class CEFevt

| CEFevt class doc
|
| This Class allows creating and manipulating CEF events objects.
|
| A CEF instance has following attributes:
| CEF Header : obj.header : the mandatory parameters located at the beginning of a CEF event
| CEF Tail : obj.tail : the optional key=value pairs defining the CEF extension and located at the
end of the CEF event
|
| The methods are split in 4 main groups:
| Generic : apply to the whole cef event
| Header : modification of the header (method name always start with 'h')
| Tail : modification of the tail (method name always start with 't')
| Internal : not meant to be called by an external program. Should only be used by the class
functions
|
| Available functions (Internal function not shown):
| __init__
| display
| help
| stringest
| hupdate
| hcleandisplay
| tupdate
| tremove
| tempty
| tcleandisplay
|
| Documentation:
| From the Python interpreter:
| >>> import cef
| >>> print(cef.CEFevt.function_name.__doc__)
|
| Methods defined here:
|

| __init__(self, inputdict={}, dictail={})

|
| Desc: initializing the CEF header. If no data provided for the header,
| default value will be used in order to have a valid CEF event.
| If no data provided for the tail, tail is kept empty.
|

| Syntax:
| obj=CEFevt([dicthead],[dictail])
|
| dicthead: dictionary containing {fieldname:value} key pairs for the tail.
| ie: {"deviceVendor": "VendorX", "deviceProduct": "ProductY"}
|
| Accepted keys:
|
| deviceVendor: string
| deviceProduct: string
| deviceVersion: string
| deviceEventClassId: string
| name: string
| severity: string
|
| other keys won't generate any errors but will be discarded. If keys are missing,
| default values will be used
|
| dictail: dictionary containing {fieldname:value} key pairs for the tail
| ie: {'msg': 'test message', 'src': '1.2.3.4'}

| display(self)

|
| Desc: just printing the cef event
|
| Syntax:
| obj.display()
|
| Return:
|
| cefstr: a string containing the cef event
|

| hcleandisplay(self)

|
| Desc: printing the cef event header with nice and readable output (1 fieldname/fieldvalue pair
| per line)
|
| Syntax:
| obj.hcleandisplay()
|

| help(self, method='')

|
| Desc: printing the doc of a method. If no method is provided, the class doc and the list of

methods is printed.

Syntax:
obj.help([method])
method: string

Return:
doc: string
The output of the method

hupdate(self, dicinput)

Desc: replace the value of one or several fields in the cef header. In order to avoid typo when calling this function
several values are valid for some field names and the field name is not case sensitive .
It is also possible to use a numerical value to design the fieldname which needs to be changed

Syntax:
obj.hupdate(dicinput)
dicinput: dictionary

Allowed fieldnames (not case sensitive):
deviceVendor 1
deviceProduct 2
deviceVersion 3
deviceEventClassId SigId SignatureId 4
name 5
severity 6

Example:
obj.hupdate({"deviceVendor": "foo", "2": "bar", "siGiD": "400"})

Return:
head: dictionary containing the header fieldname/value pairs

stringest(self, strcef)

Desc: Ingest a CEF event in string format. It can then be used with all the other functions

Syntax:
obj.stringest(cefevent)
cefevent: string

|

| tcleandisplay(self, mapdict={})

|
| Desc: printing the cef event tail with nice and readable output (1 fieldname/fieldvalue pair per
| line)
| optionally, a dictionary mapping cef fieldname with more meaningful name can be
| provided
|
| Syntax:
| obj.tcleandisplay()
|

| tempty(self)

|
| Desc: simply empty the tail
|
| Syntax:
| obj.tempty()
|

| tremove(self, fieldslist)

|
| Desc: remove one or several fieldname(s) and their value(s) from the tail
|
| Syntax:
| obj.tremove(fieldslist)
| fieldslist: list containing fieldnames to remove from the tail
| ie:('msg', 'src')
|
| Return:
| tail: dictionary containing new tail fieldnames/values key pairs
|

| tupdate(self, dict)

|
| Desc: add new fieldname(s)/value(s) or change value(s) of existing fieldname(s)
|
| Syntax:
| obj.tupdate(dict)
| dict: dictionary
| ie: {'msg': 'test message', 'src': '1.2.3.4'}
|
| Return:
| tail: dictionary containing the tail values

##################
Examples
##################

cef=CEFevt()
print cef.header

{'severity': '0', 'deviceVendor': 'test', 'deviceVersion': '1.0', 'deviceEventClassId': '100',
'deviceProduct': 'test', 'name': 'test event'}

cef.hupdate({1:"MyVendor"})
print cef.header

{'severity': '0', 'deviceVendor': 'MyVendor', 'deviceVersion': '1.0', 'deviceEventClassId': '100',
'deviceProduct': 'test', 'name': 'test event'}

cef.tupdate({"foo":"bar", "bar":"foo"})
print cef.tail

{'foo': 'bar', 'bar': 'foo'}

cef.display()

CEF:0|MyVendor|test|1.0|100|test event|0|foo=bar bar=foo

cef.stringest("CEF:0|prod|test|1.0|100|test event|0| cs1=range cs2=High cs2Label=Level of severity ")
cef.display()

CEF:0|prod|test|1.0|100|test event|0|cs1=Suspicious cs2Label=Level of severity cs2=High

cef.tcleandisplay()

CEF event tail:
================
cs1 Suspicious
cs2 High
cs2Label Level of severity

cef.hcleandisplay()

CEF event header:
=================

deviceEventClassId 100
deviceProduct test
deviceVendor prod
deviceVersion 1.0
name test event
severity 0

cef.tremove('cs1', 'cs2')
cef.display()

CEF:0|prod|test|1.0|100|test event|0|cs2Label=Level of severity

cef.tempty()
cef.display()

CEF:0|prod|test|1.0|100|test event|0|

print (cef.hupdate.__doc__)

hupdate

Desc: replace the value of one or several fields in the cef header. In order to avoid typo when calling this function
several values are valid for some field names and the field name is not case sensitive .
It is also possible to use a numerical value to design the fieldname which needs to be changed

Syntax:
obj.hupdate(dicinput)
dicinput: dictionary

Allowed fieldnames (not case sensitive):
deviceVendor 1
deviceProduct 2
deviceVersion 3
deviceEventClassId SigId SignatureId 4
name 5
severity 6

Example:
obj.hupdate({"deviceVendor": "foo", "2": "bar", "siGiD": "400"})

Return:
head: dictionary containing the header fieldname/value pairs
#help(cef.hupdate)