

---

# language\_tags Documentation

*Release 0.1.0*

**Onroerend Erfgoed**

August 11, 2016



<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>API Documentation</b>	<b>5</b>
2.1	Class tags . . . . .	5
2.2	Class Tag . . . . .	6
2.3	Class Subtag . . . . .	8
<b>3</b>	<b>History</b>	<b>11</b>
3.1	0.4.2 . . . . .	11
3.2	0.4.1 . . . . .	11
3.3	0.4.0 . . . . .	11
<b>4</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>



This Python API offers a way to validate and lookup languages tags.

**Standard**

It is based on [BCP 47 \(RFC 5646\)](#) and the latest [IANA language subtag registry](#).

This project will be updated as the standards change.

**JSON data**

See the [language-subtag-registry](#) project for the underlying JSON data.

**Javascript version**

This project is a Python version of the [language-tags](#) Javascript project.



---

## Introduction

---

This Python API offers a way to validate and lookup languages tags.

Import the module:

```
from language_tags import tags
```

To check whether the language\_tag is valid use `tags.check()`. For example 'nl-Be' is valid but 'nl-BE-BE' is invalid.

```
> print(tags.check('nl-Be'))
True
> print(tags.check('nl-BE-BE'))
False
```

For meaningful error output see `tags.tag().errors`:

```
> errors = tags.tag('nl-BE-BE').errors
> for err in errors:
>     print(err.message)
Extra region subtag 'BE' found.
```

Lookup descriptions of tags:

```
> print(tags.description('nl-Be'));
['Dutch', 'Flemish', 'Belgium']
```

Lookup descriptions of a language subtag:

```
> print(tags.language('nl').description);
['Dutch', 'Flemish']
```

Lookup tags by description:

```
> language_subtags = tags.search('Flemish')
> print(language_subtags[0])
'nl'
```

Get the language subtag of a tag:

```
> print(repr(tags.tag('nl-Be').language))
{'subtag': 'nl', 'record': {'Subtag': 'nl', 'Suppress-Script': 'Latn', 'Added': '2005-10-16', 'Type': 'Language'}}
```

A redundant tag is a grandfathered registration whose individual subtags appear with the same semantic meaning in the registry<sup>1</sup>. A redundant tag has descriptions and can have a preferred tag.

---

<sup>1</sup> RFC 5646

```
> redundant_tag = tags.tag('es-419')
> print(redundant_tag.descriptions)
['Latin American Spanish']
> print(redundant_tag.valid)
True
> print(redundant_tag.region.description)
['Latin America and the Caribbean']
> print(redundant_tag.region.language)
['Spanish', 'Castilian']
```

The remainder of the previously registered tags are “grandfathered”<sup>1</sup>. Grandfathered tags cannot be parsed into subtags. A grandfathered tag has descriptions. Most grandfathered tags have valid preferred tags.

```
> grandfathered_tag = tags.tag('i-klingingon')
> print(grandfathered_tag.descriptions)
['Klingon']
> print(grandfathered_tag.valid)
False
> print(grandfathered_tag.subtags)
[]
> print(grandfathered_tag.preferred)
tlh
> preferred_tag = grandfathered_tag.preferred
> print(preferred_tag.language.description)
['Klingon', 'tlhIngan-Hol']
```

For the complete api documentation see next chapter.



## 2.1 Class tags

**class** `language_tags.tags.tags`

**static check** (*tag*)

Check if a string (hyphen-separated) tag is valid.

**Parameters** **tag** (*str*) – (hyphen-separated) tag.

**Returns** `bool` – True if valid.

**static date** ()

Get the file date of the underlying data as a string.

**Returns** date as string (for example: '2014-03-27').

**static description** (*tag*)

Gets a list of descriptions given the tag.

**Parameters** **tag** (*str*) – (hyphen-separated) tag.

**Returns** list of string descriptions. The return list can be empty.

**static filter** (*subtags*)

Get a list of non-existing string subtag(s) given the input string subtag(s).

**Parameters** **subtags** – string subtag or a list of string subtags.

**Returns** list of non-existing string subtags. The return list can be empty.

**static language** (*subtag*)

Get a language `language_tags.Subtag.Subtag` of the subtag string.

**Parameters** **subtag** (*str*) – subtag.

**Returns** language `language_tags.Subtag.Subtag` if exists, otherwise `None`.

**static languages** (*macrolanguage*)

Get a list of `language_tags.Subtag.Subtag` objects given the string macrolanguage.

**Parameters** **macrolanguage** (*string*) – subtag macrolanguage.

**Returns** a list of the macrolanguage `language_tags.Subtag.Subtag` objects.

**Raises** **Exception** – if the macrolanguage does not exist.

**static region** (*subtag*)

Get a region `language_tags.Subtag.Subtag` of the subtag string.

**Parameters** `subtag` (*str*) – subtag.

**Returns** region `language_tags.Subtag.Subtag` if exists, otherwise None.

**static search** (*description*, *all=False*)

Gets a list of `language_tags.Subtag.Subtag` objects where the description matches.

**Parameters**

- **description** (*str* or *RegExp*) – a string or compiled regular expression. For example: `search(re.compile('\d{4}'))` if the description of the returned subtag must contain four contiguous numerical digits.
- **all** (*bool*, *optional*) – If set on True grandfathered and redundant tags will be included in the return list.

**Returns** list of `language_tags.Subtag.Subtag` objects each including the description. The return list can be empty.

**static subtags** (*subtags*)

Get a list of existing `language_tags.Subtag.Subtag` objects given the input subtag(s).

**Parameters** `subtags` – string subtag or list of string subtags.

**Returns** a list of existing `language_tags.Subtag.Subtag` objects. The return list can be empty.

**static tag** (*tag*)

Get a `language_tags.Tag.Tag` of a string (hyphen-separated) tag.

**Parameters** `tag` (*str*) – (hyphen-separated) tag.

**Returns** `language_tags.Tag.Tag`.

**static type** (*subtag*, *type*)

Get a `language_tags.Subtag.Subtag` by subtag and type. Can be None if not exists.

**Parameters**

- **subtag** (*str*) – subtag.
- **type** (*str*) – type of the subtag.

**Returns** `language_tags.Subtag.Subtag` if exists, otherwise None.

**static types** (*subtag*)

Get the types of a subtag string (excludes redundant and grandfathered).

**Parameters** `subtag` (*str*) – subtag.

**Returns** list of types. The return list can be empty.

## 2.2 Class Tag

**class** `language_tags.Tag.Tag` (*tag*)

Tags for Identifying Languages based on BCP 47 (RFC 5646) and the latest IANA language subtag registry.

**Parameters** `tag` (*str*) – (hyphen-separated) tag.

**added**

Get the date string of grandfathered or redundant tag when it was added to the registry.

**Returns** added date string if the deprecated or redundant tag has one, otherwise None.

**deprecated**

Get the deprecation date of grandfathered or redundant tag if the tag is deprecated.

**Returns** deprecation date string if the deprecated or redundant tag has one, otherwise None.

**descriptions**

Get the list of descriptions of the grandfathered or redundant tag.

**Returns** list of descriptions. If no descriptions available, it returns an empty list.

**error** (*code*, *subtag=None*)

Get the `language_tags.Tag.Tag.Error` of a specific Tag error code. The error creates a message explaining the error. It also refers to the respective (sub)tag(s).

**Parameters**

- **code** (*int*) – a Tag error error:
  - 1 = Tag.ERR\_DEPRECATED
  - 2 = Tag.ERR\_NO\_LANGUAGE
  - 3 = Tag.ERR\_UNKNOWN,
  - 4 = Tag.ERR\_TOO\_LONG
  - 5 = Tag.ERR\_EXTRA\_REGION
  - 6 = Tag.ERR\_EXTRA\_EXTLANG
  - 7 = Tag.ERR\_EXTRA\_SCRIPT,
  - 8 = Tag.ERR\_DUPLICATE\_VARIANT
  - 9 = Tag.ERR\_WRONG\_ORDER
  - 10 = Tag.ERR\_SUPPRESS\_SCRIPT,
  - 11 = Tag.ERR\_SUBTAG\_DEPRECATED
  - 12 = Tag.ERR\_EXTRA\_LANGUAGE
- **subtag** – string (sub)tag or list of string (sub)tags creating the error.

**Returns** An exception class containing: a Tag error input code, the derived message with the given (sub)tag(s). input

**errors**

Get the errors of the tag. If invalid then the list will consist of errors containing each a code and message explaining the error. Each error also refers to the respective (sub)tag(s).

**Returns** list of errors of the tag. If the tag is valid, it returns an empty list.

**format**

Get format according to algorithm defined in RFC 5646 section 2.1.1.

**Returns** formatted tag string.

**language**

Get the language `language_tags.Subtag.Subtag` of the tag.

**Returns** language `language_tags.Subtag.Subtag` that is part of the tag. The return can be None.

**preferred**

Get the preferred `language_tags.Tag.Tag` of the deprecated or redundant tag.

**Returns** preferred `language_tags.Tag.Tag` if the deprecated or redundant tag has one, otherwise `None`.

**region**

Get the region `language_tags.Subtag.Subtag` of the tag.

**Returns** region `language_tags.Subtag.Subtag` that is part of the tag. The return can be `None`.

**script**

Get the script `language_tags.Subtag.Subtag` of the tag.

**Returns** script `language_tags.Subtag.Subtag` that is part of the tag. The return can be `None`.

**subtags**

Get the `language_tags.Subtag.Subtag` objects of the tag.

**Returns** list of `language_tags.Subtag.Subtag` objects that are part of the tag. The return list can be empty.

**type**

Get the type of the tag (either grandfathered, redundant or tag see RFC 5646 section 2.2.8.).

**Returns** string – type of the tag.

**valid**

Checks whether the tag is valid.

**Returns** Bool – True if valid otherwise False.

## 2.3 Class Subtag

**class** `language_tags.Subtag.Subtag` (*subtag*, *type*)

A subtag is a part of the hyphen-separated `language_tags.Tag.Tag`.

**Parameters**

- **subtag** (*str*) – subtag.
- **type** (*str*) – can be ‘language’, ‘extlang’, ‘script’, ‘region’ or ‘variant’.

**Returns**

**raise Error** Checks for `Subtag.ERR_NONEXISTENT` and `Subtag.ERR_TAG`.

**added**

Get the date when the subtag was added to the registry.

**Returns** date (as string) when the subtag was added to the registry.

**comments**

Get the comments of the subtag.

**Returns** list of comments. The return list can be empty.

**deprecated**

Get the deprecation date.

**Returns** deprecation date as string if subtag is deprecated, otherwise `None`.

**description**

Get the subtag description.

**Returns** list of description strings.

**format**

Get the subtag code conventional format according to RFC 5646 section 2.1.1.

**Returns** string – subtag code conventional format.

**preferred**

Get the preferred subtag.

**Returns** preferred *language\_tags.Subtag.Subtag* if exists, otherwise None.

**scope**

Get the subtag scope.

**Returns** string subtag scope if exists, otherwise None.

**script**

Get the language's default script of the subtag (RFC 5646 section 3.1.9)

**Returns** string – the language's default script.

**type**

Get the subtag type.

**Returns** string – either 'language', 'extlang', 'script', 'region' or 'variant'.



### 3.1 0.4.2

- Official python 3.5 compatibility
- Upgrade to <https://github.com/mattcg/language-subtag-registry/releases/tag/v0.3.15>

### 3.2 0.4.1

- Included the data folder again in the project package.
- Added bash script (*update\_data\_files.sh*) to download the `language-subtag-registry` and move this data in the data folder of the project.

### 3.3 0.4.0

- Allow parsing a redundant tag into subtags.
- Added package.json file for easy update of the language subtag registry data using `npm` (`npm install` or `npm update`)
- Improvement of the `language-tags.tags.search` function: rank equal description at top. See [mattcg/language-tags#4](https://github.com/mattcg/language-tags#4)

#### 3.3.1 0.3.2

- Upgrade to <https://github.com/mattcg/language-subtag-registry/releases/tag/v0.3.11>
- Added wheel config
- Fixed bug under windows: opening data files using utf-8 encoding.

#### 3.3.2 0.3.1

- Upgrade to <https://github.com/mattcg/language-subtag-registry/releases/tag/v0.3.8>

### 3.3.3 0.3.0

- Upgrade to <https://github.com/mattcg/language-subtag-registry/releases/tag/v0.3.6>
- Simplify output of `__str__` functions. The previous json dump is assigned to the repr function.

```
nlbe = tags.tags('nl-Latn-BE')
> print(nlbe)
'nl-Latn-BE'
> print(nlbe.language)
'nl'
> print(nlbe.script)
'Latn'
```

### 3.3.4 0.2.0

- Adjust language, region and script properties of Tag. The properties will return `language_tags.Subtag.Subtag` instead of a list of string subtags

```
> print(tags.tag('nl-BE').language)
'{"subtag": "nl", "record": {"Subtag": "nl", "Suppress-Script": "Latn", "Added": "2005-10-16", "Type": "language", "Description": "Dutch", "Script": "Latin"}, "Type": "language", "Description": "Dutch", "Script": "Latin"}'
> print(tags.tag('nl-BE').region)
'{"subtag": "be", "record": {"Subtag": "BE", "Added": "2005-10-16", "Type": "region", "Description": "Belgium", "Script": "Latin"}, "Type": "region", "Description": "Belgium", "Script": "Latin"}'
> print(tags.tag('en-mt-arab').script)
'{"subtag": "arab", "record": {"Subtag": "Arab", "Added": "2005-10-16", "Type": "script", "Description": "Arabic", "Script": "Arabic"}, "Type": "script", "Description": "Arabic", "Script": "Arabic"}'
```

### 3.3.5 0.1.1

- Added string and Unicode functions to make it easy to print Tags and Subtags.

```
> print(tags.tag('nl-BE'))
'{"tag": "nl-be"}'
```

- Added functions to easily select either the language, region or script subtags strings of a Tag.

```
> print(tags.tag('nl-BE').language)
['nl']
```

### 3.3.6 0.1.0

- Initial version



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



|

language\_tags.Subtag, 8

language\_tags.Tag, 6

language\_tags.tags, 5



**A**

added (language\_tags.Subtag.Subtag attribute), 8  
added (language\_tags.Tag.Tag attribute), 6

**C**

check() (language\_tags.tags.tags static method), 5  
comments (language\_tags.Subtag.Subtag attribute), 8

**D**

date() (language\_tags.tags.tags static method), 5  
deprecated (language\_tags.Subtag.Subtag attribute), 8  
deprecated (language\_tags.Tag.Tag attribute), 7  
description (language\_tags.Subtag.Subtag attribute), 8  
description() (language\_tags.tags.tags static method), 5  
descriptions (language\_tags.Tag.Tag attribute), 7

**E**

error() (language\_tags.Tag.Tag method), 7  
errors (language\_tags.Tag.Tag attribute), 7

**F**

filter() (language\_tags.tags.tags static method), 5  
format (language\_tags.Subtag.Subtag attribute), 9  
format (language\_tags.Tag.Tag attribute), 7

**L**

language (language\_tags.Tag.Tag attribute), 7  
language() (language\_tags.tags.tags static method), 5  
language\_tags.Subtag (module), 8  
language\_tags.Tag (module), 6  
language\_tags.tags (module), 5  
languages() (language\_tags.tags.tags static method), 5

**P**

preferred (language\_tags.Subtag.Subtag attribute), 9  
preferred (language\_tags.Tag.Tag attribute), 7

**R**

region (language\_tags.Tag.Tag attribute), 8

region() (language\_tags.tags.tags static method), 5

**S**

scope (language\_tags.Subtag.Subtag attribute), 9  
script (language\_tags.Subtag.Subtag attribute), 9  
script (language\_tags.Tag.Tag attribute), 8  
search() (language\_tags.tags.tags static method), 6  
Subtag (class in language\_tags.Subtag), 8  
subtags (language\_tags.Tag.Tag attribute), 8  
subtags() (language\_tags.tags.tags static method), 6

**T**

Tag (class in language\_tags.Tag), 6  
tag() (language\_tags.tags.tags static method), 6  
tags (class in language\_tags.tags), 5  
type (language\_tags.Subtag.Subtag attribute), 9  
type (language\_tags.Tag.Tag attribute), 8  
type() (language\_tags.tags.tags static method), 6  
types() (language\_tags.tags.tags static method), 6

**V**

valid (language\_tags.Tag.Tag attribute), 8