        elif device == 'GoogleWeather' and key in GoogleWeather.keys():
            GoogleWeather.update({key:value})
            logger.info('GoogleWeather geaendert: %s ist nun %s', key, value)
        else:
            print 'Falsche Eingabe'
            logger.info('Keine Aenderung')
    return None

# vim: set sts=4 sw=4 et:
```