enigma2_http_api Documentation

Release 0.5.0+1.g81008da.dirty

doubleO8

Nov 10, 2018

Contents:

CHAPTER 1

Controller

enigma2_http_api.controller.**ENIGMA2_URL_FMT = '{scheme}://{remote_addr}/{path}'**
enigma2 web interface URL format string

class enigma2_http_api.controller.**Enigma2APIController**(**args*, ***kwargs*)
Enigma2 Web API Consuming Controller Class

get_about()
Retrieve information about enigma2 device.

Returns Enigma2 device information

Return type dict

get_epgbouquet(*bouquet_ref*, *filter_func=None*)
Get EPG datasets for *bouquet_ref*. (**currently** running subservices' EPG datasets)

Parameters

- **bouquet_ref** – bouquet reference
- **filter_func** – filter function

Returns EPG datasets of current subservice

Return type list

get_epgservice(*service_ref*, *filter_func=None*)
Get EPG datasets for *service_ref*.

Parameters

- **service_ref** – service reference
- **filter_func** – filter function

Returns EPG datasets of given service

Return type list

get_getallservices()
Get all services.

Returns

get_message(*messagetext*, *timeout=10*, *messagetype=1*)
Display a message on enigma2's attached screen.

Parameters

- **messagetext** – message
- **timeout** – timeout
- **messagetype** – message type

Returns

get_messageanswer(*messagetext*, *timeout=10*)
Display a message and wait for user selection. (SEEMS TO BE NON-WORKING)

Parameters

- **messagetext** – message
- **timeout** – timeout

Returns

get_moviedelete(*service_ref*)
Delete a movie item.

Parameters **service_ref** – service reference

Returns

get_movielist()
Get list of movie items available on *self.remote_addr*.

Returns

get_powerstate(*new_state*)
Change Power State

Parameters **new_state** – Desired Power State

Returns

get_search(*what*, *filter_func=None*)
Search EPG for *what*. Will filter results if *filter_func* is given.

Parameters

- **what** – Search string
- **filter_func** – result filtering function

Returns

get_services()
Get services (bouquets).

Returns list containing service name and reference

Return type list

get_subservices()
Get subservices for current service

Returns subservices of current service

Return type list

get_timeradd(*service_ref*, *params*)
Add a new timer

/web/timeradd?sRef=&repeated=&begin=&end=&name=&description=&dirname=&tags=&eit=&disabled=&justplay=&af

Parameters

- **service_ref** – service reference
- **params** – timer parameters

Returns

get_timeraddbyeventid(*service_ref*, *event_id*)
Add a new timer by ID.

Parameters

- **service_ref** – service reference
- **event_id** – ID of the event to be recorded

Returns

get_timerdelete(*service_ref*, *begin*, *end*)
Delete an existing timer.

Parameters

- **service_ref** – service reference
- **begin** – start time
- **end** – end time

Returns

get_timerlist()
Get list of timers.

Returns

get_zap(*service_ref*)
Try to zap to given service.

Parameters **service_ref** – service reference

Returns

update_movielist_map()
Update internal movie list map *self.movielist_map* by retrieving the current list of movie items on Enigma2 box.

CHAPTER 2

Utility functions

See also:

- http://dreambox.de/board/index.php?thread/13534-satellite-position-id-need-to-know-what-western-ones-mean-for-my-plugin-e&s=3f1817729cb55399e2a345953329ce54e8a1a150&postID=89843
- https://www.linuxtv.org/pipermail/linux-dvb/2005-June/002618.html
- http://radiovibrations.com/dreambox/namespace.htm
- http://www.dreambox-tools.info/print.php?threadid=1908&page=1&sid=1e12a523fe12d2073918670bff46ba23
- https://wiki.neutrino-hd.de/wiki/Enigma:Services:Formatbeschreibung
- http://radiovibrations.com/dreambox/services.htm

enigma2_http_api.utils.**LISTING_ITEM_KEY_PAIRS = [('eventname', 'descriptionExtended'), ('t**
list of tuples which may contain title and description of an event

enigma2_http_api.utils.**NS = {'DVB-C': 4294901760, 'DVB-S': 12582912, 'DVB-T': 4008574976}**
Label:Namespace map

enigma2_http_api.utils.**NS_DVB_C = 4294901760**
Namespace - DVB-C services

enigma2_http_api.utils.**NS_DVB_S = 12582912**
Namespace - DVB-S services

enigma2_http_api.utils.**NS_DVB_T = 4008574976**
Namespace - DVB-T services

enigma2_http_api.utils.**NS_LOOKUP = {0: 'File', 130: 'DVB-S', 192: 'DVB-S', 61166: 'DVB-**
Namespace:Label lookup map

enigma2_http_api.utils.**create_picon**(**args*, ***kwargs*)
Generate a (Enigma2 style) program icon string representation.

Parameters

- **args[0]** (dict) – Service Reference Parameter as dict

- **service_type** (*int*) – Service Type
- **sid** (*int*) – SID
- **tsid** (*int*) – TSID
- **oid** (*int*) – OID
- **ns** (*int*) – Enigma2 Namespace

enigma2_http_api.utils.**create_servicereference**(**args*, ***kwargs*)

Generate a (Enigma2 style) service reference string representation.

Parameters

- **args[0]** (dict) – Service Reference Parameter as dict
- **service_type** (*int*) – Service Type
- **sid** (*int*) – SID
- **tsid** (*int*) – TSID
- **oid** (*int*) – OID
- **ns** (*int*) – Enigma2 Namespace

enigma2_http_api.utils.**enigma_trunkname**(*path*)

Determine the trunk of enigma2 specific files.

Parameters **path** – filename

Returns trunk

```
>>> enigma_trunkname(None)
Traceback (most recent call last):
    ...
ValueError: None
>>> enigma_trunkname('')
Traceback (most recent call last):
    ...
ValueError: ''
>>> enigma_trunkname(2)
Traceback (most recent call last):
    ...
ValueError: 2
>>> enigma_trunkname('somefile')
Traceback (most recent call last):
    ...
ValueError: 'somefile' has no extension
>>> enigma_trunkname('somefile.bla')
Traceback (most recent call last):
    ...
ValueError: 'somefile.bla' has bad extension 'bla'
>>> enigma_trunkname('somefile.bla.bla')
Traceback (most recent call last):
    ...
ValueError: 'somefile.bla.bla' has bad extension 'bla'
>>> enigma_trunkname('somefile.ts')
'somefile'
>>> enigma_trunkname('/tmp/somefile.ts')
'somefile'
>>> enigma_trunkname('somefile.ts.sc')
'somefile'
```

enigma2_http_api.utils.**filter_simple_events**(*data*)

```
[
    u'begin',
    u'sname',
    u'end',
    u'title',
    u'id',
    u'now_timestamp',
    u'picon',
    u'longdesc',
    u'duration',
    u'duration_sec',
    u'sref',
    u'date',
    u'shortdesc',
    u'progress',
    u'tleft',
    u'begin_timestamp'],
```

Parameters **data** –

Returns

enigma2_http_api.utils.**guess_namespace_label**(*value*, *fallback='UNKNOWN'*)

Try to guess a textual representation for given namespace value.

Parameters

- **value** – namespace
- **fallback** – value to be returned if no matching label is found

Returns label

Return type basestring

```
>>> guess_namespace_label(1234)
'UNKNOWN'
>>> guess_namespace_label(99, 'DVB-S')
'DVB-S'
>>> guess_namespace_label(0x0)
'File'
>>> guess_namespace_label(0x00c00000)
'DVB-S'
>>> guess_namespace_label(0x0c0)
'DVB-S'
>>> guess_namespace_label(0x00c0)
'DVB-S'
>>> guess_namespace_label(0x00c01234)
'DVB-S'
>>> guess_namespace_label(0x00820000)
'DVB-S'
>>> guess_namespace_label(0x0082)
'DVB-S'
>>> guess_namespace_label(0x00820082)
'DVB-S'
>>> guess_namespace_label(0x00000001)
```

(continues on next page)

(continued from previous page)

```
'UNKNOWN'
>>> guess_namespace_label(0xffef1234)
'UNKNOWN'
>>> guess_namespace_label(0xeeeeffff)
'DVB-T'
```

enigma2_http_api.utils.**normalise_servicereference**(*serviceref*)

Create a normalised representation of *serviceref* to be used e.g. as sorting hint

Parameters **serviceref** – service reference

Returns

```
>>> sref = '1:0:1:300:7:85:00c00000:0:0:0:'
>>> normalise_servicereference(sref)
'0001:0300:0007:0085:00C00000'
>>> sref2 = '1:64:A:0:0:0:0:0:0:0::SKY Sport'
>>> normalise_servicereference(sref2)
'000A:0000:0000:0000:00000000'
```

enigma2_http_api.utils.**parse_servicereference**(*serviceref*)

Parse a Enigma2 style service reference string representation.

Parameters **serviceref** (*string*) – Enigma2 style service reference

```
>>> sref = '1:0:1:300:7:85:00c00000:0:0:0:'
>>> result = parse_servicereference(sref)
>>> result
{'service_type': 1, 'oid': 133, 'tsid': 7, 'ns': 12582912, 'sid': 768}
>>> sref_g = create_servicereference(**result)
>>> sref_g
'1:0:1:300:7:85:00c00000:0:0:0:'
>>> sref_g2 = create_servicereference(result)
>>> sref_g2
'1:0:1:300:7:85:00c00000:0:0:0:'
>>> sref == sref_g
True
>>> sref2 = '1:64:A:0:0:0:0:0:0:0::SKY Sport'
>>> result2 = parse_servicereference(sref2)
>>> result2
{'service_type': 10, 'oid': 0, 'tsid': 0, 'ns': 0, 'sid': 0}
>>> sref3 = '1:0:0:0:0:0:0:0:0:0:/media/hdd/movie/20170921 2055 - DASDING -␣
↪DASDING Sprechstunde - .ts'
>>> result3 = parse_servicereference(sref3)
>>> result3
{'service_type': 0, 'oid': 0, 'tsid': 0, 'ns': 0, 'sid': 0}
```

enigma2_http_api.utils.**pseudo_unique_id**(*item*)

Generate a pseudo unique ID for an event item, movie item or timer item. The ID is based on the item's title and description attribute. Neither title nor description may be empty.

Parameters **item** – event, movie or timer

Returns generated pseudo ID

Return type str

```
>>> pseudo_unique_id({'event': 1})
Traceback (most recent call last):
    ...
AssertionError: name or desc may not be None
>>> pseudo_unique_id({'eventname': 1, 'descriptionExtended': 'bla'})
Traceback (most recent call last):
    ...
AttributeError: 'int' object has no attribute 'strip'
>>> pseudo_unique_id({'eventname': "x", 'descriptionExtended': 'bla'})
'7a6615ef8ca6b06ac6a837741293759d3083a49c'
>>> pseudo_unique_id({'eventname': "x", 'description': 'bla'})
'7a6615ef8ca6b06ac6a837741293759d3083a49c'
>>> pseudo_unique_id({'eventname': " ", 'descriptionExtended': '  '})
Traceback (most recent call last):
    ...
AssertionError: name or desc may not be empty
>>> pseudo_unique_id({'title': "x", 'longdesc': 'bla'})
'7a6615ef8ca6b06ac6a837741293759d3083a49c'
>>> pseudo_unique_id({'name': "x", 'descriptionextended': 'bla'})
'7a6615ef8ca6b06ac6a837741293759d3083a49c'
>>> pseudo_unique_id({'name': "x", 'descriptionextended': ' 17 Min.'})
Traceback (most recent call last):
    ...
AssertionError: desc_mangled may not be empty
>>> pseudo_unique_id({'name': "x", 'descriptionextended': 'Bla Bla 17 Min.'})
'4db3822ad252366e57d9515b1e37d3449a45a0cb'
>>> pseudo_unique_id({'name': "x", 'descriptionextended': 'Bla Bla 18 Min.'})
'4db3822ad252366e57d9515b1e37d3449a45a0cb'
>>> pseudo_unique_id({'eventname': "x", 'descriptionExtended': 'bla', 'description
↪': ''})
'7a6615ef8ca6b06ac6a837741293759d3083a49c'
```

enigma2_http_api.utils.**set_output_encoding**(*encoding='utf-8'*)

Stolen from https://stackoverflow.com/ questions/19696652/piping-output-causes-python-program-to-fail

When piping to the terminal, python knows the encoding needed, and sets it automatically. But when piping to another program (for example, | less), python can not check the output encoding. In that case, it is None. What I am doing here is to catch this situation for both stdout and stderr and force the encoding

CHAPTER 3

Model

3.1 Unified event class

When returning resultsets containing EPG or timer items the `dict` derived class *`enigma2_http_api.model.EEvent`* is used.

EEvent attribute	epg	timer	movie
duration[2]	duration_sec	– n/a –	length
item_id	id	eit	– n/a –
longinfo	longdesc	descriptionextended	descriptionExtended
service_name	sname	servicename	servicename
service_reference	sref	serviceref	serviceref
shortinfo	shortdesc	description	description
start_time[1]	begin_timestamp	begin	recordingtime
stop_time[1]	– n/a –	end	– n/a –
title	title	name	eventname

3.1.1 Model classes and constants.

enigma2_http_api.model.**DEFAULT_LOCALTIMEZONE = 'Europe/Berlin'**
default/fallback value for local timezone as the enigma2 API returns localised timestamps (not UTC!) one need to set the correct timezone or the results will not match the values shown on the enigma2 device.

enigma2_http_api.model.**DT_FORMAT__REAL__KEYS = '%d.%m.%Y %H:%M'**
datetime format as used for *realbegin*/*realend* key/value pairs of timer items returned by enigma2 API

class enigma2_http_api.model.**EEvent**(**args*, ***kwargs*)
This is a thin wrapper class to allow unified access to event items' data.

[2] `datetime.timedelta` instances.
[1] `datetime.datetime` instances.

As the data keys for EPG and timer events (returned by the enigma2 API) are different for the same data values the data is exposed as attributes common to all item types.

Attribute names are inspired by names used in the ETSI EPG specification documents.

1. ETSI EN 300 707 V1.2.1 (2002-12)
2. ETSI ETR 288 TECHNICAL October 1996

See also:

- http://www.etsi.org/deliver/etsi_en/300700_300799/300707/01.02.01_40/en_300707v010201o.pdf
- http://www.etsi.org/deliver/etsi_etr/200_299/288/01_60/etr_288e01p.pdf

enigma2_http_api.model.**ITEM_TYPE_EPG = 'epg'**
type identifier for EPG items

enigma2_http_api.model.**ITEM_TYPE_MOVIE = 'movie'**
type identifier for MOVIE items

enigma2_http_api.model.**ITEM_TYPE_TIMER = 'timer'**
type identifier for timer items

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index

N

P

S

U