clova-cek-sdk Documentation

Release 1.0.0

Vogel Frederik, Ryuichi Yamamoto

Aug 03, 2018

References

This is a python library to simplify the use of the Clova Extensions Kit (CEK). If you want to create your own service, you first need to create your own Extension. https://clova-developers.line.me/

CHAPTER 1

Quick start

1. Create a *cek.clova.Clova* instance.

```
import os
from cek import Clova

# application_id is used to verify requests.
application_id = os.environ.get("APPLICATION_ID")
# Set debug_mode=True if you are testing your extension. If True, this disables␣
↪request verification
clova = Clova(application_id=application_id, default_language="ja", debug_mode=False)
```

2. Define request handlers for CEK (on LaunchRequest, IntentRequest, etc).

```
@clova.handle.launch
def launch_request_handler(clova_request):
    return clova.response("")

@clova.handle.default
def default_handler(clova_request):
    return clova.response("")
```

3. Setup a web API endpoint. Use *cek.clova.Clova.route()* to route requests.

```
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/app', methods=['POST'])
def my_service():
    resp = clova.route(request.data, request.headers)
    resp = jsonify(resp)
    # make sure we have correct Content-Type that CEK expects
    resp.headers['Content-Type'] = 'application/json;charset-UTF-8'
    return resp
```

4. Save as `app.py` and run app

```
FLASK_APP=app.py flask run
```

For a detailed example, see clova-cek-sdk-python-sample example extension. See also the API documentation to know how to use the SDK.

1.1 Core

Provides types and helpers to create and work with CEK responses easily. For a high-level interface upon the *cek.core* module, see documentation for *cek.clova* module.

1.1.1 Request

class cek.core.**Request**(*request_dict*)

Type represents a request from CEK

Parameters **request_dict** (*dict*) – Dictioanry represents a request from CEK

Variables

- **request_type** (*str*) – can be LaunchRequest, IntentRequest, etc.
- **intent_name** (*str*) – name of intent if exist, otherwise None.
- **is_intent** (*str*) – IntentRequest or not.
- **user_id** (*str*) – user id.
- **application_id** (*str*) – application id.
- **access_token** (*str*) – access token.
- **session_id** (*str*) – session id.
- **session_attributes** (*dict*) – session attributes.
- **slots_dict** (*dict*) – slots as dictionary

slot_value(*slot_name*)

Returns slot value or None if missing.

Parameters **slot_name** (*str*) – slot name

Returns slot value if exists, None otherwise.

Return type str

Raises TypeError: if the request is not an IntentRequest

Usage:

```
>>> req.slot_value('Light')
''
```

verify_application_id(*application_id*)

Verify application id

Raises `RuntimeError` – if application id is incorrect.

1.1.2 Response

class cek.core.**Response**

Type represents a response from CEK

Variables

- **session_attributes** (*dict*) – Session attributes in a dictionary format
- **reprompt** (*dict*) – reprompt value, can be SimpleSpeech, SpeechSet or SpeechList.

1.1.3 SpeechBuilder

class cek.core.**SpeechBuilder**(*default_language='ja'*)

Helper class to build speech objects that can be part of CEK response.

Parameters **default_language** (*str*) – Set default language for all messages. Can be `ja`, `ko` or `en`.

Raises `ValueError` – if unsupported language is specified.

Usage: All the examples below assume the following helper is defined in advance.

```
>>> from cek import SpeechBuilder
>>> from pprint import pprint
>>> speech_builder = SpeechBuilder(default_language="ja")
```

Building a plain text object:

```
>>> speech_builder.plain_text("")
{'type': 'PlainText', 'lang': 'ja', 'value': ''}
```

plain_text(*message*, *language=None*)

Build a PlainText object

Parameters

- **message** (*str*) – String Message which clova should speak out
- **language** (*str*) – Language code of the message

Returns Dictionary with the format of a SpeechInfo with type PlainText

Return type dict

Raises `ValueError` – if unsupported language is specified.

Usage:

```
>>> speech_builder.plain_text("")
{'type': 'PlainText', 'lang': 'ja', 'value': ''}
```

simple_speech(*speech_value*)

Build a SimpleSpeech object

Parameters **speech_value** (*dict*) – speech_value can be plain_text or url SpeechInfo

Returns Dictionary in the format of a SimpleSpeech

Return type dict

Usage:

```
>>> text = builder.plain_text("")
>>> pprint(speech_builder.simple_speech(text))
{'type': 'SimpleSpeech',
 'values': {'lang': 'ja', 'type': 'PlainText', 'value': ''}}
```

speech_list (*speech_values*)
Build a SpeechList object

Parameters **speech_values** (*list*) – List which can consist of plain_text SpeechInfo or url SpeechInfo

Returns Dictionary in the format of a SpeechList

Return type dict

Usage:

```
>>> from pprint import pprint
>>> text = speech_builder.plain_text("")
>>> url = speech_builder.url("https://dummy.mp3")
>>> speech_list = speech_builder.speech_list([text, url])
>>> pprint(speech_list)
{'type': 'SpeechList',
 'values': [{'lang': 'ja', 'type': 'PlainText', 'value': ''},
            {'lang': '', 'type': 'URL', 'value': 'https://dummy.mp3'}]}
```

speech_set (*brief*, *verbose*)
Build a SpeechSet object

Parameters

- **brief** (*dict*) – A Dictionary of a plain_text SpeechInfo or url SpeechInfo
- **verbose** (*dict*) – A Dictionary of a SpeechList or SimpleSpeech

Returns Dictionary in the format of a SpeechSet

Return type dict

url (*url*)
Build an URL object

Parameters **url** (*str*) – URL of the audio file which should be played by clova

Returns Dictionary with the format of a SpeechInfo with type URL

Return type dict

Usage:

```
>>> speech_builder.url("https://dummy.mp3")
{'type': 'URL', 'lang': '', 'value': 'https://dummy.mp3'}
```

1.1.4 ResponseBuilder

class cek.core.**ResponseBuilder**(*default_language='ja'*)

Helper class to build responses for CEK

Parameters **default_language** (*str*) – Set default language for all messages. Can be ja, ko or en.

Raises **ValueError** – if unsupported language is specified.

Usage: All the examples below assume the following helper is defined in advance.

```
>>> from cek import SpeechBuilder, ResponseBuilder
>>> from pprint import pprint
>>> speech_builder = SpeechBuilder(default_language="ja")
>>> response_builder = ResponseBuilder(default_language="ja")
```

Building a SimpleSpeech response:

```
>>> resp = response_builder.simple_speech_text("")
>>> pprint(resp)
{'response': {'card': {},
              'directives': [],
              'outputSpeech': {'type': 'SimpleSpeech',
                               'values': {'lang': 'ja',
                                          'type': 'PlainText',
                                          'value': ''}},
              'shouldEndSession': False},
 'sessionAttributes': {},
 'version': '1.0'}
```

add_reprompt(*response*, *speech*)

Add a repromt to your response. It is recommended to use a SimpleSpeech to keep the reprompt short

Parameters

- **response** (*dict-like*) – Response Dictionary to which the reprompt should be added
- **speech** (*dict*) – Speech can be a Dictionary of Simple Speech, SpeechList or SpeechSet

Returns Response with added Speech reprompt

Return type dict-like

simple_speech(*speech_value*, *end_session=False*)

Build a SimpleSpeech response

Parameters

- **speech_value** (*dict*) – Value which can consist of plain_text SpeechInfo or url SpeechInfo
- **end_session** (*bool*) – Whether Clova should continue to listen or end the session

Returns Response that wraps a Dictionary in the format of a response for a SimpleSpeech

Return type *cek.core.Response*

Usage:

```
>>> text = speech_builder.plain_text("")
>>> resp = response_builder.simple_speech(text)
>>> pprint(resp)
{'response': {'card': {},
              'directives': [],
              'outputSpeech': {'type': 'SimpleSpeech',
                               'values': {'lang': 'ja',
                                          'type': 'PlainText',
                                          'value': ''}},
              'shouldEndSession': False},
 'sessionAttributes': {},
 'version': '1.0'}
```

simple_speech_text(*message*, *language=None*, *end_session=False*)

Build SimpleSpeech response with plain_text value

Parameters

- **message** (*str*) – String Message which clova should speak out
- **language** (*str*) – Language code of the message
- **end_session** (*bool*) – Whether Clova should continue to listen or end the session

Returns Response that wraps a Dictionary in the format of a response for a SimpleSpeech

Return type *cek.core.Response*

Raises **ValueError** – if unsupported language is specified.

Usage:

```
>>> resp = response_builder.simple_speech_text("")
>>> pprint(resp)
{'response': {'card': {},
              'directives': [],
              'outputSpeech': {'type': 'SimpleSpeech',
                               'values': {'lang': 'ja',
                                          'type': 'PlainText',
                                          'value': ''}},
              'shouldEndSession': False},
 'sessionAttributes': {},
 'version': '1.0'}
```

speech_list(*speech_values*, *end_session=False*)

Build a SpeechList response

Parameters

- **speech_values** (*list*) – List which can consist of plain_text SpeechInfo or url SpeechInfo
- **end_session** (*bool*) – Whether Clova should continue to listen or end the session

Returns Response that wraps a Dictionary in the format of a response for a SpeechList

Return type *cek.core.Response*

Usage:

```
>>> text = speech_builder.plain_text("")
>>> url = speech_builder.url("https://dummy.mp3")
>>> resp = response_builder.speech_list([text, url])
>>> pprint(resp)
{'response': {'card': {},
              'directives': [],
              'outputSpeech': {'type': 'SpeechList',
                               'values': [{'lang': 'ja',
                                           'type': 'PlainText',
                                           'value': ''},
                                          {'lang': '',
                                           'type': 'URL',
                                           'value': 'https://dummy.mp3'}]}
↪,
              'shouldEndSession': False},
 'sessionAttributes': {},
 'version': '1.0'}
```

speech_set(*brief*, *verbose*, *end_session=False*)
Build a SpeechSet response

Parameters

- **brief** (*dict*) – A Dictionary of a plain_text SpeechInfo or url SpeechInfo
- **verbose** (*dict*) – A Dictionary of a SpeechList or SimpleSpeech
- **end_session** (*bool*) – Whether Clova should continue to listen or end the session

Returns Response that wraps a Dictionary in the format of a response for a SpeechSet

Return type *cek.core.Response*

speech_url(*message*, *url*, *language=None*, *end_session=False*)
Build a SpeechList response with a message and an URL

Parameters

- **message** (*str*) – String Message which clova should speak out
- **url** (*str*) – String URL of the audio file which should be played by clova
- **language** (*str*) – Language code of the message
- **end_session** (*bool*) – Whether Clova should continue to listen or end the session

Returns Response that wraps a Dictionary in the format of a response for a SpeechList

Return type *cek.core.Response*

Raises **ValueError** – if unsupported language is specified.

Usage:

```
>>> resp = response_builder.speech_url("", "https://dummy.mp3")
>>> pprint(resp)
{'response': {'card': {},
              'directives': [],
              'outputSpeech': {'type': 'SpeechList',
                               'values': [{'lang': 'ja',
                                           'type': 'PlainText',
                                           'value': ''},
```

(continues on next page)

(continued from previous page)

```
                                                {'lang': '',
                                                 'type': 'URL',
                                                 'value': 'https://dummy.mp3'}]}
↪,
                'shouldEndSession': False},
 'sessionAttributes': {},
 'version': '1.0'}
```

1.1.5 RequestHandler

class cek.core.**RequestHandler**(*application_id*, *debug_mode=False*)

Helper class to handle requests from CEK.

Parameters

- **application_id** (*str*) – Application ID that was used to register this Extension f.e.
- **debug_mode** (*bool*) – When set to True, application_id and request verification are ignored.

Usage: All the examples below assume the following helpers are defined in advance.

```
>>> from cek import RequestHandler, ResponseBuilder
>>> clova_handler = RequestHandler(application_id="", debug_mode=True)
>>> builder = ResponseBuilder(default_language="ja", debug_mode=False)
```

Defining handlers can be done by decorators:

```
>>> @clova_handler.launch
... def launch_request_handler(clova_request):
...     return builder.simple_speech_text("")
```

```
>>> @clova_handler.default
... def default_handler(clova_request):
...     return builder.simple_speech_text("")
```

If you have defined request handlers, then setup a web API endpoint as follows:

```
>>> rom flask import Flask, request, jsonify
>>> app = Flask(__name__)
>>> @app.route('/app', methods=['POST'])
... def my_service():
...     resp = clova_handler.route_request(request.data, request.headers)
...     resp = jsonify(resp)
...     resp.headers['Content-Type'] = 'application/json;charset-UTF-8'
...     return resp
```

default(*func*)

Default handler

Parameters **func** – Function

Usage:

```
>>> @clova_handler.default
... def default_handler(clova_request):
...     return builder.simple_speech_text("")
```

end (*func*)
End handler called on SessionEndedRequest.

Parameters **func** – Function

Usage:

```
>>> @clova_handler.end
... def end_handler(clova_request):
...     # Session ended, this handler can be used to clean up
...     return
```

intent (*intent*)
Intent handler called on IntentRequest.

Parameters **intent** (*str*) – intent name

Usage:

```
>>> @clova_handler.intent("Clova.YesIntent")
... def intent_handler(clova_request):
...     return builder.simple_speech_text("")
```

launch (*func*)
Launch handler called on LaunchRequest.

Parameters **func** – Function

Usage:

```
>>> @clova_handler.launch
... def launch_request_handler(clova_request):
...     return builder.simple_speech_text("")
```

route_request (*request_body*, *request_header_dict*)
Route request from CEK.

Parameters

- **request_body** (*bytes*) – Binary Request body from CEK
- **request_header_dict** (*dict*) – Request Header as dictionary

Returns Returns body for CEK response

Return type *cek.core.Response*

Raises

- **cryptography.exceptions.InvalidSignature** – (non-debug mode only) if request verification failed.
- **RuntimeError** – (non-debug mode only) if application id is incorrect.

Usage:

```
>>> from flask import Flask, request, Response
>>> app = Flask(__name__)
>>> @app.route('/app', methods=['POST'])
... def my_service():
...     resp = clova_handler.route_request(request.data, request.headers)
...     resp = jsonify(resp)
...     resp.headers['Content-Type'] = 'application/json;charset-UTF-8'
...     return resp
```

1.2 Clova

An abstracted layer on top of the *cek.core* module.

1.2.1 URL

class cek.clova.**URL**(*url*)

Parameters **url** (*str*) – As str which points directly to a source which should be played by Clover

1.2.2 Message

class cek.clova.**Message**(*message*, *language=None*)

Parameters

- **message** (*str*) – Message of type str which Clova should speak out
- **language** (*str*) – Language code of the message. Can be ja, ko or en.

1.2.3 MessageSet

class cek.clova.**MessageSet**(*brief*, *verbose*)

Parameters

- **brief** – Can be of type str, *Message* or *URL*
- **verbose** – Can be of type str, *Message* or *URL* or a list containing any of those three types

1.2.4 Clova

class cek.clova.**Clova**(*application_id*, *default_language='ja'*, *debug_mode=False*)

Clova provides the easiest way to create your extension.

Parameters

- **application_id** (*str*) – Set registered application id to verify all incoming requests
- **default_language** (*str*) – Set default language for all messages. Can be ja, ko or en.

- **debug_mode** (*bool*) – Use only for development. If set to True, request and applicationId verification are turned off

Raises **ValueError** – if unsupported language is specified.

Variables **handle** (RequestHandler) – Helper to handle requests from CEK. Request handlers must be defined using the handler.

Usage: Create *Clova* instance:

```
>>> from cek import Clova
>>> clova = Clova(application_id='', default_language='jp', debug_mode=True)
```

Define request handlers using (Clova.handle). Response can be created using *Clova.response()*.

```
>>> @clova.handle.launch
... def launch_request_handler(clova_request):
...     return clova.response("")
```

```
>>> @clova.handle.default
... def default_handler(clova_request):
...     return clova.response("")
```

Plug into your web application using *Clova.route()*:

```
>>> rom flask import Flask, request, jsonify
>>> app = Flask(__name__)
>>> @app.route('/app', methods=['POST'])
... def my_service():
...     resp = clova.route(request.data, request.headers)
...     resp = jsonify(resp)
...     resp.headers['Content-Type'] = 'application/json;charset-UTF-8'
...     return resp
```

See docs for *Clova.route()* and *Clova.response()* for detailed usages.

response(*message*, *reprompt=None*, *end_session=False*)

Create a Response that should be sent back to CEK

Parameters

- **message** – Can be of type str, *Message* or *URL* or a list, *MessageSet* containing any of these types
- **reprompt** – Can be of type str, *Message* or *URL* or a list, *MessageSet* containing any of these types
- **end_session** (*bool*) – Whether Clova should continue to listen or end the session

Returns Response containing passed message

Return type *cek.core.Response*

Raises **ValueError** – if unsupported language is specified.

Usage:

```
>>> import cek
>>> from cek import Clova
>>> from pprint import pprint
>>> clova = Clova(application_id="", default_language="ja", debug_
↪mode=True)
```

Simplest case:

```
>>> resp = clova.response("")
>>> pprint(resp)
{'response': {'card': {},
              'directives': [],
              'outputSpeech': {'type': 'SimpleSpeech',
                               'values': {'lang': 'ja',
                                          'type': 'PlainText',
                                          'value': ''}},
              'shouldEndSession': False},
 'sessionAttributes': {},
 'version': '1.0'}
```

With explicit language:

```
>>> resp = clova.response(cek.Message("English", language="en"))
>>> pprint(resp)
{'response': {'card': {},
              'directives': [],
              'outputSpeech': {'type': 'SimpleSpeech',
                               'values': {'lang': 'en',
                                          'type': 'PlainText',
                                          'value': 'English'}},
              'shouldEndSession': False},
 'sessionAttributes': {},
 'version': '1.0'}
```

URL:

```
>>> resp = clova.response(cek.URL("https://dummy.mp3"))
>>> pprint(resp)
{'response': {'card': {},
              'directives': [],
              'outputSpeech': {'type': 'SimpleSpeech',
                               'values': {'lang': '',
                                          'type': 'URL',
                                          'value': 'https://dummy.mp3'}},
              'shouldEndSession': False},
 'sessionAttributes': {},
 'version': '1.0'}
```

List:

```
>>> resp = clova.response(["", cek.URL("https://dummy.mp3")])
>>> pprint(resp)
{'response': {'card': {},
              'directives': [],
              'outputSpeech': {'type': 'SpeechList',
                               'values': [{'lang': 'ja',
                                           'type': 'PlainText',
```

(continues on next page)

(continued from previous page)

```
                                            'value': ''},
                                           {'lang': '',
                                            'type': 'URL',
                                            'value': 'https://dummy.mp3'}]}
↪,
              'shouldEndSession': False},
 'sessionAttributes': {},
 'version': '1.0'}
```

With reprompt message:

```
>>> resp = clova.response("", reprompt="")
>>> pprint(resp)
{'response': {'card': {},
              'directives': [],
              'outputSpeech': {'type': 'SimpleSpeech',
                               'values': {'lang': 'ja',
                                          'type': 'PlainText',
                                          'value': ''}},
              'reprompt': {'outputSpeech': {'type': 'SimpleSpeech',
                                            'values': {'lang': 'ja',
                                                       'type': 'PlainText
↪',
                                                       'value': ''}}},
              'shouldEndSession': False},
 'sessionAttributes': {},
 'version': '1.0'}
```

route(*body*, *header*)

Route request from CEK to handlers

Depending on the request type (intent, launch, etc), the function routes the request to the proper handler defined by the user and returns the response as a dictionary. Request verifiation is done per request before routing.

The method is an alias to *cek.core.RequestHandler.route_request()*.

Parameters

- **body** (*bytes*) – Binary Request body from CEK
- **header** (*dict*) – Request Header as a dictionary

Returns Returns body for CEK response

Return type *cek.core.Response*

Raises

- **cryptography.exceptions.InvalidSignature** – (non-debug mode only) if request verification failed.
- **RuntimeError** – (non-debug mode only) if application id is incorrect.

Usage:

```
>>> from cek import Clova
>>> clova = Clova(application_id="", default_language="ja", debug_
↪mode=True)
```

(continues on next page)

(continued from previous page)

```
>>> from flask import Flask, request, jsonify
>>> app = Flask(__name__)
>>> @app.route('/app', methods=['POST'])
... def my_service():
...     resp = clova.route(request.data, request.headers)
...     resp = jsonify(resp)
...     resp.headers['Content-Type'] = 'application/json;charset-UTF-8'
...     return resp
```

1.3 Change log

1.3.1 1.0.0 <2018-08-03>

- Initial release

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index