PowerMeter daemon settings

Generated by Doxygen 1.7.6.1

Tue Apr 10 2012 15:55:44

Contents

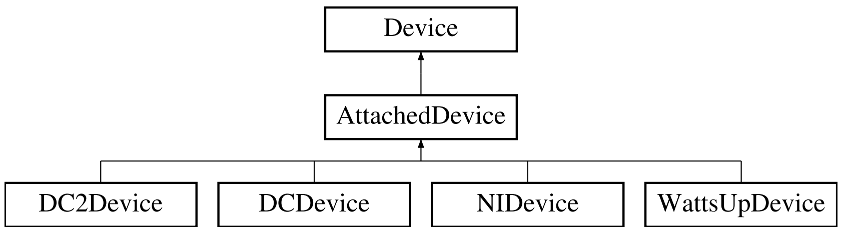
Chapter 1

Data Structure Documentation

1.1 AttachedDevice Class Reference

A device attached to a computer description.

Inheritance diagram for AttachedDevice:



Public Member Functions

- def __init__

  Creates device attached to a computer description and adds it to the devices dictionary.

Data Fields

- computer

1.1.1 Detailed Description

A device attached to a computer description.

Definition at line 148 of file settings_classes.py.

1.1.2 Constructor & Destructor Documentation

1.1.2.1 def __init__ (*self*, *name*, *computer*, *url*, *max_frequency*)

Creates device attached to a computer description and adds it to the devices dictionary.

Before adding the given device description to the devices dictionary, it checks that the name of the new device has not been used by a previously added device.

Parameters

`in`	*name*	The device name (used for identification, must be unique)
`in`	*computer*	The computer the device is attached to
`in`	*url*	The url of this device
`in`	*max_-frequency*	The maximum sample frequency of the device

Reimplemented in WattsUpDevice, NIDevice, DC2Device, and DCDevice.

Definition at line 162 of file settings_classes.py.

1.1.3 Field Documentation

1.1.3.1 computer

Definition at line 162 of file settings_classes.py.

The documentation for this class was generated from the following file:

- /home/barrachi/datos/aplicaciones/powermeter/settings_classes.py

1.2 Computer Class Reference

A computer description.

Public Member Functions

- def __init__

  Creates a computer description and adds it to the computers dictionary.
- def __repr__

  Returns a string representation for this computer.
- def add

  Adds a device description to the computer.

Data Fields

- name
- ip
- devices

1.2.1 Detailed Description

A computer description.

Definition at line 20 of file settings_classes.py.

1.2.2 Constructor & Destructor Documentation

1.2.2.1 def __init__ (*self*, *name*, *ip*)

Creates a computer description and adds it to the computers dictionary.

Parameters

`in`	*name*	The name of the computer
`in`	*ip*	The IP address of the computer

Definition at line 28 of file settings_classes.py.

1.2.3 Member Function Documentation

1.2.3.1 def __repr__ (*self*)

Returns a string representation for this computer.

Definition at line 36 of file settings_classes.py.

1.2.3.2 def add (*self*, *device*)

Adds a device description to the computer.

Parameters

`in`	*device*	A device description object

Definition at line 43 of file settings_classes.py.

1.2.4 Field Documentation

1.2.4.1 devices

Definition at line 28 of file settings_classes.py.

1.2.4.2 ip

Definition at line 28 of file settings_classes.py.

1.2.4.3 name

Definition at line 28 of file settings_classes.py.

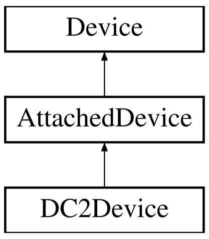
The documentation for this class was generated from the following file:

- /home/barrachi/datos/aplicaciones/powermeter/settings_classes.py

1.3 DC2Device Class Reference

A DC2Meter device description.

Inheritance diagram for DC2Device:



Public Member Functions

- def __init__

  Creates a DC2Meter device description and adds it to the devices dictionary.

1.3.1 Detailed Description

A DC2Meter device description.

Definition at line 191 of file settings_classes.py.

1.3.2 Constructor & Destructor Documentation

1.3.2.1 def __init__ (*self*, *name*, *computer*, *url*, *max_frequency*)

Creates a DC2Meter device description and adds it to the devices dictionary.

Parameters

`in`	*name*	The device name (used for identification, must be unique)
`in`	*computer*	The computer the device is attached to
`in`	*url*	The url of this device
`in`	*max_-frequency*	The maximum sample frequency of the device

Reimplemented from AttachedDevice.

Definition at line 201 of file settings_classes.py.

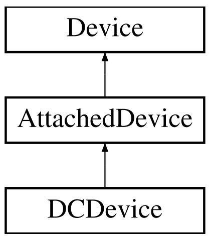
The documentation for this class was generated from the following file:

- /home/barrachi/datos/aplicaciones/powermeter/settings_classes.py

1.4 DCDevice Class Reference

A DCMeter device description.

Inheritance diagram for DCDevice:



Public Member Functions

- def __init__

  Creates a DC2Meter device description and adds it to the devices dictionary.

1.4.1 Detailed Description

A DCMeter device description.

Definition at line 174 of file settings_classes.py.

1.4.2 Constructor & Destructor Documentation

1.4.2.1 def __init__ (*self*, *name*, *computer*, *url*, *max_frequency*)

Creates a DC2Meter device description and adds it to the devices dictionary.

Parameters

in	*name*	The device name (used for identification, must be unique)
in	*computer*	The computer the device is attached to
in	*url*	The url of this device
in	*max_-frequency*	The maximum sample frequency of the device

Reimplemented from AttachedDevice.

Definition at line 184 of file settings_classes.py.

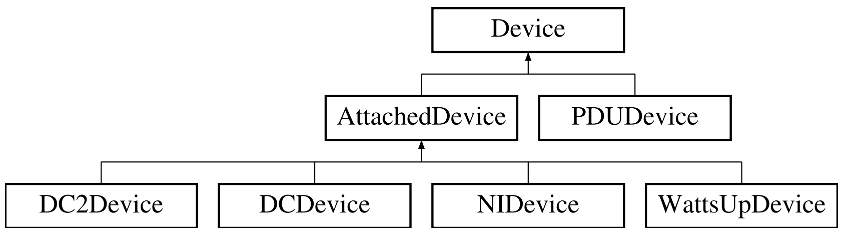
The documentation for this class was generated from the following file:

- /home/barrachi/datos/aplicaciones/powermeter/settings_classes.py

1.5 Device Class Reference

A device description.

Inheritance diagram for Device:



Public Member Functions

- def __init__

    Creates a device description and adds it to the devices dictionary.
- def __repr__

    Returns a string representation for this device.
- def add_line

    Adds a line description to the device.

Data Fields

- name
- url
- max_frequency
- lines

1.5.1 Detailed Description

A device description.

Definition at line 100 of file settings_classes.py.

1.5.2 Constructor & Destructor Documentation

1.5.2.1 def __init__ (*self*, *name*, *url*, *max_frequency*)

Creates a device description and adds it to the devices dictionary.

Before adding the given device description to the devices dictionary, it checks that the name of the new device has not been used by a previously added device.

Parameters

`in`	*name*	The device name (used for identification, must be unique)
`in`	*url*	The url of this device
`in`	*max_-frequency*	The maximum sample frequency of the device

Reimplemented in PDUDevice.

Definition at line 113 of file settings_classes.py.

1.5.3 Member Function Documentation

1.5.3.1 def __repr__ (*self*)

Returns a string representation for this device.

Definition at line 125 of file settings_classes.py.

1.5.3.2 def add_line (*self*, *name*, *voltage*, *description*)

Adds a line description to the device.

Before adding the given line description to the device, it checks that the name of the new line has not been used by a previously added line.

Parameters

`in`	*name*	The line name (used for identification)
`in`	*voltage*	The line voltage
`in`	*description*	A text description of the line

Reimplemented in WattsUpDevice.

Definition at line 138 of file settings_classes.py.

1.5.4 Field Documentation

1.5.4.1 lines

Definition at line 113 of file settings_classes.py.

1.5.4.2 max_frequency

Definition at line 113 of file settings_classes.py.

1.5.4.3 name

Definition at line 113 of file settings_classes.py.

1.5.4.4 url

Definition at line 113 of file settings_classes.py.

The documentation for this class was generated from the following file:

- /home/barrachi/datos/aplicaciones/powermeter/settings_classes.py

1.6 Line Class Reference

A line description.

Public Member Functions

- def __init__

  Creates a line description.
- def __repr__

  Returns a string representation for this line.

Data Fields

- name
- voltage
- description

1.6.1 Detailed Description

A line description.

Definition at line 53 of file settings_classes.py.

1.6.2 Constructor & Destructor Documentation

1.6.2.1 def __init__ (*self*, *name*, *voltage*, *description*)

Creates a line description.

Parameters

`in`	*name*	The line name (used for identification)
`in`	*voltage*	The line voltage
`in`	*description*	A text description of the line

Definition at line 61 of file settings_classes.py.

1.6.3 Member Function Documentation

1.6.3.1 def __repr__ (*self*)

Returns a string representation for this line.

Definition at line 67 of file settings_classes.py.

1.6.4 Field Documentation

1.6.4.1 description

Definition at line 61 of file settings_classes.py.

1.6.4.2 name

Definition at line 61 of file settings_classes.py.

1.6.4.3 voltage

Definition at line 61 of file settings_classes.py.

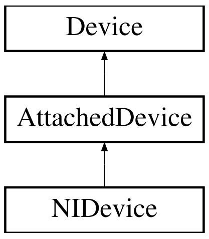
The documentation for this class was generated from the following file:

- /home/barrachi/datos/aplicaciones/powermeter/settings_classes.py

1.7 NIDevice Class Reference

A National Instruments device description.

Inheritance diagram for NIDevice:



Public Member Functions

- def __init__
  Creates a National Instruments device description and adds it to the devices dictionary.

1.7.1 Detailed Description

A National Instruments device description.

Definition at line 208 of file settings_classes.py.

1.7.2 Constructor & Destructor Documentation

1.7.2.1 def __init__ (*self, name, computer, url, max_frequency*)

Creates a National Instruments device description and adds it to the devices dictionary.

Parameters

`in`	*name*	The device name (used for identification, must be unique)
`in`	*computer*	The computer the device is attached to
`in`	*url*	The url of this device
`in`	*max_-frequency*	The maximum sample frequency of the device

Reimplemented from AttachedDevice.

Definition at line 218 of file settings_classes.py.

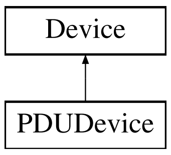
The documentation for this class was generated from the following file:

- /home/barrachi/datos/aplicaciones/powermeter/settings_classes.py

1.8 PDUDevice Class Reference

A PDU device description.

Inheritance diagram for PDUDevice:



Public Member Functions

- def __init__

  Creates a PDU device description and adds it to the devices dictionary.
- def add_line

  Adds a pdu line description to the device.

1.8.1 Detailed Description

A PDU device description.

Definition at line 255 of file settings_classes.py.

1.8.2 Constructor & Destructor Documentation

1.8.2.1 def __init__ (*self, name, url, max_frequency*)

Creates a PDU device description and adds it to the devices dictionary.

Parameters

`in`	*name*	The device name (used for identification, must be unique)
`in`	*url*	The url of this device
`in`	*max_-frequency*	The maximum sample frequency of the device

Reimplemented from Device.

Definition at line 264 of file settings_classes.py.

1.8.3 Member Function Documentation

1.8.3.1 def **add_line** (*self,* *name,* *computer,* *voltage,* *description* = " ")

Adds a pdu line description to the device.

Before adding the given line description to the device, it checks that the name of the new line has not been used by a previously added line.

Parameters

in	*name*	The line name (used for identification)
in	*computer*	The computer the line is attached to
in	*voltage*	The line voltage
in	*description*	An optional text description of the line

Definition at line 279 of file settings_classes.py.

The documentation for this class was generated from the following file:

- /home/barrachi/datos/aplicaciones/powermeter/settings_classes.py

1.9 PDULine Class Reference

A PDU line description.

Public Member Functions

- def __init__

  Creates a PDU line description.
- def __repr__

  Returns a string representation for this line.

Data Fields

- name
- computer
- voltage
- description

1.9.1 Detailed Description

A PDU line description.

Definition at line 74 of file settings_classes.py.

1.9.2 Constructor & Destructor Documentation

1.9.2.1 def __init__ (*self*, *name*, *computer*, *voltage*, *description* = " ")

Creates a PDU line description.

Parameters

`in`	*name*	The line name (used for identification)
`in`	*computer*	The computer this PDU line is attached to
`in`	*voltage*	The line voltage
`in`	*description*	An optional text description of the line

Definition at line 83 of file settings_classes.py.

1.9.3 Member Function Documentation

1.9.3.1 def __repr__ (*self*)

Returns a string representation for this line.

Definition at line 93 of file settings_classes.py.

1.9.4 Field Documentation

1.9.4.1 computer

Definition at line 83 of file settings_classes.py.

1.9.4.2 description

Definition at line 83 of file settings_classes.py.

1.9.4.3 name

Definition at line 83 of file settings_classes.py.

1.9.4.4 voltage

Definition at line 83 of file settings_classes.py.

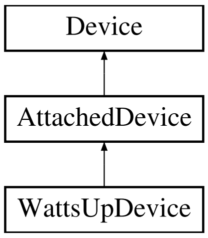
The documentation for this class was generated from the following file:

- /home/barrachi/datos/aplicaciones/powermeter/settings_classes.py

1.10 WattsUpDevice Class Reference

A WattsUp device description.

Inheritance diagram for WattsUpDevice:



Public Member Functions

- def __init__

  Creates a WattsUp device description and adds it to the devices dictionary.
- def add_line

  Fake adding of line description to the device.

1.10.1 Detailed Description

A WattsUp device description.

Definition at line 225 of file settings_classes.py.

1.10.2 Constructor & Destructor Documentation

1.10.2.1 def __init__ (*self*, *name*, *computer*, *url*, *max_frequency*)

Creates a WattsUp device description and adds it to the devices dictionary.

Parameters

`in`	*name*	The device name (used for identification, must be unique)
`in`	*computer*	The computer the device is attached to
`in`	*url*	The url of this device
`in`	*max_-frequency*	The maximum sample frequency of the device

Reimplemented from AttachedDevice.

Definition at line 235 of file settings_classes.py.

1.10.3 Member Function Documentation

1.10.3.1 def add_line (*self*, *name*, *voltage*, *description*)

Fake adding of line description to the device.

The WattsUp Device does not have lines. This method avoids the base class method silently been called.

Parameters

`in`	*name*	The line name (used for identification)
`in`	*voltage*	The line voltage
`in`	*description*	A text description of the line

Reimplemented from Device.

Definition at line 247 of file settings_classes.py.

The documentation for this class was generated from the following file:

- /home/barrachi/datos/aplicaciones/powermeter/settings_classes.py

Chapter 2

File Documentation

2.1 /home/barrachi/datos/aplicaciones/powermeter/settings-use-example.py File Reference

Functions

- def **header**
- def **main**

  Shows the information given by the settings file.

2.2 /home/barrachi/datos/aplicaciones/powermeter/settings-use-example.py

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

#=====================================================================
# PowerMeter daemon settings use example
#=====================================================================

## Read settings
import settings

def header(txt):
    print "------------------------------------------------------------"
    print "{0:^60}".format(txt)
    print "------------------------------------------------------------"


## Shows the information given by the settings file
def main():

    # Show all the computers
    header("Computers")
    for computer in settings.computers.values():
        print computer
    print
    print

    # Show all the devices
    header("Devices")
    for device in settings.devices.values():
        print device
    print
    print

    # Access the devices of a given computer
    header("Devices in computer 'lorca'")
    for device in settings.computers['lorca'].devices.values():
        print device
    print
    print

    header("Devices in computer 'matserv'")
    for device in settings.computers['matserv'].devices.values():
        print device
    print
    print

    # Access to which computer a given device is attached to
    header("Computer to which device 'DCMeter' is attached")
    print settings.devices["DCMeter"].computer
    print
    print

    # Access the lines of a given device
    header("Lines in device 'DCMeter'")
    for line in settings.devices["DCMeter"].lines.values():
        print line
    print
    print

    header("Lines in device 'PDU'")
    for line in settings.devices["PDU"].lines.values():
        print line
    print
    print

    # Access individual properties of a given line of a given device
    header("Properties of line 'DC 12V' of device 'DCMeter'")
    line=settings.devices["DCMeter"].lines["DC 12V"]
    print "Line name: {0}".format(line.name)
    print "Line voltage: {0}".format(line.voltage)
    print "Line description: '{0}'".format(line.description)
    print


if __name__=="__main__":
    main()
```

2.3 /home/barrachi/datos/aplicaciones/powermeter/settings.py File Reference

Variables

- int **PORT** = 6526
- string **LOGFILENAME** = "/var/log/powermeter.log"
- tuple **comp1** = Computer(name="lorca", ip="150.128.83.25")
- tuple **comp2** = Computer(name="matserv", ip="150.128.83.35")
- tuple **dev1** = DCDevice(name="DCMeter", computer=comp1, url="file-://dev/usb0", max_frequency=25)
- tuple **dev2** = DC2Device(name="DCMeter2", computer=comp1, url="file-://dev/usb1", max_frequency=1000)

- tuple **dev3** = WattsUpDevice(name="WattsUp", computer=comp2, url="file-://dev/usb3", max_frequency=50)
- tuple **dev4** = PDUDevice(name="PDU", url="ssh://user:pass@matserv.uji.es", max_frequency=1000)

2.4 /home/barrachi/datos/aplicaciones/powermeter/settings.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

#=======================================================================
# PowerMeter daemon settings
#=======================================================================

from settings_classes import computers, devices, Computer
from settings_classes import DCDevice, DC2Device, NIDevice, WattsUpDevice,
PDUDevice

#-----------------------------------------------------------------------
# General section
#-----------------------------------------------------------------------

# Port in which the daemon will be listening (default: 6526)
PORT=6526

# Log file name (default: "/var/log/powermeter.log")
LOGFILENAME="/var/log/powermeter.log"

#-----------------------------------------------------------------------
# Computers section
#-----------------------------------------------------------------------

comp1=Computer(name="lorca", ip="150.128.83.25")
comp2=Computer(name="matserv", ip="150.128.83.35")

#-----------------------------------------------------------------------
# Devices section
#-----------------------------------------------------------------------

dev1=DCDevice(name="DCMeter", computer=comp1, url="file://dev/usb0",
max_frequency=25)
dev1.add_line(name="DC 12V", voltage=12, description="A 12V power line")
dev1.add_line(name="DC 3V", voltage=3, description="A 3V power line")

dev2=DC2Device(name="DCMeter2", computer=comp1, url="file://dev/usb1",
max_frequency=1000)
dev2.add_line(name="DC2 12V", voltage=12, description="A 12V power line")
dev2.add_line(name="DC2 3V", voltage=3, description="A 3V power line")

dev2=NIDevice(name="National Instruments", computer=comp1, url="file://dev/usb2
", max_frequency=1000)
dev2.add_line(name="DC2 12V", voltage=12, description="A 12V power line")
dev2.add_line(name="DC2 3V", voltage=3, description="A 3V power line")

dev3=WattsUpDevice(name="WattsUp", computer=comp2, url="file://dev/usb3",
max_frequency=50)

dev4=PDUDevice(name="PDU", url="ssh://user:pass@matserv.uji.es", max_frequency=
1000)
dev4.add_line(name="PDU lorca", computer=comp1, voltage=220, description="lorca
 watts")
dev4.add_line(name="PDU matserv", computer=comp2, voltage=220, description="
matserv watts")
```

2.5 /home/barrachi/datos/aplicaciones/powermeter/settings_-classes.py File Reference

Data Structures

- class Computer

  A computer description.
- class Line

  A line description.
- class PDULine

  A PDU line description.
- class Device

  A device description.
- class AttachedDevice

  A device attached to a computer description.
- class DCDevice

  A DCMeter device description.
- class DC2Device

  A DC2Meter device description.
- class NIDevice

  A National Instruments device description.
- class WattsUpDevice

  A WattsUp device description.
- class PDUDevice

  A PDU device description.

Variables

- dictionary **devices** = {}

  Dictionary of devices.
- dictionary **computers** = {}

  Dictionary of computers.

2.6 /home/barrachi/datos/aplicaciones/powermeter/settings_-classes.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

#====================================================================
# PowerMeter daemon setting classes
#
# This module defines the classes and dictionaries used by settings.py
#====================================================================


## Dictionary of devices
devices={}

## Dictionary of computers
computers={}


## A computer description
#
class Computer(object):

    ## Creates a computer description and adds it to the computers
    ## dictionary
    #
    # @param [in] name  The name of the computer
    # @param [in] ip    The IP address of the computer
    #
    def __init__(self, name, ip):
        self.name=name
        self.ip=ip
        self.devices={}
        # Register the computer
        computers[name]=self

    ## Returns a string representation for this computer
    def __repr__(self):
        return "Computer {0} ({1}): {2} device(s)".format(self.name, self.ip, len(self.devices))

    ## Adds a device description to the computer
    #
    # @param [in] device  A device description object
    #
    def add(self, device):
        if not isinstance(device, Device):
            msg="the given device parameter is not a Device object"
            raise SyntaxError, msg
        self.devices[device.name]=device



## A line description
#
class Line(object):

    ## Creates a line description
    #
    # @param [in] name        The line name (used for identification)
    # @param [in] voltage     The line voltage
    # @param [in] description A text description of the line
    #
    def __init__(self, name, voltage, description):
        self.name=name
        self.voltage=voltage
        self.description=description

    ## Returns a string representation for this line
    def __repr__(self):
        return "Line {0} (voltage: {1}, description: '{2}')".format(self.name, self.voltage, self.description)



## A PDU line description
#
class PDULine(object):

    ## Creates a PDU line description
    #
    # @param [in] name        The line name (used for identification)
    # @param [in] computer    The computer this PDU line is attached to
    # @param [in] voltage     The line voltage
    # @param [in] description An optional text description of the line
    #
    def __init__(self, name, computer, voltage, description=""):
        if not isinstance(computer, Computer):
            msg="the given computer parameter is not a Computer object"
            raise SyntaxError, msg
        self.name=name
        self.computer=computer
        self.voltage=voltage
        self.description=description

    ## Returns a string representation for this line
    def __repr__(self):
        return "Line {0} (computer: '{1}', voltage: {2}, description: '{3}')".format(self.name, self.computer.name, self.voltage, self.description)



## A device description
#
class Device(object):

    ## Creates a device description and adds it to the devices
    ## dictionary
    #
    #  Before adding the given device description to the devices
    #  dictionary, it checks that the name of the new device has not
    #  been used by a previously added device.
    #
    # @param [in] name           The device name (used for identification, must be unique)
    # @param [in] url            The url of this device
    # @param [in] max_frequency  The maximum sample frequency of the device
    #
    def __init__(self, name, url, max_frequency):
        self.name=name
        self.url=url
        self.max_frequency=max_frequency
        self.lines={}
        if devices.has_key(name):
            msg="there are at least two devices with the same name ({0}).".format(self.name)
            raise SyntaxError, msg
        # Register the device
        devices[name]=self

    ## Returns a string representation for this device
    def __repr__(self):
        return "Device {0} (url: '{1}', max frequency: {2}, lines: {3})".format(self.name, self.url, self.max_frequency, len(self.lines))

    ## Adds a line description to the device
    #
    #  Before adding the given line description to the device, it
    #  checks that the name of the new line has not been used by a
    #  previously added line.
    #
    # @param [in] name        The line name (used for identification)
    # @param [in] voltage     The line voltage
    # @param [in] description A text description of the line
    #
    def add_line(self, name, voltage, description):
        if self.lines.has_key(name):
            msg="there are at least two lines with the same name, '{0}', in device '{1}'.".format(name, self.name)
            raise SyntaxError, msg
        self.lines[name]=Line(name, voltage, description)



## A device attached to a computer description
#
class AttachedDevice(Device):

    ## Creates device attached to a computer description and adds it
    ## to the devices dictionary
    #
    #  Before adding the given device description to the devices
    #  dictionary, it checks that the name of the new device has not
    #  been used by a previously added device.
    #
    # @param [in] name           The device name (used for identification, must be unique)
    # @param [in] computer       The computer the device is attached to
    # @param [in] url            The url of this device
    # @param [in] max_frequency  The maximum sample frequency of the device
    #
    def __init__(self, name, computer, url, max_frequency):
        if not isinstance(computer, Computer):
            msg="the given computer parameter is not a Computer object"
            raise SyntaxError, msg
        self.computer=computer
        super(AttachedDevice, self).__init__(name, url, max_frequency)
        # Register the device in the computer it is attached to
        self.computer.add(self)


## A DCMeter device description
#
class DCDevice(AttachedDevice):

    ## Creates a DC2Meter device description and adds it to the
    ## devices dictionary
    #
    # @param [in] name           The device name (used for identification, must be unique)
    # @param [in] computer       The computer the device is attached to
    # @param [in] url            The url of this device
    # @param [in] max_frequency  The maximum sample frequency of the device
    #
    def __init__(self, name, computer, url, max_frequency):
        super(DCDevice, self).__init__(name, computer, url, max_frequency)



## A DC2Meter device description
#
class DC2Device(AttachedDevice):

    ## Creates a DC2Meter device description and adds it to the
    ## devices dictionary
    #
    # @param [in] name           The device name (used for identification, must be unique)
    # @param [in] computer       The computer the device is attached to
    # @param [in] url            The url of this device
    # @param [in] max_frequency  The maximum sample frequency of the device
    #
    def __init__(self, name, computer, url, max_frequency):
        super(DC2Device, self).__init__(name, computer, url, max_frequency)



## A National Instruments device description
#
class NIDevice(AttachedDevice):

    ## Creates a National Instruments device description and adds it
    ## to the devices dictionary
    #
    # @param [in] name           The device name (used for identification, must be unique)
    # @param [in] computer       The computer the device is attached to
    # @param [in] url            The url of this device
    # @param [in] max_frequency  The maximum sample frequency of the device
    #
    def __init__(self, name, computer, url, max_frequency):
        super(NIDevice, self).__init__(name, computer, url, max_frequency)

```

```


## A WattsUp device description
#
class WattsUpDevice(AttachedDevice):

    ## Creates a WattsUp device description and adds it to the devices
    ## dictionary
    #
    # @param [in] name           The device name (used for identification, must be unique)
    # @param [in] computer       The computer the device is attached to
    # @param [in] url            The url of this device
    # @param [in] max_frequency  The maximum sample frequency of the device
    #
    def __init__(self, name, computer, url, max_frequency):
        super(WattsUpDevice, self).__init__(name, computer, url, max_frequency)

    ## Fake adding of line description to the device
    #
    #  The WattsUp Device does not have lines. This method avoids the
    #  base class method silently been called.
    #
    # @param [in] name        The line name (used for identification)
    # @param [in] voltage     The line voltage
    # @param [in] description A text description of the line
    #
    def add_line(self, name, voltage, description):
        msg="a WattsUp Device can not have lines"
        raise SyntaxError, msg



## A PDU device description
#
class PDUDevice(Device):

    ## Creates a PDU device description and adds it to the devices
    ## dictionary
    #
    # @param [in] name           The device name (used for identification, must be unique)
    # @param [in] url            The url of this device
    # @param [in] max_frequency  The maximum sample frequency of the device
    #
    def __init__(self, name, url, max_frequency):
        super(PDUDevice, self).__init__(name, url, max_frequency)


    ## Adds a pdu line description to the device
    #
    #  Before adding the given line description to the device, it
    #  checks that the name of the new line has not been used by a
    #  previously added line.
    #
    # @param [in] name        The line name (used for identification)
    # @param [in] computer    The computer the line is attached to
    # @param [in] voltage     The line voltage
    # @param [in] description An optional text description of the line
    #
    def add_line(self, name, computer, voltage, description=""):
        if self.lines.has_key(name):
            msg="there are at least two lines with the same name, '{0}', in device '{1}'.".format(name, self.name)
            raise SyntaxError, msg
        self.lines[name]=PDULine(name, computer, voltage, description)
        computer.add(self)
```