
CPE v1.2.0 Documentation

Release 1.2.0

Roberto Abdelkader Martínez Pérez, Alejandro Galindo García

February 26, 2016

1	Introduction	3
1.1	Matching	3
1.1.1	Conceptual model	3
1.1.2	Matching Algorithm: Known Instance Based Matching	4
2	List of implemented CPE versions	5
3	Compatibility among CPE versions	7
4	Model	9
4.1	CPE hierarchy	9
4.1.1	Class list	9
	CPE class	9
	CPE1_1 class	12
	CPE2_2 class	13
	CPE2_3 class	14
	CPE2_3_URI class	15
	CPE2_3_WFN class	16
	CPE2_3_FS class	17
4.1.2	Class diagram	17
4.2	CPESet hierarchy	18
4.2.1	Class list	19
	CPESet class	19
	CPESet1_1 class	20
	CPESet2_2 class	21
	CPESet2_3 class	21
4.2.2	Class diagram	23
4.3	CPELanguage hierarchy	24
4.3.1	Class list	24
	CPELanguage class	24
	CPELanguage2_2 class	25
	CPELanguage2_3 class	26
4.3.2	Class diagram	26
4.4	CPEComponent hierarchy	27
4.4.1	Class list	27
	CPEComponent class	27
	CPEComponentLogical class	28
	CPEComponentAnyValue class	29
	CPEComponentEmpty class	29
	CPEComponentNotApplicable class	30
	CPEComponentUndefined class	30
	CPEComponentSimple class	31

CPEComponent1_1 class	32
CPEComponent2_2 class	34
CPEComponent2_3 class	34
CPEComponent2_3_URI class	34
CPEComponent2_3_edpacked class	34
CPEComponent2_3_WFN class	35
CPEComponent2_3_FS class	35
4.4.2 Class diagram	35
4.5 Categories of main classes	37
5 Installation	39
6 Usage examples	41
6.1 Naming	41
6.2 Name matching	42
6.3 Language matching	43
7 Important issues	45
8 Bugtracker	47
9 TODO	49
10 References	51
11 Indices and tables	53

Welcome to cpe package documentation. In this pages you can find a tutorial about cpe package, that implements the validation of both CPE Names and platform (set of CPE Names) and the comparisons between them, corresponding to versions 1.1, 2.2 and 2.3 of CPE (Common Platform Enumeration) specification. Also, this package gives support to name and language matching algorithms.

This documentation contains a brief introduction about Common Platform Enumeration (CPE) specification, the CPE version list implemented and the compatibility among them, the class model designed, the steps to install and use the cpe package (with examples), and several important issues to consider associated with the functionality of the package.

For more information about cpe package implementation, please visite: <https://github.com/nilp0inter/cpe>.

Contents:

Introduction

Common Platform Enumeration (CPE) is a standardized method of describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise's computing assets. CPE does not identify unique instantiations of products on systems, such as the installation of XYZ Visualizer Enterprise Suite 4.2.3 with serial number Q472B987P113. Rather, CPE identifies abstract classes of products, such as XYZ Visualizer Enterprise Suite 4.2.3, XYZ Visualizer Enterprise Suite (all versions), or XYZ Visualizer (all variations).

IT management tools can collect information about installed products, identifying these products using their CPE Names, and then use this standardized information to help make fully or partially automated decisions regarding the assets. For example, identifying the presence of XYZ Visualizer Enterprise Suite could trigger a vulnerability management tool to check the system for known vulnerabilities in the software, and also trigger a configuration management tool to verify that the software is configured securely in accordance with the organization's policies. This example illustrates how CPE Names can be used as a standardized source of information for enforcing and verifying IT management policies across tools [1].

CPE provides [2]:

- A standard machine-readable format for encoding names of IT products and platforms (naming).
- A set of procedures for comparing names (name matching).
- A language for constructing “applicability statements” that combine CPE Names with simple logical operators (language matching).
- A standard notion of a CPE Dictionary.

For more information, please visit the official website of CPE, maintained by MITRE: <http://cpe.mitre.org/>

1.1 Matching

Matching [5] is the process of determining if a given CPE Name or CPE Language statement specifies a platform that is defined by a set of known CPE Names. It helps define the relationship between different CPE Names (or language statements).

1.1.1 Conceptual model

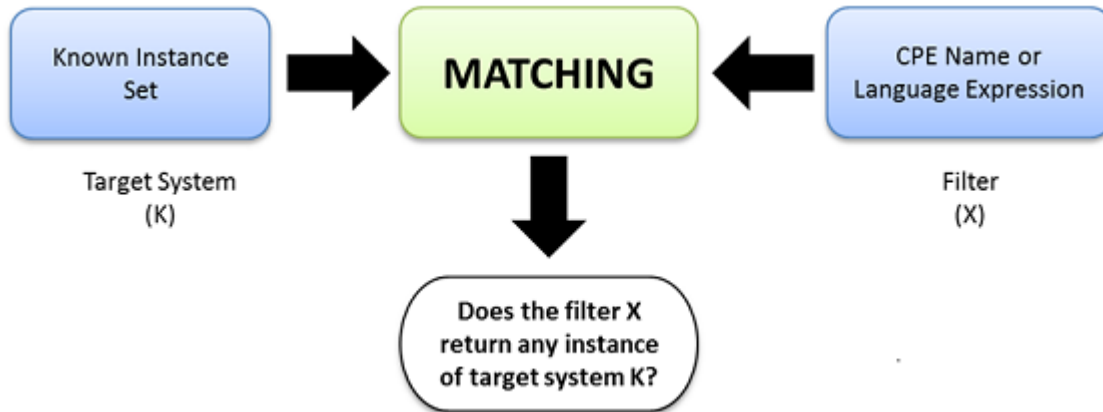
The conceptual model for matching consists of two steps:

1. Make a list of all the CPE Names and logical connections.
2. For each name, check whether the target system has the hardware, software, or operating system indicated by the name. If the check succeeds for all names, and satisfies the logical constraints, then the target is an instance of the CPE Name or Language representation.

1.1.2 Matching Algorithm: Known Instance Based Matching

Two elements participate in the matching process: the known instance set (CPE Names that define the target system) and the candidate CPE Name or Language expression. The algorithm applies the “filter” indicated by X and replies to the question: “Does the filter X return any instance of target system K?”

The figure bellow illustrates the basic concept of known instance based matching:



Matching consists of two algorithms.

- CPE_Name_Match:
 - This algorithm accepts a set of CPE Names K and a candidate CPE Name X.
 - It returns true if X matches any member of K, and false otherwise.
- CPE_Language_Match:
 - This algorithm accepts a set of CPE Names K and a candidate expression E.
 - It returns true if the expression can be satisfied by name matching against the members of K, and false otherwise.

List of implemented CPE versions

This package implements the validation of both CPE Names and platforms (set of CPE Names), and the comparisons between them, corresponding to some versions of CPE specification [3].

The functionality implemented in this package, associated with versions 1.1, 2.2 and 2.3 of CPE specification, is below:

- **Version 1.1** [4]:
 - CPE naming
 - CPE Name matching
- **Version 2.2** [5]:
 - CPE naming
 - CPE Name matching
 - CPE Language matching
- **Version 2.3**:
 - CPE naming [6]
 - CPE Name matching [7]
 - CPE Applicability Language matching [8]

The CPE naming of version 2.3 supports the definition of three different styles of CPE Name:

- **WFN**: Well-Formed Name
- **URI**: Uniform Resource Identifier
- **FS**: Formatted String

Compatibility among CPE versions

VER- SIONS	1.1	2.2	2.3 WFN	2.3 URI	2.3 FS
1.1	Yes	Depends of count of parts	Depends of count of parts	Depends of count of parts	Depends of count of parts
2.2	Depends of characters	Yes	Yes	Yes	Yes
2.3 WFN	Depends of characters	Depends of characters	Yes	Yes	Yes
2.3 URI	Depends of characters	Depends of characters	Yes	Yes	Yes
2.3 FS	Depends of characters	Depends of characters	Yes	Yes	Yes

Model

This section shows the diagrams of model parts of cpe package. These diagrams have been generated with the PyNSource tool version 1.61 (<https://code.google.com/p/pynsource/>). Each model class is stored in a different file. The model parts are as follows.

4.1 CPE hierarchy

This section contains the classes associated with versions of CPE specification implemented in this package.

4.1.1 Class list

CPE class

class `cpe.cpe.CPE` (*cpe_str*, *args, **kwargs)

Represents a generic CPE Name compatible with all versions of CPE specification.

Parts of CPE are stored in a dictionary.

CPE structure (dictionary):

- **part** {hw, os, sw, undefined}

- **element list (list)** - component list (dictionary)

__eq__ (*other*)

Returns True if other (first element of operation) and self (second element of operation) are equal CPE Names, false otherwise.

Parameters *other* (*CPE*) – CPE Name to compare

Returns True if other == self, False otherwise

Return type boolean

__getitem__ (*i*)

Returns the i'th component name of CPE Name.

Parameters *i* (*int*) – component index to find

Returns component string found

Return type *CPEComponent*

Exception IndexError - index not found in CPE Name

TEST: good index

```
>>> str = 'cpe:///sun_microsystem:sun@os:5.9:#update'
>>> c = CPE(str)
>>> c[0]
CPEComponent1_1(sun_microsystem)
```

__init__(*cpe_str*, **args*, ***kwargs*)
Store the CPE Name.

Parameters *cpe_str* (*string*) – CPE Name

Returns None

__len__()
Returns the number of components of CPE Name.

Returns count of components of CPE Name

Return type *int*

TEST: a CPE Name with two parts (hw and os) and some elements empty and with values

```
>>> str = "cpe:/cisco::3825/cisco:ios:12.3"
>>> c = CPE(str)
>>> len(c)
6
```

static __new__(*cpe_str*, *version=None*, **args*, ***kwargs*)
Generator of CPE Names.

Parameters

- *cpe_str* (*string*) – CPE Name string
- *version* (*string*) – version of CPE specification of CPE Name

Returns CPE object with version of CPE detected correctly

Return type *CPE*

Exception `NotImplementedError` - incorrect CPE Name or version of CPE not implemented

This class implements the factory pattern, that is, this class centralizes the creation of objects of a particular CPE version, hiding the user the requested object instance.

__repr__()
Returns a unambiguous representation of CPE Name.

Returns Representation of CPE Name as string

Return type *string*

__str__()
Returns a human-readable representation of CPE Name.

Returns Representation of CPE Name as string

Return type *string*

__weakref__
list of weak references to the object (if defined)

as_dict()
Returns the CPE Name dict as string.

Returns CPE Name dict as string

Return type *string*

as_fs()
Returns the CPE Name as formatted string of version 2.3.

Returns CPE Name as formatted string

Return type `string`

Exception `TypeError` - incompatible version

as_uri_2_3 ()

Returns the CPE Name as URI string of version 2.3.

Returns CPE Name as URI string of version 2.3

Return type `string`

Exception `TypeError` - incompatible version

as_wfn ()

Returns the CPE Name as Well-Formed Name string of version 2.3.

Returns CPE Name as WFN string

Return type `string`

Exception `TypeError` - incompatible version

get_edition ()

Returns the edition of product of CPE Name as a list. According to the CPE version, this list can contains one or more items.

Returns Value of edition attribute as string list.

Return type `list`

get_language ()

Returns the internationalization information of CPE Name as a list. According to the CPE version, this list can contains one or more items.

Returns Value of language attribute as string list.

Return type `list`

get_other ()

Returns the other information part of CPE Name.

Returns Value of other attribute as string list.

Return type `list`

get_part ()

Returns the part component of CPE Name as a list. According to the CPE version, this list can contains one or more items.

Returns Value of part attribute as string list.

Return type `list`

get_product ()

Returns the product name of CPE Name as a list. According to the CPE version, this list can contains one or more items.

Returns Value of product attribute as string list.

Return type `list`

get_software_edition ()

Returns the software edition of CPE Name.

Returns Value of sw_edition attribute as string list.

Return type `list`

get_target_hardware ()

Returns the architecture of CPE Name.

Returns Value of target_hw attribute as string list.

Return type `list`

get_target_software()

Returns the software computing environment of CPE Name within which the product operates.

Returns Value of target_sw attribute as string list.

Return type `list`

get_update()

Returns the update or service pack information of CPE Name as a list. According to the CPE version, this list can contains one or more items.

Returns Value of update attribute as string list.

Return type `list`

get_vendor()

Returns the vendor name of CPE Name as a list. According to the CPE version, this list can contains one or more items.

Returns Value of vendor attribute as string list.

Return type `list`

get_version()

Returns the version of product of CPE Name as a list. According to the CPE version, this list can contains one or more items.

Returns Value of version attribute as string list.

Return type `list`

is_application()

Returns True if CPE Name corresponds to application elem.

Returns True if CPE Name corresponds to application elem, False otherwise.

Return type `boolean`

is_hardware()

Returns True if CPE Name corresponds to hardware elem.

Returns True if CPE Name corresponds to hardware elem, False otherwise.

Return type `boolean`

is_operating_system()

Returns True if CPE Name corresponds to operating system elem.

Returns True if CPE Name corresponds to operating system elem, False otherwise.

Return type `boolean`

CPE1_1 class

class `cpe.cpe1_1.CPE1_1(cpe_str, *args, **kwargs)`

Implementation of version 1.1 of CPE specification.

Basic structure of CPE Name:

- Hardware part: the physical platform supporting the IT system.
- Operating system part: the operating system controls and manages the IT hardware.
- Application part: software systems, services, servers, and packages installed on the system.

CPE Name syntax:

`cpe:/ {hardware-part} [ / {OS-part} [ / {application-part} ] ]`

__getitem__ (*i*)

Returns the *i*'th component name of CPE Name.

Parameters *i* (*int*) – component index to find

Returns component string found

Return type *CPEComponent*

Exception IndexError - index not found in CPE Name

TEST: good index

```
>>> str = 'cpe:///sun_microsystem:sun@os:5.9:#update'
>>> c = CPE1_1(str)
>>> c[0]
CPEComponent1_1(sun_microsystem)
```

__len__ ()

Returns the number of components of CPE Name.

Returns count of components of CPE Name

Return type *int*

TEST: a CPE Name with two parts (hw and os) and some elements empty and with values

```
>>> str = "cpe:/cisco::3825/cisco:ios:12.3:enterprise"
>>> c = CPE1_1(str)
>>> len(c)
7
```

static __new__ (*cpe_str*, **args*, ***kwargs*)

Create a new CPE Name of version 1.1.

Parameters *cpe_str* (*string*) – CPE Name string

Returns CPE object of version 1.1 of CPE specification.

Return type *CPE1_1*

as_wfn ()

Returns the CPE Name as Well-Formed Name string of version 2.3.

Returns CPE Name as WFN string

Return type *string*

Exception TypeError - incompatible version

get_attribute_values (*att_name*)

Returns the values of attribute “att_name” of CPE Name. By default a only element in each part.

Parameters *att_name* (*string*) – Attribute name to get

Returns List of attribute values

Return type *list*

Exception ValueError - invalid attribute name

CPE2_2 class

class *cpe.cpe2_2.CPE2_2* (*cpe_str*, **args*, ***kwargs*)

Implementation of version 2.2 of CPE specification.

A CPE Name is a percent-encoded URI with each name starting with the prefix (the URI scheme name) ‘cpe:’.

Each platform can be broken down into many distinct parts. A CPE Name specifies a simple part and is used to identify any platform that matches the description of that part.

The distinct parts are:

- Hardware part: the physical platform supporting the IT system.
- Operating system part: the operating system controls and manages the IT hardware.
- Application part: software systems, services, servers, and packages installed on the system.

CPE Name syntax:

`cpe/{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}`

`__len__()`

Returns the number of components of CPE Name.

Returns count of components of CPE Name

Return type `int`

static `__new__(cpe_str, *args, **kwargs)`

Create a new CPE Name of version 2.2.

Parameters `cpe_str` (`string`) – CPE Name string

Returns CPE object of version 2.2 of CPE specification.

Return type `CPE2_2`

as_wfn()

Returns the CPE Name as WFN string of version 2.3. Only shows the first seven components.

Returns CPE Name as WFN string

Return type `string`

Exception `TypeError` - incompatible version

get_attribute_values (`att_name`)

Returns the values of attribute “att_name” of CPE Name. By default a only element in each part.

Parameters `att_name` (`string`) – Attribute name to get

Returns List of attribute values

Return type `list`

Exception `ValueError` - invalid attribute name

CPE2_3 class

class `cpe.cpe2_3.CPE2_3(cpe_str, *args, **kwargs)`

Represents a generic CPE name compatible with all CPE name style of version 2.3 of CPE specification.

static `__new__(cpe_str, *args, **kwargs)`

Generator of CPE Names according to version 2.3.

Parameters `cpe_str` (`string`) – CPE Name string

Returns CPE object of version 2.3 with style detected correctly

Return type `CPE2_3`

Exception `NotImplementedError` - incorrect CPE Name or version of CPE not implemented

This class implements the factory pattern, that is, this class centralizes the creation of objects of a particular CPE style of version 2.3, hiding the user the requested object instance.

`__str__()`

Returns a human-readable representation of CPE Name.

Returns Representation of CPE Name as string

Return type `string`

CPE2_3_URI class

class `cpe.cpe2_3_uri.CPE2_3_URI` (*cpe_str*, **args*, ***kwargs*)

Implementation of binding style URI of version 2.3 of CPE specification.

A CPE Name is a percent-encoded URI with each name starting with the prefix (the URI scheme name) 'cpe:'.

Each platform can be broken down into many distinct parts. A CPE Name specifies a simple part and is used to identify any platform that matches the description of that part.

The distinct parts are:

- Hardware part: the physical platform supporting the IT system.
- Operating system part: the operating system controls and manages the IT hardware.
- Application part: software systems, services, servers, and packages installed on the system.

CPE Name syntax:

`cpe:{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}`

__getitem__ (*i*)

Returns the *i*'th component name of CPE Name.

Parameters *i* (*int*) – component index to find

Returns component string found

Return type `CPEComponent`

Exception `IndexError` - index not found in CPE Name

__len__ ()

Returns the number of components of CPE Name.

Returns count of components of CPE Name

Return type `int`

static **__new__** (*cpe_str*, **args*, ***kwargs*)

Create a new CPE Name of version 2.3 with URI style.

Parameters *cpe_str* (*string*) – CPE Name string

Returns CPE object of version 2.3 of CPE specification with URI style.

Return type `CPE2_3_URI`

as_wfn ()

Returns the CPE Name as Well-Formed Name string of version 2.3. If edition component is not packed, only shows the first seven components, otherwise shows all.

Returns CPE Name as WFN string

Return type `string`

Exception `TypeError` - incompatible version

get_attribute_values (*att_name*)

Returns the values of attribute "att_name" of CPE Name. By default a only element in each part.

Parameters *att_name* (*string*) – Attribute name to get

Returns List of attribute values

Return type `list`

Exception ValueError - invalid attribute name

CPE2_3_WFN class

class `cpe.cpe2_3_wfn.CPE2_3_WFN` (*cpe_str*, **args*, ***kwargs*)

Implementation of WFN of version 2.3 of CPE specification.

A CPE Name is a percent-encoded WFN with each name starting with the prefix 'wfn:'.

Each platform can be broken down into many distinct parts. A CPE Name specifies a simple part and is used to identify any platform that matches the description of that part.

The distinct parts are:

- Hardware part: the physical platform supporting the IT system.
- Operating system part: the operating system controls and manages the IT hardware.
- Application part: software systems, services, servers, and packages installed on the system.

CPE Name syntax: wfn:[a1=v1, a2=v2, ..., an=vn]

Only the following attributes SHALL be permitted in a WFN attribute-value pair:

- 1.part
- 2.vendor
- 3.product
- 4.version
- 5.update
- 6.edition
- 7.language
- 8.sw_edition
- 9.target_sw
- 10.target_hw
- 11.other

static `__new__` (*cpe_str*, **args*, ***kwargs*)

Create a new CPE Name of version 2.3 with WFN style.

Parameters `cpe_str` (*string*) – CPE Name string

Returns CPE object of version 2.3 of CPE specification with WFN style.

Return type `CPE2_3_WFN`

get_attribute_values (*att_name*)

Returns the values of attribute “att_name” of CPE Name. By default a only element in each part.

Parameters `att_name` (*string*) – Attribute name to get

Returns List of attribute values

Return type `list`

Exception ValueError - invalid attribute name

CPE2_3_FS class

class `cpe.cpe2_3_fs.CPE2_3_FS` (*cpe_str*, **args*, ***kwargs*)

Implementation of binding style formatted string of version 2.3 of CPE specification.

Each name starts with the prefix 'cpe:2.3:'.

Each platform can be broken down into many distinct parts. A CPE Name specifies a simple part and is used to identify any platform that matches the description of that part.

The distinct parts are:

- Hardware part: the physical platform supporting the IT system.
- Operating system part: the operating system controls and manages the IT hardware.
- Application part: software systems, services, servers, and packages installed on the system.

CPE Name syntax:

cpe:2.3:part:vendor:product:version:update:edition:language:sw_edition:target_sw:target_hw:other

__len__ ()

Returns the number of components of CPE Name. This CPE Name always have eleven components set.

Returns count of components of CPE Name

Return type `int`

static **__new__** (*cpe_str*, **args*, ***kwargs*)

Create a new CPE Name of version 2.3 with formatted string style.

Parameters **cpe_str** (*string*) – CPE Name string

Returns CPE object of version 2.3 of CPE specification with formatted string style.

Return type `CPE2_3_FS`

get_attribute_values (*att_name*)

Returns the values of attribute “att_name” of CPE Name. By default a only element in each part.

Parameters **att_name** (*string*) – Attribute name to get

Returns List of attribute values

Return type `list`

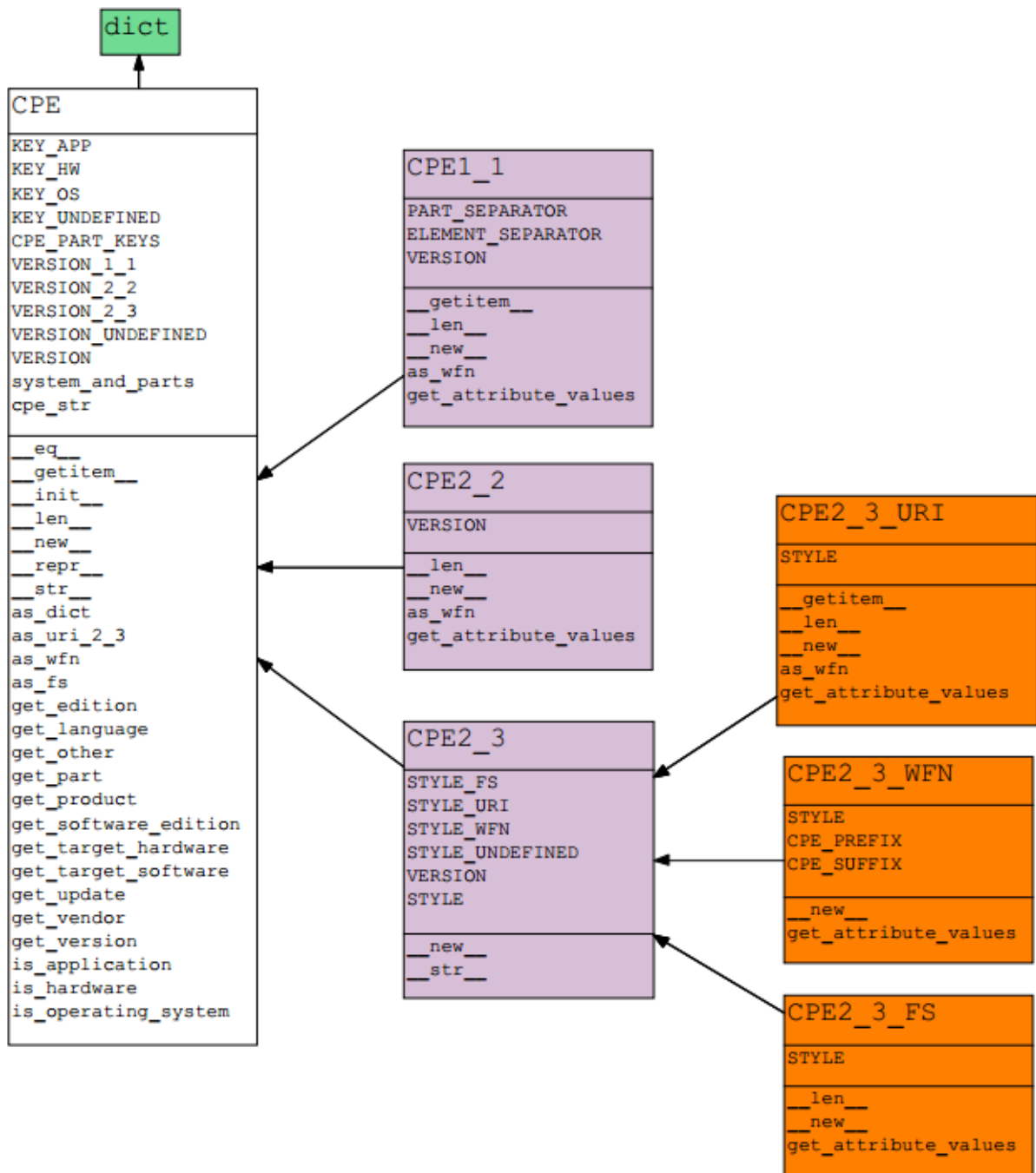
Exception `ValueError` - invalid attribute name

4.1.2 Class diagram

The class diagrams of CPE hierarchy are available in multiple format: PNG, PDF and PYNS (output format of PyNSource tool). Listed below are the download links of these diagrams and their formats:

- Classes with public and private attributes and methods:
 - Formato PNG
 - Formato PDF
 - Formato PYNS
- Classes with only public attributes and methods:
 - Formato PNG
 - Formato PDF
 - Formato PYNS

Next image presents the public attributes and methods of CPE classes:



4.2 CPESet hierarchy

This section contains the classes associated with the name matching algorithm of versions of CPE specification implemented in this package.

4.2.1 Class list

CPESet class

class `cpe.cpeset.CPESet`

Represents a set of CPE Names.

This class allows:

- create a set of CPE Names.
- match a CPE Name against a set of CPE Names.

__getitem__ (*i*)

Returns the *i*'th CPE Name of set.

Parameters *i* (*int*) – CPE Name index to find

Returns CPE Name found

Return type *CPE*

Exception IndexError - list index out of range

__init__ ()

Creates an empty set of CPE Names.

Returns None

__len__ ()

Returns the count of CPE Names of set.

Returns count of components of CPE Name

Return type *int*

TEST: empty set

```
>>> from .cpeset1_1 import CPESet1_1
>>> s = CPESet1_1()
>>> len(s)
0
```

__str__ ()

Returns a human-readable representation of CPE set.

Returns Representation of CPE set as string

Return type *string*

__weakref__

list of weak references to the object (if defined)

append (*cpe*)

Adds a CPE Name to the set if not already.

Parameters *cpe* (*CPE*) – CPE Name to store in set

Returns None

Exception NotImplementedError - Method not implemented

name_match (*cpe*)

Accepts a set of known instances of CPE Names and a candidate CPE Name, and returns 'True' if the candidate can be shown to be an instance based on the content of the known instances. Otherwise, it returns 'False'.

Parameters

- **self** (*CPESet*) – A set of *m* known CPE Names $K = \{K_1, K_2, \dots, K_m\}$.

- **cpe** (CPE) – A candidate CPE Name X.

Returns True if X matches K, otherwise False.

Return type boolean

CPESet1_1 class

class `cpe.cpeset1_1.CPESet1_1`

Represents a set of CPE Names.

This class allows:

- create set of CPE Names.
- match a CPE element against a set of CPE Names.

append (cpe)

Adds a CPE Name to the set if not already.

Parameters **cpe** (CPE) – CPE Name to store in set

Returns None

Exception ValueError - invalid version of CPE Name

TEST:

```
>>> from .cpeset1_1 import CPESet1_1
>>> from .cpe1_1 import CPE1_1
>>> uri1 = 'cpe://microsoft:windows:xp!vista'
>>> c1 = CPE1_1(uri1)
>>> s = CPESet1_1()
>>> s.append(c1)
```

name_match (cpe)

Accepts a set of known instances of CPE Names and a candidate CPE Name, and returns 'True' if the candidate can be shown to be an instance based on the content of the known instances. Otherwise, it returns 'False'.

Parameters

- **self** (CPESet) – A set of m known CPE Names $K = \{K_1, K_2, \dots, K_m\}$.
- **cpe** (CPE) – A candidate CPE Name X.

Returns True if X matches K, otherwise False.

Return type boolean

TEST: matching with identical CPE in set

```
>>> from .cpe1_1 import CPE1_1
>>> from .cpeset1_1 import CPESet1_1
>>> uri1 = 'cpe://microsoft:windows:xp!vista'
>>> uri2 = 'cpe://cisco:3825;cisco:2:44/cisco:ios:12.3:enterprise'
>>> c1 = CPE1_1(uri1)
>>> c2 = CPE1_1(uri2)
>>> s = CPESet1_1()
>>> s.append(c1)
>>> s.append(c2)
>>> s.name_match(c2)
True
```

CPESet2_2 class

class `cpe.cpeset2_2.CPESet2_2`

Represents a set of CPE Names.

This class allows:

- create set of CPE Names.
- match a CPE element against a set of CPE Names.

append (*cpe*)

Adds a CPE Name to the set if not already.

Parameters `cpe` (CPE) – CPE Name to store in set

Returns None

Exception ValueError - invalid version of CPE Name

TEST:

```
>>> from .cpeset2_2 import CPESet2_2
>>> from .cpe2_2 import CPE2_2
>>> uri1 = 'cpe:/h:hp'
>>> c1 = CPE2_2(uri1)
>>> s = CPESet2_2()
>>> s.append(c1)
```

name_match (*cpe*)

Accepts a set of known instances of CPE Names and a candidate CPE Name, and returns 'True' if the candidate can be shown to be an instance based on the content of the known instances. Otherwise, it returns 'False'.

Parameters

- **self** (CPESet) – A set of m known CPE Names $K = \{K_1, K_2, \dots, K_m\}$.
- **cpe** (CPE) – A candidate CPE Name X.

Returns True if X matches K, otherwise False.

Return type boolean

TEST: matching with ANY values explicit

```
>>> from .cpe2_2 import CPE2_2
>>> uri1 = 'cpe:/o:microsoft:windows:vista'
>>> uri2 = 'cpe:/o:cisco:ios:12.3:enterprise'
>>> c1 = CPE2_2(uri1)
>>> c2 = CPE2_2(uri2)
>>> s = CPESet2_2()
>>> s.append(c1)
>>> s.append(c2)
>>> uri3 = 'cpe:/o:microsoft::vista'
>>> c3 = CPE2_2(uri3)
>>> s.name_match(c3)
True
```

CPESet2_3 class

class `cpe.cpeset2_3.CPESet2_3`

Represents a set of CPEs.

This class allows:

- create set of CPE elements.

- match a CPE element against a set of CPE elements.

append (*cpe*)

Adds a CPE element to the set if not already. Only WFN CPE Names are valid, so this function converts the input CPE object of version 2.3 to WFN style.

Parameters **cpe** (*CPE*) – CPE Name to store in set

Returns None

Exception ValueError - invalid version of CPE Name

classmethod **compare_wfns** (*source, target*)

Compares two WFNs and returns a generator of pairwise attribute-value comparison results. It provides full access to the individual comparison results to enable use-case specific implementations of novel name-comparison algorithms.

Compare each attribute of the Source WFN to the Target WFN:

Parameters

- **source** (*CPE2_3_WFN*) – first WFN CPE Name
- **target** (*CPE2_3_WFN*) – seconds WFN CPE Name

Returns generator of pairwise attribute comparison results

Return type generator

classmethod **cpe_disjoint** (*source, target*)

Compares two WFNs and returns True if the set-theoretic relation between the names is DISJOINT.

Parameters

- **source** (*CPE2_3_WFN*) – first WFN CPE Name
- **target** (*CPE2_3_WFN*) – seconds WFN CPE Name

Returns True if the set relation between source and target is DISJOINT, otherwise False.

Return type boolean

classmethod **cpe_equal** (*source, target*)

Compares two WFNs and returns True if the set-theoretic relation between the names is EQUAL.

Parameters

- **source** (*CPE2_3_WFN*) – first WFN CPE Name
- **target** (*CPE2_3_WFN*) – seconds WFN CPE Name

Returns True if the set relation between source and target is EQUAL, otherwise False.

Return type boolean

classmethod **cpe_subset** (*source, target*)

Compares two WFNs and returns True if the set-theoretic relation between the names is (non-proper) SUBSET.

Parameters

- **source** (*CPE2_3_WFN*) – first WFN CPE Name
- **target** (*CPE2_3_WFN*) – seconds WFN CPE Name

Returns True if the set relation between source and target is SUBSET, otherwise False.

Return type boolean

classmethod **cpe_superset** (*source, target*)

Compares two WFNs and returns True if the set-theoretic relation between the names is (non-proper) SUPERSET.

Parameters

- **source** (`CPE2_3_WFN`) – first WFN CPE Name
- **target** (`CPE2_3_WFN`) – seconds WFN CPE Name

Returns True if the set relation between source and target is SUPERSET, otherwise False.

Return type boolean

name_match (*wfn*)

Accepts a set of CPE Names K and a candidate CPE Name X. It returns 'True' if X matches any member of K, and 'False' otherwise.

Parameters

- **self** (`CPESet`) – A set of m known CPE Names $K = \{K1, K2, \dots, Km\}$.
- **cpe** (`CPE`) – A candidate CPE Name X.

Returns True if X matches K, otherwise False.

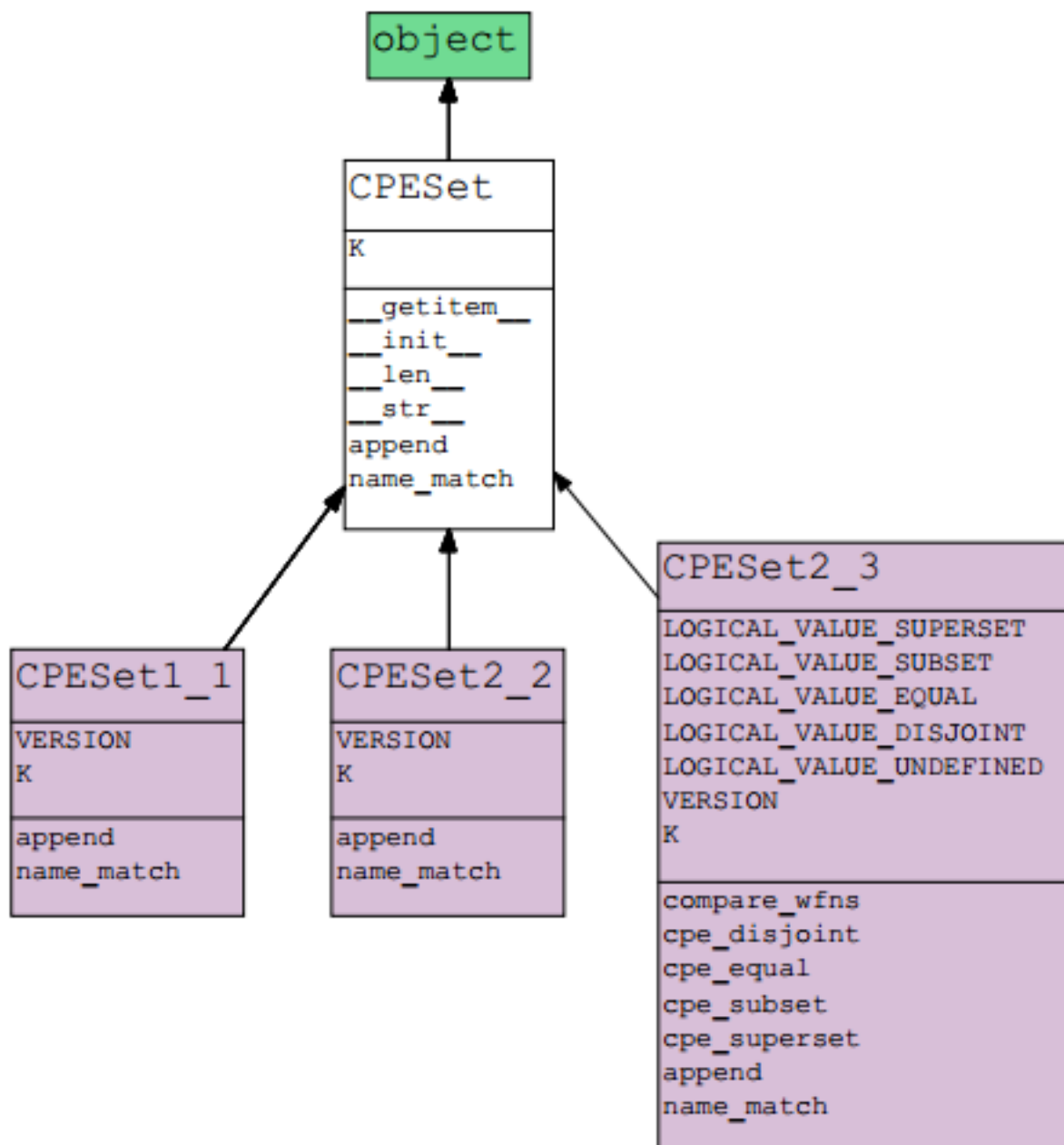
Return type boolean

4.2.2 Class diagram

The class diagrams of CPESet hierarchy are available in multiple format: PNG, PDF and PYNS (output format of PyNSource tool). Listed below are the download links of these diagrams and their formats:

- Classes with public and private attributes and methods:
 - Format PNG
 - Format PDF
 - Format PYNS
- Classes with only public attributes and methods:
 - Format PNG
 - Format PDF
 - Format PYNS

Next image presents the public attributes and methods of CPESet classes:



4.3 CPELanguage hierarchy

This section contains the classes associated with the language matching algorithm of versions of CPE specification implemented in this package.

4.3.1 Class list

CPELanguage class

class `cpe.cpelang.CPELanguage` (*expression, isFile=False*)

Represents an expression in the CPE Language.

This class allows match a CPE element against an expression in the CPE Language, that is, a XML document format for binding descriptive prose and diagnostic test to a CPE Name (CPE Description Format).

`__init__(expression, isFile=False)`

Create an object that contains the input expression in the CPE Language (a set of CPE Names) and the DOM tree associated with expression.

Parameters

- **expression** (*string*) – XML content in string or a path to XML file
- **isFile** (*string*) – indicates whether expression is a XML file or XML content string

Returns None

`__str__()`

Returns a human-readable representation of CPE Language expression.

Returns Representation of CPE Language expression as string

Return type *string*

`__weakref__`

list of weak references to the object (if defined)

`language_match(cpeset, cpe_dom=None)`

Accepts a set of known CPE Names and an expression in the CPE language, and delivers the answer True if the expression matches with the set. Otherwise, it returns False.

Parameters

- **self** (*CPELanguage*) – An expression in the CPE Language, represented as the XML info set for the platform element.
- **cpeset** (*CPESet*) – CPE set object to match with self expression.
- **cpe_dom** (*string*) – An expression in the CPE Language, represented as DOM tree.

Returns True if self expression can be satisfied by language matching against cpeset, False otherwise.

Return type boolean

Exception NotImplementedError - Method not implemented

CPELanguage2_2 class

`class cpe.cpelang2_2.CPELanguage2_2(expression, isFile=False)`

Represents an expression in the CPE Language.

This class allows match a CPE element against an expression in the CPE Language, that is, a XML document format for binding descriptive prose and diagnostic test to a CPE name (CPE Description Format).

`language_match(cpeset, cpe_dom=None)`

Accepts a set of known CPE Names and an expression in the CPE language, and delivers the answer True if the expression matches with the set. Otherwise, it returns False.

Parameters

- **self** (*CPELanguage*) – An expression in the CPE Applicability Language, represented as the XML info set for the platform element.
- **cpeset** (*CPESet*) – CPE set object to match with self expression.
- **cpe_dom** (*string*) – An expression in the CPE Applicability Language, represented as DOM tree.

Returns True if self expression can be satisfied by language matching against cpeset, False otherwise.

Return type boolean

CPELanguage2_3 class

class `cpe.cpelang2_3.CPELanguage2_3` (*expression, isFile=False*)

Represents an expression in the CPE Language.

This class allows match a CPE element against an expression in the CPE Language, that is, a XML document format for binding descriptive prose and diagnostic test to a CPE name (CPE Description Format).

language_match (*cpeset, cpel_dom=None*)

Accepts a set of known CPE Names and an expression in the CPE language, and delivers the answer True if the expression matches with the set. Otherwise, it returns False.

Parameters

- **self** (`CPELanguage`) – An expression in the CPE Applicability Language, represented as the XML info set for the platform element.
- **cpeset** (`CPESet`) – CPE set object to match with self expression.
- **cpel_dom** (*string*) – An expression in the CPE Applicability Language, represented as DOM tree.

Returns True if self expression can be satisfied by language matching against cpeset, False otherwise.

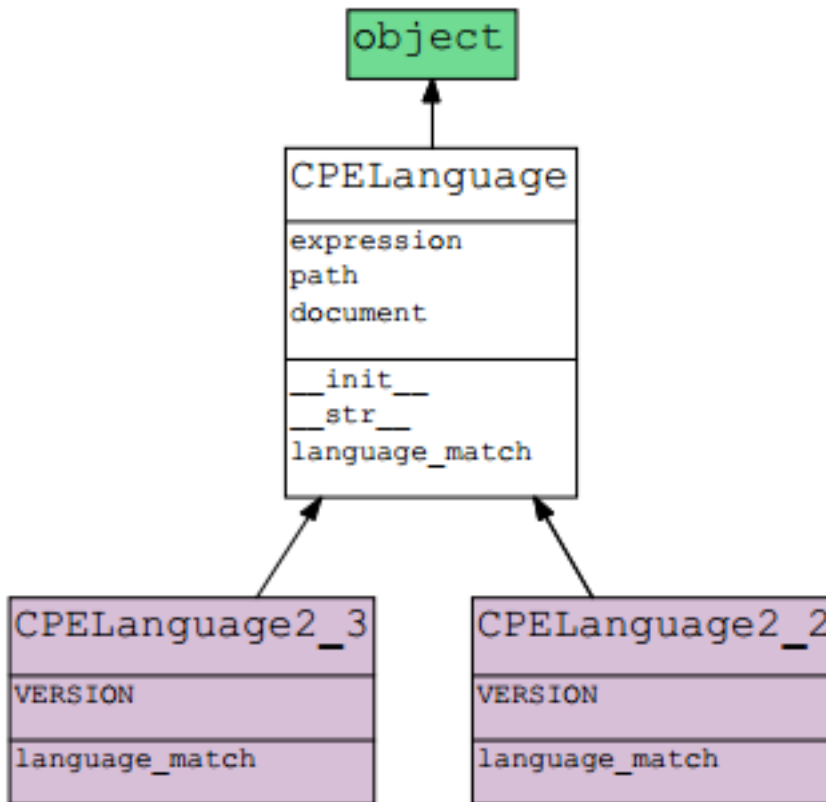
Return type boolean

4.3.2 Class diagram

The class diagrams of CPELanguage hierarchy are available in multiple format: PNG, PDF and PYNS (output format of PyNSource tool). Listed below are the download links of these diagrams and their formats:

- Classes with public and private attributes and methods:
 - [Formato PNG](#)
 - [Formato PDF](#)
 - [Formato PYNS](#)
- Classes with only public attributes and methods:
 - [Formato PNG](#)
 - [Formato PDF](#)
 - [Formato PYNS](#)

Next image presents the public attributes and methods of CPELanguage classes:



4.4 CPEComponent hierarchy

This section contains the classes associated with the types of components of versions of CPE Names implemented in this package: logical and simple.

4.4.1 Class list

The generic component class is:

CPEComponent class

class `cpe.comp.cpecomp.CPEComponent` (*comp_str*)

Represents a generic component of CPE Name, compatible with the components of all versions of CPE specification.

__contains__ (*item*)

Returns True if item is included in set of values of self.

Parameters *item* (`CPEComponent`) – component to find in self

Returns True if item is included in set of self, otherwise False

Return type boolean

__eq__ (*other*)

Returns True if other (first element of operation) and self (second element of operation) are equal components, false otherwise.

Parameters *other* (`CPEComponent`) – component to compare

Returns True if other == self, False otherwise

Return type boolean

__init__ (*comp_str*)

Store the value of component.

Parameters **comp_str** (*string*) – value of component value

Returns None

__ne__ (*other*)

Returns True if other (first element of operation) and self (second element of operation) are not equal components, false otherwise.

Parameters **other** (*CPEComponent*) – component to compare

Returns True if other != self, False otherwise

Return type boolean

__repr__ ()

Returns a unambiguous representation of CPE component.

Returns Representation of CPE component as string

Return type *string*

__weakref__

list of weak references to the object (if defined)

classmethod is_valid_attribute (*att_name*)

Check if input attribute name is correct.

Parameters **att_name** (*string*) – attribute name to check

Returns True is attribute name is valid, otherwise False

Return type boolean

TEST: a wrong attribute

```
>>> from .cpecomp import CPEComponent
>>> att = CPEComponent.ATT_PRODUCT
>>> CPEComponent.is_valid_attribute(att)
True
```

The logical components classes are:

CPEComponentLogical class

class `cpe.comp.cpecomp_logical.CPEComponentLogical` (*comp_str*)

Represents a generic logical component of CPE name, compatible with the components of all versions of CPE specification.

__contains__ (*item*)

Returns True if item is included in set of values of self.

Parameters **item** (*CPEComponent*) – component to find in self

Returns True if item is included in set of self

Return type boolean

__eq__ (*other*)

Returns True if other (first element of operation) and self (second element of operation) are equal components, false otherwise.

Parameters **other** (*CPEComponent*) – component to compare

Returns True if other == self, False otherwise

Return type boolean

Exception NotImplementedError - class method not implemented

`__str__()`

Returns a human-readable representation of CPE component.

Returns Representation of CPE component as string

Return type string

Exception NotImplementedError - class method not implemented

CPEComponentAnyValue class

class `cpe.comp.cpecomp_anyvalue.CPEComponentAnyValue`

Represents a component of CPE name without a particular value, compatible with the components of all versions of CPE specification.

For example, in version 2.3 of CPE specification, an component “any value” is other attribute in CPE name `cpe:2.3:a:microsoft:windows:xp:*:*:*:*:*`.

`__eq__(other)`

Returns True if other (first element of operation) and self (second element of operation) are equal components, false otherwise.

Parameters `other` (`CPEComponent`) – component to compare

Returns True if other == self, False otherwise

Return type boolean

`__init__()`

Initializes the component.

`__str__()`

Returns a human-readable representation of CPE component.

Returns Representation of CPE component as string

Return type string

CPEComponentEmpty class

class `cpe.comp.cpecomp_empty.CPEComponentEmpty`

Represents an empty component of CPE name, compatible with the components of all versions of CPE specification.

For example, in version 1.1 of CPE specification, an empty component is version attribute in CPE name `cpe:/microsoft:windows::sp2`.

`__eq__(other)`

Returns True if other (first element of operation) and self (second element of operation) are equal components, false otherwise.

Parameters `other` (`CPEComponent`) – component to compare

Returns True if other == self, False otherwise

Return type boolean

`__init__()`

Initializes the component.

`__str__()`

Returns a human-readable representation of CPE component.

Returns Representation of CPE component as string

Return type `string`

CPEComponentNotApplicable class

class `cpe.comp.cpecomp_notapplicable.CPEComponentNotApplicable`

Represents a component of CPE name with a not applicable value, compatible with the components of all versions of CPE specification.

For example, in version 2.3 of CPE specification, an component “not applicable” is update attribute in CPE name `cpe:/a:microsoft:windows:me:-`.

__contains__ (*item*)

Returns True if item is included in set of values of self.

Parameters *item* (`CPEComponent`) – component to find in self

Returns True if item is included in set of self

Return type `boolean`

__eq__ (*other*)

Returns True if other (first element of operation) and self (second element of operation) are equal components, false otherwise.

Parameters *other* (`CPEComponent`) – component to compare

Returns True if other == self, False otherwise

Return type `boolean`

__init__ ()

Initializes the component.

__str__ ()

Returns a human-readable representation of CPE component.

Returns Representation of CPE component as string

Return type `string`

CPEComponentUndefined class

class `cpe.comp.cpecomp_undefined.CPEComponentUndefined`

Represents an undefined component of CPE name, compatible with the components of all versions of CPE specification.

For example, in version 1.1 of CPE specification, an undefined component is edition attribute in CPE name `cpe:/microsoft:windows:xp`.

__contains__ (*item*)

Returns True if item is included in set of values of self.

Parameters *item* (`CPEComponent`) – component to find in self

Returns True if item is included in set of self

Return type `boolean`

__eq__ (*other*)

Returns True if other (first element of operation) and self (second element of operation) are equal components, false otherwise.

Parameters *other* (`CPEComponent`) – component to compare

Returns True if other == self, False otherwise

__init__ ()

Initializes the component.

`__str__()`

Returns a human-readable representation of CPE component.

Returns Representation of CPE component as string

Return type `string`

The simple components classes are:

CPEComponentSimple class

class `cpe.comp.cpecomp_simple.CPEComponentSimple` (*comp_str*, *comp_att*)

Represents a generic string component of CPE name, compatible with the components of all versions of CPE specification.

`__init__` (*comp_str*, *comp_att*)

Store the value of component.

Parameters

- **comp_str** (*string*) – value of component value
- **comp_att** (*string*) – attribute associated with component value

Returns None

Exception ValueError - incorrect value of component

`__str__()`

Returns a human-readable representation of CPE component.

Returns Representation of CPE component as string

Return type `string`

as_fs ()

Returns the value of component encoded as formatted string.

Inspect each character in value of component. Certain nonalpha characters pass thru without escaping into the result, but most retain escaping.

Returns Formatted string associated with component

Return type `string`

as_uri_2_3 ()

Returns the value of component encoded as URI string.

Scans an input string *s* and applies the following transformations:

- Pass alphanumeric characters thru untouched
- Percent-encode quoted non-alphanumerics as needed
- Unquoted special characters are mapped to their special forms.

Returns URI string associated with component

Return type `string`

as_wfn ()

Returns the value of component encoded as Well-Formed Name (WFN) string.

Returns WFN string associated with component

Return type `string`

get_value ()

Returns the encoded value of component.

Returns The encoded value of component

Return type `string`

set_value (*comp_str*, *comp_att*)

Set the value of component. By default, the component has a simple value.

Parameters

- **comp_str** (*string*) – new value of component
- **comp_att** (*string*) – attribute associated with value of component

Returns None

Exception ValueError - incorrect value of component

CPEComponent1_1 class

class `cpe.comp.cpecomp1_1.CPEComponent1_1` (*comp_str*, *comp_att*)

Represents a component of version 1.1 of CPE specification.

TEST: simple value

```
>>> value = "microsoft"
>>> comp = CPEComponent1_1(value, CPEComponentSimple.ATT_VENDOR)
```

__contains__ (*item*)

Returns True if item is included in set of values of self.

Comparatives in name matching of version 1.1 of CPE:

`c = self._standard_value`

`d = item._standard_value`

IF `c` is empty THEN match True.

ELSE IF `c` is a singleton AND `c = d` THEN match True.

ELSE IF `c` has form `~v` AND `v != d` THEN match True.

ELSE IF `c` has form `v1!v2!...!vn` AND `v = d` for some `v` THEN match True.

ENDIF.

Parameters **item** (`CPEComponent`) – component to find in self

Returns True if item is included in set of self

Return type boolean

TEST: two different simple values

```
>>> comp1 = CPEComponent1_1('5.0', CPEComponentSimple.ATT_VERSION)
>>> comp2 = CPEComponent1_1('9.0', CPEComponentSimple.ATT_VERSION)
>>> comp1 in comp2
False
```

__repr__ ()

Returns a unambiguous representation of CPE component.

Returns Representation of CPE component as string

Return type `string`

as_fs()

Returns the value of component encoded as formatted string.

Inspect each character in value of component. Certain nonalpha characters pass thru without escaping into the result, but most retain escaping.

Returns Formatted string associated with the component

Return type `string`

TEST:

```
>>> val = 'xp!vista'
>>> comp1 = CPEComponent1_1(val, CPEComponentSimple.ATT_VERSION)
>>> comp1.as_fs()
'xp\\!vista'
```

as_uri_2_3()

Returns the value of component encoded as URI string.

Scans an input string `s` and applies the following transformations:

- Pass alphanumeric characters thru untouched
- Percent-encode quoted non-alphanumerics as needed
- Unquoted special characters are mapped to their special forms.

Returns URI string

Return type `string`

TEST:

```
>>> val = '#nvidi@'
>>> comp1 = CPEComponent1_1(val, CPEComponentSimple.ATT_VENDOR)
>>> comp1.as_uri_2_3()
'%23nvidi%40'
```

as_wfn()

Returns the value of component encoded as Well-Formed Name (WFN) string.

Returns WFN string

Return type `string`

TEST:

```
>>> val = 'xp!vista'
>>> comp1 = CPEComponent1_1(val, CPEComponentSimple.ATT_VERSION)
>>> comp1.as_wfn()
'xp\\!vista'
```

set_value(comp_str, comp_att)

Set the value of component. By default, the component has a simple value.

Parameters `comp_att` (*string*) – attribute associated with value of component

Returns None

Exception ValueError - incorrect value of component

TEST:

```
>>> val = 'xp!vista'
>>> val2 = 'sp2'
>>> att = CPEComponentSimple.ATT_VERSION
>>> comp1 = CPEComponent1_1(val, att)
>>> comp1.set_value(val2, att)
```

```
>>> comp1.get_value()
'sp2'
```

CPEComponent2_2 class

class `cpe.comp.cpecomp2_2.CPEComponent2_2 (comp_str, comp_att)`

Represents a component of version 2.2 of CPE specification.

TEST: simple value

```
>>> value = "microsoft"
>>> comp = CPEComponent2_2(value, CPEComponentSimple.ATT_VENDOR)
```

__repr__()

Returns a unambiguous representation of CPE component.

Returns Representation of CPE component as string

Return type `string`

CPEComponent2_3 class

class `cpe.comp.cpecomp2_3.CPEComponent2_3 (comp_str, comp_att)`

Represents a component of version 2.3 of CPE specification.

__repr__()

Returns a unambiguous representation of CPE component.

Returns Representation of CPE component as string

Return type `string`

CPEComponent2_3_URI class

class `cpe.comp.cpecomp2_3_uri.CPEComponent2_3_URI (comp_str, comp_att)`

Represents a component of version 2.3 of CPE specification with URI style.

CPEComponent2_3_edpacked class

class `cpe.comp.cpecomp2_3_uri_edpacked.CPEComponent2_3_URI_edpacked (comp_str)`

Represents a packd edition component of version 2.3 of CPE specification with URI style.

__init__ (comp_str)

Store the value of component.

Parameters `comp_str (string)` – value of component value

Returns None

Exception `ValueError` - incorrect value of component

set_value (comp_str)

Set the value of component.

Parameters `comp_str (string)` – value of component

Returns None

Exception `ValueError` - incorrect value of component

CPEComponent2_3_WFN class

class `cpe.comp.cpecomp2_3_wfn.CPEComponent2_3_WFN` (*comp_str*, *comp_att*)
Represents a component of version 2.3 of CPE specification with WFN style.

get_value ()

Returns the encoded value of component.

Returns encoded value of component

Return type `string`

set_value (*comp_str*, *comp_att*)

Set the value of component.

Parameters

- **comp_str** (*string*) – value of component
- **comp_att** (*string*) – attribute associated with comp_str

Returns None

Exception `ValueError` - incorrect value of component

CPEComponent2_3_FS class

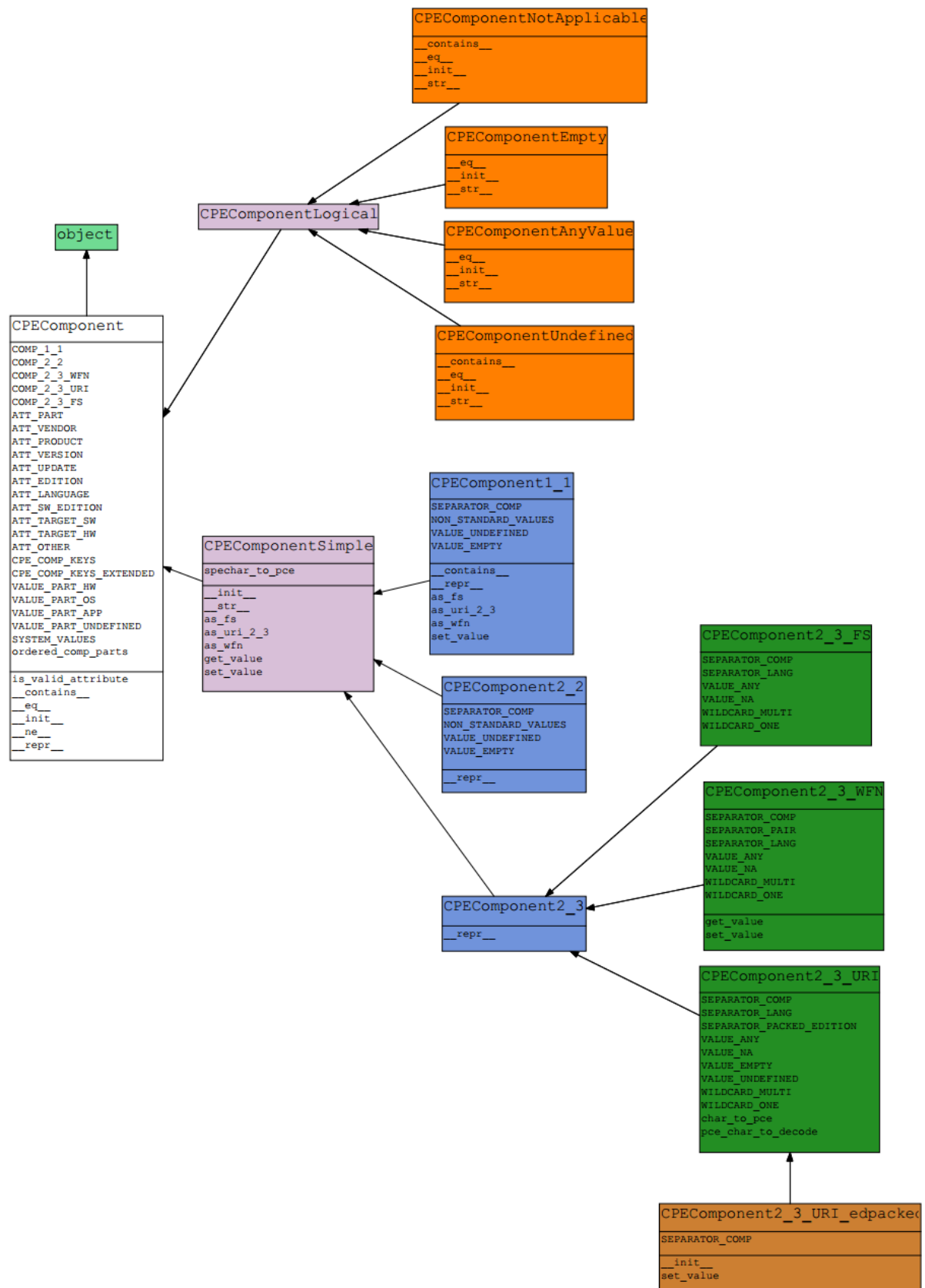
class `cpe.comp.cpecomp2_3_fs.CPEComponent2_3_FS` (*comp_str*, *comp_att*)
Represents a component of version 2.3 of CPE specification with URI style.

4.4.2 Class diagram

The class diagrams of CPE component hierarchy are available in multiple format: PNG, PDF and PYNS (output format of PyNSource tool). Listed below are the download links of these diagrams and their formats:

- Classes with public and private attributes and methods:
 - [Formato PNG](#)
 - [Formato PDF](#)
 - [Formato PYNS](#)
- Classes with only public attributes and methods:
 - [Formato PNG](#)
 - [Formato PDF](#)
 - [Formato PYNS](#)

Next image presents the public attributes and methods of CPE component classes:



4.5 Categories of main classes

The main classes of model can be grouped in four categories:

- Auto version (classes to create CPE Names without setting their version of CPE specification associated):
 - **cpe.py** (generic auto version class)
 - **cpe2_3.py** (auto version class of version 2.3)
- Manual version (classes to create CPE Names of particular version of CPE specification):
 - **cpe1_1.py** (version 1.1)
 - **cpe2_2.py** (version 2.2)
 - **cpe2_3_wfn.py** (version 2.3 with WFN style)
 - **cpe2_3_uri.py** (version 2.3 with URI style)
 - **cpe2_3_fs.py** (version 2.3 with formatted style style)
- CPE Name matching (classes to realize the name matching of CPE specification):
 - **cpeset1_1.py** (version 1.1)
 - **cpeset2_2.py** (version 2.2)
 - **cpeset2_3.py** (version 2.3)
- CPE Language matching (classes to realize the language matching of CPE specification):
 - **cpelang2_2.py** (version 2.2)
 - **cpelang2_3.py** (version 2.3)

Installation

Install the package using pip:

```
pip install cpe
```

or execute the setup.py file in package:

```
python setup.py install
```

Usage examples

This section explains with several examples how to use this package to create both CPE Names and platforms in a particular version of CPE specification.

6.1 Naming

To create a new CPE Name, the `cpe` package provides a generating class of CPE objects called `CPE`. It implements the factory pattern and receive two parameters: version of CPE specification and URI associated with CPE Name. Also, it is possible create a instance of a particular version of CPE Name directly using the class associated with the version.

In the following example, some CPE Names of different versions of CPE specification are created:

- Imports the class:

```
>>> from cpe import CPE
```

- Creates a CPE Name of version 1.1 with an operating system and an application parts, without setting the version directly (auto version):

```
>>> str11 = 'cpe://redhat:enterprise_linux:3:as/apache:httpd:2.0.52'
>>> c11 = CPE(str11)
```

- Creates a CPE Name of version 2.2 with an operating system where the version is set (manual version):

```
>>> str22 = 'cpe:/o:redhat:enterprise_linux:4:update4'
>>> c22 = CPE(str22, CPE.VERSION_2_2)
```

- Creates a CPE Name of version 2.3 (URI style) with an application system where the value of edition component is packed:

```
>>> str23_uri = 'cpe:/a:hp:insight_diagnostics:8::~~online~win2003~x64~'
>>> c23_uri = CPE(str23_uri)
```

- Creates a CPE Name of version 2.3 (WFN style) with an application system where some values have wild-cards:

```
>>> str23_wfn = 'wfn:[part="a", vendor="hp", product="?insight_diagnostics?", version="8\.*",
>>> c23_wfn = CPE(str23_wfn)
```

- Creates a CPE Name of version 2.3 (formatted string style) with a hardware system:

```
>>> str23_fs = 'cpe:2.3:h:cisco:ios:12.3:enterprise:~::~~::~~'
>>> c23_fs = CPE(str23_fs)
```

The `cpe` package provides methods to get the value of components of a CPE Name (these functions always return a string list) and identify the type of system associated with it (hardware, operating system or application):

```
>>> c11.get_product()           # Compound product attribute (v1.1)
['enterprise_linux', 'httpd']
>>> c22.get_update()           # Simple Update attribute (v2.2)
['update4']
>>> c23_uri.get_target_hardware() # Simple Target_hw attribute (v2.3, URI style)
['x64']
>>> c23_wfn.get_target_hardware() # Simple Target_hw attribute (v2.3, WFN style)
['"x32"']
>>> c23_wfn.get_target_software() # Target_sw attribute with logical value(v2.3, WFN style)
['ANY']
>>> c23_fs.is_hardware()        # Type of system (v2.3, formatted string style)
True
```

Finally, the `cpe` package contains methods to convert any CPE Name defined under a particular style (URI version 2.3, WFN or formatted string) in other different styles:

```
>>> c22.as_wfn()
'wfn:[part="o", vendor="redhat", product="enterprise_linux", version="4", update="update4"]'
>>> c23_uri.as_uri_2_3()
'cpe:/a:hp:insight_diagnostics:8::~~online~win2003~x64~'
>>> c23_wfn.as_fs()
'cpe:2.3:a:hp:?insight_diagnostics?:8.*:*:*:*:*:*:*:x32:*'
```

6.2 Name matching

To create a set of CPE Name the package `cpe` provides the `CPESetX_Y` class, where `X_Y` is the target version of CPE specification. The `name_match` function of set allows do the name matching of CPE specification.

In the following example, a set of CPE Names of version 2.2 is created and the name matching is realized:

- Imports the classes of version:

```
>>> from cpe.cpe2_2 import CPE2_2
>>> from cpe.cpeset2_2 import CPESet2_2
```

- Creates the CPE Names of target system:

```
>>> c1 = CPE2_2('cpe:/o:microsoft:windows_2000::sp3:pro')
>>> c2 = CPE2_2('cpe:/a:microsoft:ie:5.5')
```

- Creates a set that contains the above CPE Names (known instances): $K = \{\text{"cpe:/o:microsoft:windows 2000::sp3:pro"}, \text{"cpe:/a:microsoft:ie:5.5"}\}$:

```
>>> K = CPESet2_2()
>>> K.append(c1)
>>> K.append(c2)
```

- Create the candidate CPE Name. It represents a rule in a security guidance checklist describes some settings to check on a system running Microsoft Windows 2000: X = "cpe:/o:microsoft:windows_2000":

```
>>> X = CPE2_2('cpe:/o:microsoft:windows_2000')
```

- Does the name matching:

```
>>> K.name_match(X)
True
```

There are three components in X: C1=o, C2=microsoft, C3=windows_2000. Each component matches the corresponding component of the first CPE Name in K. So, the algorithm returns true and the rule can be applied to the target system.

6.3 Language matching

To create an expression of CPE Language the `cpe` package provides the `CPELanguageX_Y` class, where `X_Y` is the version of CPE specification used. The `language_match` function of class allows do the language matching of CPE specification.

In the following example, an expression of CPE Language of version 2.2 is created and the language matching is done:

- Imports the classes of version:

```
>>> from cpe import CPE
>>> from cpe.cpeset2_2 import CPESet2_2
>>> from cpe.cpelang2_2 import CPELanguage2_2
```

- Creates the CPE Names of target system:

```
>>> c1 = CPE('cpe:/o:sun:solaris:5.9:::en-us', CPE.VERSION_2_2)
>>> c2 = CPE('cpe:/a:bea:weblogic:8.1', CPE.VERSION_2_2)
```

- Creates a set that contains the above CPE Names (known instances): `K = {"cpe:/o:sun:sunos:5.9:::en-us", "cpe:/a:bea:weblogic:8.1"}:`

```
>>> K = CPESet2_2()
>>> K.append(c1)
>>> K.append(c2)
```

- Creates the expression in XML of candidate CPE Language statement:

X = <cpe:platform id="123">

<cpe:title>Sun Solaris 5.8 or 5.9 with BEA Weblogic 8.1 installed</cpe:title>

<cpe:logical-test operator="AND" negate="FALSE">

<cpe:logical-test operator="OR" negate="FALSE">

<cpe:fact-ref name="cpe:/o:sun:solaris:5.8" />

<cpe:fact-ref name="cpe:/o:sun:solaris:5.9" />

</cpe:logical-test>

<cpe:fact-ref name="cpe:/a:bea:weblogic:8.1" />

</cpe:logical-test>

</cpe:platform>

```
>>> It is necessary specify the "cpe:platform-specification" tag
>>> document = '''<?xml version="1.0" encoding="UTF-8"?><cpe:platform-specification xmlns:cpe
```

- Does the language matching:

```
>>> X = CPELanguage2_2(document)
>>> X.language_match(K)
True
```

Important issues

- The **auto version classes** receive an CPE Name and try to find out what version is associated.
- The **functions to get the values of attributes of a CPE Name** always return a list of string. That is so because the attributes of version 1.1 of CPE specification can be linked with several system and elements. For example, the attribute *vendor* in CPE Name *cpe://sun:sunos:5.9/bea:weblogic:8.1/mysql:server:5.0* get three values: *sun*, *bea* and *mysql*.
- The **not logical values of the attributes in version 2.3** of CPE specification always start and end with double quotes. For example, the value of attribute *product* in CPE Name *wfn:[part="a", vendor="microsoft", product="internet_explorer", version="8", update="beta"]* is *"internet_explorer"*, not *internet_explorer* without double quotes.
- Some **CPE Names of version 1.1 with several systems or elements defined** cannot convert into other CPE versions, for example, the CPE Name *cpe://sun:sunos:5.9/bea:weblogic:8.1/mysql:server:5.0*
- **Comparing a CPE Name of version 1.1 with others**, if versions are incompatible, then the return value is *False* instead of raising an exception.
- The methods `ovalcheck()` and `ocilcheck()` of `CPELanguage2_3` class is **not implemented**.
- The language attribute of CPE Names only allow the normal language tags according to the shortest ISO 639 code in language part and the ISO 3166-1 and UN M.49 code in region part. The extended, registered or reserved subtags are not supported.

Bugtracker

If you have any suggestions, bug reports or annoyances please report them to the issue tracker at <https://github.com/nlp0inter/cpe/issues>

TODO

- Implement methods *ovalcheck* and *ocilcheck* of `CPELanguage2_3` class.
- Implement versions 2.0 and 2.1 of CPE specification.
- Implement methods *as_uri_1_1* and *as_uri_2_2* to convert any CPE Name into a CPE Name of versions 1.1 and 2.2 respectively.

References

- [1] CPE: <http://scap.nist.gov/specifications/cpe/>
- [2] About CPE: <http://cpe.mitre.org/about/>
- [3] CPE Archive: <http://cpe.mitre.org/cpe/archive/>
- [4] CPE 1.1: http://cpe.mitre.org/specification/1.1/cpe-specification_1.1.pdf
- [5] CPE 2.2: http://cpe.mitre.org/specification/2.2/cpe-specification_2.2.pdf
- [6] CPE 2.3 - Naming Specification: <http://csrc.nist.gov/publications/nistir/ir7695/NISTIR-7695-CPE-Naming.pdf>
- [7] CPE 2.3 - Name Matching Specification: <http://csrc.nist.gov/publications/nistir/ir7696/NISTIR-7696-CPE-Matching.pdf>
- [8] CPE 2.3 - Applicability Language Specification: <http://csrc.nist.gov/publications/nistir/ir7698/NISTIR-7698-CPE-Language.pdf>

Indices and tables

- `genindex`
- `search`

Symbols

__contains__() (cpe.comp.cpecomp.CPEComponent method), 27
 __contains__() (cpe.comp.cpecomp1_1.CPEComponent1_1 method), 32
 __contains__() (cpe.comp.cpecomp_logical.CPEComponentLogical method), 28
 __contains__() (cpe.comp.cpecomp_notapplicable.CPEComponentNotApplicable method), 30
 __contains__() (cpe.comp.cpecomp_undefined.CPEComponentUndefined method), 30
 __eq__() (cpe.comp.cpecomp.CPEComponent method), 27
 __eq__() (cpe.comp.cpecomp_anyvalue.CPEComponentAnyValue method), 29
 __eq__() (cpe.comp.cpecomp_empty.CPEComponentEmpty method), 29
 __eq__() (cpe.comp.cpecomp_logical.CPEComponentLogical method), 28
 __eq__() (cpe.comp.cpecomp_notapplicable.CPEComponentNotApplicable method), 30
 __eq__() (cpe.comp.cpecomp_undefined.CPEComponentUndefined method), 30
 __eq__() (cpe.cpe.CPE method), 9
 __getitem__() (cpe.cpe.CPE method), 9
 __getitem__() (cpe.cpe1_1.CPE1_1 method), 12
 __getitem__() (cpe.cpe2_3_uri.CPE2_3_URI method), 15
 __getitem__() (cpe.cpeset.CPESet method), 19
 __init__() (cpe.comp.cpecomp.CPEComponent method), 28
 __init__() (cpe.comp.cpecomp2_3_uri_edpacked.CPEComponent2_3_URI_edpacked method), 34
 __init__() (cpe.comp.cpecomp_anyvalue.CPEComponentAnyValue method), 29
 __init__() (cpe.comp.cpecomp_empty.CPEComponentEmpty method), 29
 __init__() (cpe.comp.cpecomp_notapplicable.CPEComponentNotApplicable method), 30
 __init__() (cpe.comp.cpecomp_simple.CPEComponentSimple method), 31
 __init__() (cpe.comp.cpecomp_undefined.CPEComponentUndefined method), 30
 __init__() (cpe.cpe.CPE method), 10
 __init__() (cpe.cpelang.CPELanguage method), 24
 __init__() (cpe.cpeset.CPESet method), 19
 __len__() (cpe.cpe.CPE method), 10
 __len__() (cpe.cpe1_1.CPE1_1 method), 13
 __len__() (cpe.cpe2_2.CPE2_2 method), 14
 __len__() (cpe.cpe2_3_fs.CPE2_3_FS method), 17
 __len__() (cpe.cpe2_3_uri.CPE2_3_URI method), 15
 __len__() (cpe.cpeset.CPESet method), 19
 __ne__() (cpe.comp.cpecomp.CPEComponent method), 28
 __new__() (cpe.cpe.CPE static method), 10
 __new__() (cpe.cpe1_1.CPE1_1 static method), 13
 __new__() (cpe.cpe2_2.CPE2_2 static method), 14
 __new__() (cpe.cpe2_3.CPE2_3 static method), 14
 __new__() (cpe.cpe2_3_fs.CPE2_3_FS static method), 17
 __new__() (cpe.cpe2_3_uri.CPE2_3_URI static method), 15
 __new__() (cpe.cpe2_3_wfn.CPE2_3_WFN static method), 16
 __repr__() (cpe.comp.cpecomp.CPEComponent method), 28
 __repr__() (cpe.comp.cpecomp1_1.CPEComponent1_1 method), 32
 __repr__() (cpe.comp.cpecomp2_2.CPEComponent2_2 method), 34
 __repr__() (cpe.comp.cpecomp2_3.CPEComponent2_3 method), 34
 __repr__() (cpe.cpe.CPE method), 10
 __str__() (cpe.comp.cpecomp_anyvalue.CPEComponentAnyValue method), 29
 __str__() (cpe.comp.cpecomp_empty.CPEComponentEmpty method), 29
 __str__() (cpe.comp.cpecomp_logical.CPEComponentLogical method), 29
 __str__() (cpe.comp.cpecomp_notapplicable.CPEComponentNotApplicable method), 30
 __str__() (cpe.comp.cpecomp_simple.CPEComponentSimple method), 31
 __str__() (cpe.comp.cpecomp_undefined.CPEComponentUndefined method), 31
 __str__() (cpe.cpe.CPE method), 10
 __str__() (cpe.cpe2_3.CPE2_3 method), 14
 __str__() (cpe.cpelang.CPELanguage method), 25
 __str__() (cpe.cpeset.CPESet method), 19

`__weakref__` (cpe.comp.cpecomp.CPEComponent attribute), 28
`__weakref__` (cpe.cpe.CPE attribute), 10
`__weakref__` (cpe.cpelang.CPELanguage attribute), 25
`__weakref__` (cpe.cpeset.CPESet attribute), 19

A

`append()` (cpe.cpeset.CPESet method), 19
`append()` (cpe.cpeset1_1.CPESet1_1 method), 20
`append()` (cpe.cpeset2_2.CPESet2_2 method), 21
`append()` (cpe.cpeset2_3.CPESet2_3 method), 22
`as_dict()` (cpe.cpe.CPE method), 10
`as_fs()` (cpe.comp.cpecomp1_1.CPEComponent1_1 method), 32
`as_fs()` (cpe.comp.cpecomp_simple.CPEComponentSimple method), 31
`as_fs()` (cpe.cpe.CPE method), 10
`as_uri_2_3()` (cpe.comp.cpecomp1_1.CPEComponent1_1 method), 33
`as_uri_2_3()` (cpe.comp.cpecomp_simple.CPEComponentSimple method), 31
`as_uri_2_3()` (cpe.cpe.CPE method), 11
`as_wfn()` (cpe.comp.cpecomp1_1.CPEComponent1_1 method), 33
`as_wfn()` (cpe.comp.cpecomp_simple.CPEComponentSimple method), 31
`as_wfn()` (cpe.cpe.CPE method), 11
`as_wfn()` (cpe.cpe1_1.CPE1_1 method), 13
`as_wfn()` (cpe.cpe2_2.CPE2_2 method), 14
`as_wfn()` (cpe.cpe2_3_uri.CPE2_3_URI method), 15

C

`compare_wfn()` (cpe.cpeset2_3.CPESet2_3 class method), 22
CPE (class in cpe.cpe), 9
CPE1_1 (class in cpe.cpe1_1), 12
CPE2_2 (class in cpe.cpe2_2), 13
CPE2_3 (class in cpe.cpe2_3), 14
CPE2_3_FS (class in cpe.cpe2_3_fs), 17
CPE2_3_URI (class in cpe.cpe2_3_uri), 15
CPE2_3_WFN (class in cpe.cpe2_3_wfn), 16
`cpe_disjoint()` (cpe.cpeset2_3.CPESet2_3 class method), 22
`cpe_equal()` (cpe.cpeset2_3.CPESet2_3 class method), 22
`cpe_subset()` (cpe.cpeset2_3.CPESet2_3 class method), 22
`cpe_superset()` (cpe.cpeset2_3.CPESet2_3 class method), 22
CPEComponent (class in cpe.comp.cpecomp), 27
CPEComponent1_1 (class in cpe.comp.cpecomp1_1), 32
CPEComponent2_2 (class in cpe.comp.cpecomp2_2), 34
CPEComponent2_3 (class in cpe.comp.cpecomp2_3), 34
CPEComponent2_3_FS (class in cpe.comp.cpecomp2_3_fs), 35

CPEComponent2_3_URI (class in cpe.comp.cpecomp2_3_uri), 34
CPEComponent2_3_URI_edpacked (class in cpe.comp.cpecomp2_3_uri_edpacked), 34
CPEComponent2_3_WFN (class in cpe.comp.cpecomp2_3_wfn), 35
CPEComponentAnyValue (class in cpe.comp.cpecomp_anyvalue), 29
CPEComponentEmpty (class in cpe.comp.cpecomp_empty), 29
CPEComponentLogical (class in cpe.comp.cpecomp_logical), 28
CPEComponentNotApplicable (class in cpe.comp.cpecomp_notapplicable), 30
CPEComponentSimple (class in cpe.comp.cpecomp_simple), 31
CPEComponentUndefined (class in cpe.comp.cpecomp_undefined), 30
CPELanguage (class in cpe.cpelang), 24
CPELanguage2_2 (class in cpe.cpelang2_2), 25
CPELanguage2_3 (class in cpe.cpelang2_3), 26
CPESet (class in cpe.cpeset), 19
CPESet1_1 (class in cpe.cpeset1_1), 20
CPESet2_2 (class in cpe.cpeset2_2), 21
CPESet2_3 (class in cpe.cpeset2_3), 21

G

`get_attribute_values()` (cpe.cpe1_1.CPE1_1 method), 13
`get_attribute_values()` (cpe.cpe2_2.CPE2_2 method), 14
`get_attribute_values()` (cpe.cpe2_3_fs.CPE2_3_FS method), 17
`get_attribute_values()` (cpe.cpe2_3_uri.CPE2_3_URI method), 15
`get_attribute_values()` (cpe.cpe2_3_wfn.CPE2_3_WFN method), 16
`get_edition()` (cpe.cpe.CPE method), 11
`get_language()` (cpe.cpe.CPE method), 11
`get_other()` (cpe.cpe.CPE method), 11
`get_part()` (cpe.cpe.CPE method), 11
`get_product()` (cpe.cpe.CPE method), 11
`get_software_edition()` (cpe.cpe.CPE method), 11
`get_target_hardware()` (cpe.cpe.CPE method), 11
`get_target_software()` (cpe.cpe.CPE method), 12
`get_update()` (cpe.cpe.CPE method), 12
`get_value()` (cpe.comp.cpecomp2_3_wfn.CPEComponent2_3_WFN method), 35
`get_value()` (cpe.comp.cpecomp_simple.CPEComponentSimple method), 31
`get_vendor()` (cpe.cpe.CPE method), 12
`get_version()` (cpe.cpe.CPE method), 12

I

`is_application()` (cpe.cpe.CPE method), 12
`is_hardware()` (cpe.cpe.CPE method), 12
`is_operating_system()` (cpe.cpe.CPE method), 12

`is_valid_attribute()` (`cpe.comp.cpecomp.CPEComponent`
class method), [28](#)

L

`language_match()` (`cpe.cpelang.CPELanguage`
method), [25](#)

`language_match()` (`cpe.cpelang2_2.CPELanguage2_2`
method), [25](#)

`language_match()` (`cpe.cpelang2_3.CPELanguage2_3`
method), [26](#)

N

`name_match()` (`cpe.cpeset.CPESet` method), [19](#)

`name_match()` (`cpe.cpeset1_1.CPESet1_1` method), [20](#)

`name_match()` (`cpe.cpeset2_2.CPESet2_2` method), [21](#)

`name_match()` (`cpe.cpeset2_3.CPESet2_3` method), [23](#)

S

`set_value()` (`cpe.comp.cpecomp1_1.CPEComponent1_1`
method), [33](#)

`set_value()` (`cpe.comp.cpecomp2_3_uri_edpacked.CPEComponent2_3_URI_edpacked`
method), [34](#)

`set_value()` (`cpe.comp.cpecomp2_3_wfn.CPEComponent2_3_WFN`
method), [35](#)

`set_value()` (`cpe.comp.cpecomp_simple.CPEComponentSimple`
method), [32](#)