

pytttsx3

Navigation

Supported synthesizers

Using pytttsx3

- The Engine factory
- The Engine interface
- The Voice metadata
- Examples

Implementing drivers

Quick search

Go

Using pytttsx3

An application invokes the `pytttsx3.init()` factory function to get a reference to a `pytttsx3.Engine` instance. During construction, the engine initializes a `pytttsx3.driver.DriverProxy` object responsible for loading a speech engine driver implementation from the `pytttsx3.drivers` module. After construction, an application uses the engine object to register and unregister event callbacks; produce and stop speech; get and set speech engine properties; and start and stop event loops.

The Engine factory

`pytttsx3.init([driverName : string, debug : bool]) → pytttsx3.Engine`

Gets a reference to an engine instance that will use the given driver. If the requested driver is already in use by another engine instance, that engine is returned. Otherwise, a new engine is created.

- Parameters:**
- driverName** – Name of the `pytttsx3.drivers` module to load and use. Defaults to the best available driver for the platform, currently:
 - `sapi5` - SAPI5 on Windows
 - `nss` - NSSpeechSynthesizer on Mac OS X
 - `espeak` - eSpeak on every other platform
 - debug** – Enable debug output or not.
- Raises:**
- ImportError** – When the requested driver is not found
 - RuntimeError** – When the driver fails to initialize

The Engine interface

`class pytttsx3.engine.Engine`

Provides application access to text-to-speech synthesis.

connect(topic : string, cb : callable) → dict
Registers a callback for notifications on the given topic.

Parameters:

- topic** – Name of the event to subscribe to.
- cb** – Function to invoke when the event fires.

Returns: A token that the caller can use to unsubscribe the callback later.

The following are the valid topics and their callback signatures.

started-utterance
Fired when the engine begins speaking an utterance. The associated callback must have the following signature.

onStartUtterance(name : string) → None
Parameters: **name** – Name associated with the utterance.

started-word
Fired when the engine begins speaking a word. The associated callback must have the following signature.

onStartWord(name : string, location : integer, length : integer)
Parameters: **name** – Name associated with the utterance.

finished-utterance
Fired when the engine finishes speaking an utterance. The associated callback must have the following signature.

onFinishUtterance(name : string, completed : bool) → None
Parameters:

- name** – Name associated with the utterance.
- completed** – True if the utterance was output in its entirety or not.

error
Fired when the engine encounters an error. The associated callback must have the following signature.

onError(name : string, exception : Exception) → None
Parameters:

- name** – Name associated with the utterance that caused the error.
- exception** – Exception that was raised.

disconnect(token : dict)
Unregisters a notification callback.

Parameters: **token** – Token returned by **connect()** associated with the callback to be disconnected.

endLoop() → None
Ends a running event loop. If **startLoop()** was called with *useDriverLoop* set to True, this method stops processing of engine commands and immediately exits the event loop. If it was called with False, this method stops processing of engine commands, but it is up to the caller to end the external event loop it started.

Raises: **RuntimeError** – When the loop is not running

getProperty(name : string) → object
Gets the current value of an engine property.

Parameters: **name** – Name of the property to query.
Returns: Value of the property at the time of this invocation.

The following property names are valid for all drivers.

rate
Integer speech rate in words per minute. Defaults to 200 word per minute.

voice
String identifier of the active voice.

voices
List of `pytttsx3.voice.Voice` descriptor objects.

volume
Floating point volume in the range of 0.0 to 1.0 inclusive. Defaults to 1.0.

isBusy() → bool
Gets if the engine is currently busy speaking an utterance or not.

Returns: True if speaking, false if not.

runAndWait() → None
Blocks while processing all currently queued commands. Invokes callbacks for engine notifications appropriately. Returns when all commands queued before this call are emptied from the queue.

say(text : unicode, name : string) → None
Queues a command to speak an utterance. The speech is output according to the properties set before this command in the queue.

Parameters:

- text** – Text to speak.
- name** – Name to associate with the utterance. Included in notifications about this utterance.

setProperty(name, value) → None
Queues a command to set an engine property. The new property value affects all utterances queued after this command.

Parameters:

- name** – Name of the property to change.
- value** – Value to set.

The following property names are valid for all drivers.

rate
Integer speech rate in words per minute.

voice
String identifier of the active voice.

volume
Floating point volume in the range of 0.0 to 1.0 inclusive.

startLoop([useDriverLoop : bool]) → None
Starts running an event loop during which queued commands are processed and notifications are fired.

Parameters: **useDriverLoop** – True to use the loop provided by the selected driver. False to indicate the caller will enter its own loop after invoking this method. The caller's loop must pump events for the driver in use so that pytttsx3 notifications are delivered properly (e.g., SAPI5 requires a COM message pump). Defaults to True.

stop() → None
Stops the current utterance and clears the command queue.

The Voice metadata

`class pytttsx3.voice.Voice`

Contains information about a speech synthesizer voice.

age
Integer age of the voice in years. Defaults to **None** if unknown.

gender
String gender of the voice: *male*, *female*, or *neutral*. Defaults to **None** if unknown.

id
String identifier of the voice. Used to set the active voice via `pytttsx3.engine.Engine.setPropertyValue()`. This attribute is always defined.

languages
List of string languages supported by this voice. Defaults to an empty list of unknown.

name
Human readable name of the voice. Defaults to **None** if unknown.

Examples

Speaking text

```
import pytttsx3
engine = pytttsx3.init()
engine.say('Sally sells seashells by the seashore.')
engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Saving voice to a file

```
import pytttsx3
engine = pytttsx3.init()
engine.save_to_file('Hello World' , 'test.mp3')
engine.runAndWait()
```

Listening for events

```
import pytttsx3
def onStart(name):
    print 'starting', name
def onWord(name, location, length):
    print 'word', name, location, length
def onEnd(name, completed):
    print 'finishing', name, completed
engine = pytttsx3.init()
engine.connect('started-utterance', onStart)
engine.connect('started-word', onWord)
engine.connect('finished-utterance', onEnd)
engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Interrupting an utterance

```
import pytttsx3
def onWord(name, location, length):
    print 'word', name, location, length
    if location > 10:
        engine.stop()
engine = pytttsx3.init()
engine.connect('started-word', onWord)
engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Changing voices

```
engine = pytttsx3.init()
voices = engine.getProperty('voices')
for voice in voices:
    engine.setProperty('voice', voice.id)
    engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Changing speech rate

```
engine = pytttsx3.init()
rate = engine.getProperty('rate')
engine.setProperty('rate', rate+50)
engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Changing volume

```
engine = pytttsx3.init()
volume = engine.getProperty('volume')
engine.setProperty('volume', volume-0.25)
engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Running a driver event loop

```
engine = pytttsx3.init()
def onStart(name):
    print 'starting', name
def onWord(name, location, length):
    print 'word', name, location, length
def onEnd(name, completed):
    print 'finishing', name, completed
    if name == 'fox':
        engine.say('What a lazy dog!', 'dog')
    elif name == 'dog':
        engine.endLoop()
engine = pytttsx3.init()
engine.connect('started-utterance', onStart)
engine.connect('started-word', onWord)
engine.connect('finished-utterance', onEnd)
engine.say('The quick brown fox jumped over the lazy dog.', 'fox')
engine.startLoop()
```

Using an external event loop

```
engine = pytttsx3.init()
engine.say('The quick brown fox jumped over the lazy dog.', 'fox')
engine.startLoop(False)
# engine.iterate() must be called inside externalLoop()
externalLoop()
engine.endLoop()
```