addonpayments Documentation

Release 0.1

Marc Galofré

Apr 15, 2021

Contents:

AddonPayments SDK Python is a library that allows integration with the SDK's of AddonPayments in a easy and fast way. There are two types of integration: HPP (Hosted Payment Page) and API.

CHAPTER 1

Features

1.1 HPP

- Integrates in minutes
- Minimises the merchant's PCI overheads
- Customisable, for a seamless checkout process

1.2 API

- Gives you full control of the payment process
- Provides access to a full suite of transaction management requests
- Suitable for call centre integrations

1.2.1 Installation

The installation process i very easy and quick. This process will install everything you need to use both types of integrations, the HPP and the API. Before you have an account on Addon Payments if you do not have it, you can do so via the Addon Payments registration form.

Please follow these steps to installation:

1. Install AddonPayments SDK and its dependencies via `pip install addonpayments-sdk-python`.
2. Configure the SDK with your Addon Payments account. Create the settings.ini file in your root project with the content that we provide in the addonpayments/settings.ini.template file:

```
[settings]
DEBUG=True
MERCHANT_ID = yourmerchantid
```

(continues on next page)

(continued from previous page)

```
SHARED_SECRET = yoursharedsecret
ADDONPAYMENTS_HPP_URL = https://hpp.sandbox.addonpayments.com/pay
ADDONPAYMENTS_API_URL = https://remote.sandbox.addonpayments.com/remote
```

1.2.2 Getting started

Here you will find the documentation for the SDK Addon Payemnts integration with Python. This library will allow you to use SDKs in any Python project (Django, Flask, . . .) very easily.

The more specific details of the internal operation of AddonPayments will be found in the Addon Payments developer hub.

Currently the library supports the following features:

Services	HPP	API
Process a Payment (Auth)	Yes	Yes
Card Storage	Yes	Yes
3D Secure	No	Yes
DCC	No	Yes
PayPal	No	Yes

1.2.3 HPP Reference

The HPP (Hosted Payment Page) generates JSON requests and receives the information from the selected transaction. Later, the JavaScript library provided by AddonPayments will manage the requests and answers. For more information about HPP, please visit the HPP reference from Addon Payments developer hub.

Then we will see the steps to follow to generate JSON requests for each type of transaction.

Note: There are mandatory fields that are auto-generated by the SDK generation utilities

- sha1hash: The SDK automatically generates a SHA-1 digital signature for each JSON requests.
- order_id: The transaction ID can be defined by user or auto-generated by the SDK.
- timestamp: The transaction timestamp can be defined by user or auto-generated by the SDK.

1.2.3.1 HPP

All JSON are generated with HPP class. HPP carries out the following actions:

- Generate security hash
- Validates inputs
- Base64 encodes inputs
- Serialises request object to JSON

```
from addonpayments.hpp.hpp import Hpp

# create request/response handler
```

(continues on next page)

(continued from previous page)

```
hpp = Hpp('my_secrect')
# parse request object to json object
json_request = hpp.request_to_json(req)
```

1.2.3.2 Process a Payment

For more detailed information, please see the section Process a Payment into Addon Payments developer hub.

1. Create the Payment request.
2. Auto-generation of the fields and transformation to JSON.

```
from addonpayments.hpp.payment.requests import PaymentRequest
from addonpayments.hpp.hpp import Hpp

# create request to send
request = PaymentRequest(
    merchant_id='my_merchant_id',
    amount=100,
    currency='EUR',
    auto_settle_flag=True
)
# create request/response handler
hpp = Hpp('my_secrect')
# parse request object to json object
json_request = hpp.request_to_json(req)
```

1.2.3.3 Card storage

Please see the Card storage section into Addon Payments developer hub.

Create a payer and store card

For more detailed information, please see the section Create a payer and store card (Card storage) into Addon Payments developer hub.

1. **Create the Card storage request:**
    - if payer reference (payer_ref) and payment reference (pmt_ref) are empty, these references are internally generates by AddonPayments.
2. Auto-generation of the fields and transformation to JSON.

```
from addonpayments.hpp.card_storage.requests import CardStorageRequest
from addonpayments.hpp.hpp import Hpp

# create request to send
request = CardStorageRequest(
    merchant_id='my_merchant_id',
    amount=100,
    currency='EUR',
    auto_settle_flag=True,
    card_storage_enable=True,
    offer_save_card=True,
```

(continues on next page)

(continued from previous page)

```
    payer_exists=False,
    payer_ref='payer_ref',
    pmt_ref='pmt_ref'
)
# create request/response handler
hpp = Hpp('my_secrect')
# parse request object to json object
json_request = hpp.request_to_json(req)
```

Display stored cards to the customer

For more detailed information, please see the section Display stored cards to the customer (Card storage) into Addon Payments developer hub.

1. Create the Display stored cards request.
2. Auto-generation of the fields and transformation to JSON.

```
from addonpayments.hpp.card_storage.requests import DisplayCardsRequest
from addonpayments.hpp.hpp import Hpp

# create request to send
request = DisplayCardsRequest(
    merchant_id='my_merchant_id',
    amount=100,
    currency='EUR',
    auto_settle_flag=True,
    hpp_select_stored_card='payer_ref',
    payer_exists=True,
    offer_save_card=True,
)
# create request/response handler
hpp = Hpp('my_secrect')
# parse request object to json object
json_request = hpp.request_to_json(req)
```

Recurring

For more detailed information, please see the section Recurring (Card storage) into Addon Payments developer hub.

1. Create the Recurring request.
2. Auto-generation of the fields and transformation to JSON.

```
from addonpayments.hpp.card_storage.requests import RecurringPaymentRequest
from addonpayments.hpp.hpp import Hpp

# create request to send
request = RecurringPaymentRequest(
    merchant_id='my_merchant_id',
    amount=100,
    currency='EUR',
    auto_settle_flag=True,
    card_storage_enable=True,
```

(continues on next page)

(continued from previous page)

```
    offer_save_card=True,
    payer_exists=True,
    payer_ref='payer_ref',
    pmt_ref='pmt_ref'
)
# create request/response handler
hpp = Hpp('my_secrect')
# parse request object to json object
json_request = hpp.request_to_json(req)
```

1.2.4 API Reference

The API generates and send XML requests to AddonPayments. For more information about API, please visit the API reference from Addon Payments developer hub.

Then we will see the steps to follow to generate XML requests for each type of transaction and send to Addon Payments.

Note: There are mandatory fields that are auto-generated by the SDK generation utilities

- sha1hash: The SDK automatically generates a SHA-1 digital signature for each JSON requests.
- orderid: The transaction ID can be defined by user or auto-generated by the SDK.
- timestamp: The transaction timestamp can be defined by user or auto-generated by the SDK.

1.2.4.1 API Client

All requests are send to Addon Payments with ApiClient class. ApiClient carries out the following actions:

- Generate security hash
- Parse request object to XML
- Send POST request
- Receive AddonPayments response
- Unparse XML response to ApiResponse
- Validate hash

```
from addonpayments.api.client import ApiClient

# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

1.2.4.2 Process a Payment

Please see the Process a Payment section into Addon Payments developer hub.

Authorisation

For more detailed information, please see the section Authorisation into Addon Payments developer hub.

1. Create the Authorisation request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Cvn, CardWithCvn
from addonpayments.api.client import ApiClient
from addonpayments.api.payment.requests import AuthRequest

# Create API elements
cvn = Cvn(number='123', presind=1)
card = CardWithCvn(
    type='VISA', number='4263970000005262',
    expdate='1220', cvn=cvn, chname='card owner'
)
# create request to send
request = AuthRequest(
    merchantid='my_merchant_id',
    card=card,
    amount=100,
    currency='EUR',
    autosettle='1',
    comments=['comment one', 'comment two']
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

1.2.4.3 Card storage

Please see the Card storage section into Addon Payments developer hub.

Receipt In

For more detailed information, please see the section Receipt-in into Addon Payments developer hub.

1. Create the Receipt-in request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements PaymentData
from addonpayments.api.client import ApiClient
from addonpayments.api.card_storage.requests import ReceiptInRequest

# create request to send
request = ReceiptInRequest(
    merchantid='my_merchant_id,
    amount=100,
    currency='EUR',
    autosettle='1',
```

(continues on next page)

(continued from previous page)

```
    comments=['comment one', 'comment two'],
    payerref='my_payer_ref',
    paymentmethod='my_payment_ref',
    paymentdata=PaymentData('123')
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Verify Stored Card Enrolled

For more detailed information, please see the section Verify Stored Card Enrolled into Addon Payments developer hub.

1. Create the Verify Stored Card Enrolled request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.client import ApiClient
from addonpayments.api.card_storage.requests import RealVaultThreeDsVerifyEnrolled

# create request to send
request = RealVaultThreeDsVerifyEnrolled(
    merchantid='my_merchant_id,
    amount=100,
    currency='EUR',
    payerref='my_payer_ref',
    paymentmethod='my_payment_ref',
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Payer new

For more detailed information, please see the section Payer new into Addon Payments developer hub.

1. Create the Payer new request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Address, PhoneNumbers, Payer
from addonpayments.api.client import ApiClient
from addonpayments.api.card_storage.requests import PayerNewRequest

# Create API elements
address = Address(
    line1='Flat 123',
    line2='House 456',
    line3='The Cul-De-Sac',
    city='Palma',
```

(continues on next page)

(continued from previous page)

```
    county='Balearic islands',
    postcode='07121',
    code='ES',
    country='Spain'
)
phone_numbers = PhoneNumbers(
    home='+34312345678',
    work='+3431987654321',
    fax='+124546871258',
    mobile='+25544778544'
)
payer = Payer(
    type='Retail',
    ref=BaseTest().payer_ref,
    title='Mr',
    firstname='James',
    surname='Mason',
    company='Addon Payments',
    address=address,
    phonenumbers=phone_numbers,
    email='text@example.com',
    comments=['comment one', 'comment two']
)

# create request to send
request = PayerNewRequest(
    merchantid='my_merchant_id',
    payer=payer
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Payer edit

For more detailed information, please see the section Payer edit into Addon Payments developer hub.

1. Create the Payer edit request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Address, PhoneNumbers, Payer
from addonpayments.api.client import ApiClient
from addonpayments.api.card_storage.requests import PayerEditRequest

# Create API elements
address = Address(
    line1='Flat 123',
    line2='House 456',
    line3='The Cul-De-Sac',
    city='Palma',
    county='Balearic islands',
    postcode='07121',
    code='ES',
```

(continues on next page)

(continued from previous page)

```
    country='Spain'
)
phone_numbers = PhoneNumbers(
    home='+34312345678',
    work='+3431987654321',
    fax='+124546871258',
    mobile='+25544778544'
)
payer = Payer(
    type='Retail',
    ref=BaseTest().payer_ref,
    title='Mr',
    firstname='James',
    surname='Mason',
    company='Addon Payments',
    address=address,
    phonenumbers=phone_numbers,
    email='text@example.com',
    comments=['comment one', 'comment two']
)

# create request to send
request = PayerEditRequest(
    merchantid=s'my_merchant_id,
    payer=payer,
    comments=['comment one', 'comment two'],
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Card new

For more detailed information, please see the section Card new into Addon Payments developer hub.

1. Create the Card new request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Cvn, CardWithCvn
from addonpayments.api.client import ApiClient
from addonpayments.api.card_storage.requests import CardNewRequest

# Create API elements
cvn = Cvn(number='123', presind=1)
card = CardWithCvn(
    type='VISA', number='4263970000005262',
    expdate='1220', cvn=cvn, chname='card owner'
)

# create request to send
request = CardNewRequest(
    merchantid='my_merchant_id',
    comments=['comment one', 'comment two'],
```

(continues on next page)

(continued from previous page)

```
    card=card
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Card update

For more detailed information, please see the section Card update into Addon Payments developer hub.

1. Create the Card update request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Cvn, CardWithCvn
from addonpayments.api.client import ApiClient
from addonpayments.api.card_storage.requests import CardUpdateRequest

# Create API elements
cvn = Cvn(number='123', presind=1)
card = CardWithCvn(
    type='VISA', number='4263970000005262',
    expdate='1220', cvn=cvn, chname='card owner updated'
)

# create request to send
request = CardUpdateRequest(
    merchantid='my_merchant_id',
    comments=['comment one', 'comment two'],
    card=card
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Card cancel

For more detailed information, please see the section Card cancel into Addon Payments developer hub.

1. Create the Card cancel request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Cvn, CardWithCvn
from addonpayments.api.client import ApiClient
from addonpayments.api.card_storage.requests import CardCancelRequest

# Create API elements
cvn = Cvn(number='123', presind=1)
card = CardWithCvn(
    type='VISA', number='4263970000005262',
```

(continues on next page)

(continued from previous page)

```
    expdate='1220', cvn=cvn, chname='card owner'
)

# create request to send
request = CardCancelRequest(
    merchantid='my_merchant_id',
    comments=['comment one', 'comment two'],
    card=card
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Card DCC rate

For more detailed information, please see the section Card Dccrate into Addon Payments developer hub.

1. Create the Card DCC rate request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import DccInfo
from addonpayments.api.client import ApiClient
from addonpayments.api.card_storage.requests import CardDccRateRequest

# Create API elements
dcc_info = DccInfo(ccp='ccp', type=1)

# create request to send
request = CardDccRateRequest(
    merchantid='my_merchant_id',
    amount=100,
    currency='EUR',
    payerref='my_payer_ref',
    paymentmethod='my_payment_ref',
    comments=['comment one', 'comment two'],
    dccinfo=dcc_info
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Recurring

For more detailed information, please see the section Recurring into Addon Payments developer hub.

1. Create the Authorisation with recurring request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Cvn, CardWithCvn, Recurring
from addonpayments.api.client import ApiClient
from addonpayments.api.card_storage.requests import AuthRequestWithRecurring

# Create API elements
cvn = Cvn(number='123', presind=1)
card = CardWithCvn(
    type='VISA', number='4263970000005262',
    expdate='1220', cvn=cvn, chname='card owner'
)
recurring = Recurring(type='fixed', sequence='first')
# create request to send
request = AuthRequestWithRecurring(
    merchantid='my_merchant_id,
    card=card,
    amount=100,
    currency='EUR',
    autosettle='1',
    comments=['comment one', 'comment two'],
    recurring=recurring
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

1.2.4.4 3D Secure

Please see the 3D Secure section into Addon Payments developer hub.

3DS Verify enrolled

For more detailed information, please see the section 3DS verify enrolled into Addon Payments developer hub.

1. Create the 3ds-verify enrolled request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Card
from addonpayments.api.client import ApiClient
from addonpayments.api.three_ds.requests import ThreeDsVerifyEnrolled

# Create API elements
card = Card(type='VISA', number='4263970000005262', expdate='1220', chname='card owner
↪')

# create request to send
request = ThreeDsVerifyEnrolled(
    merchantid='my_merchant_id,
    card=card,
    amount=100,
    currency='EUR',
    comments=['comment one', 'comment two']
)
```

(continues on next page)

(continued from previous page)

```
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

3DS Verify sig

For more detailed information, please see the section 3DS verify sig into Addon Payments developer hub.

1. Create the 3ds-verify sig request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Card
from addonpayments.api.client import ApiClient
from addonpayments.api.three_ds.requests import ThreeDsVerifySig

# Create API elements
card = Card(type='VISA', number='4263970000005262', expdate='1220', chname='card owner
↪')

# create request to send
request = ThreeDsVerifySig(
    merchantid='my_merchant_id,
    card=card,
    amount=100,
    currency='EUR',
    pares=
↪"eJzNV1uTokoSft9fMdHn0ZiB4qIwQbtR3FFBQZDLGzcB5SagIL9+sZ3p6T3RJ87sPmzsE1UZWV9+WZUfWcX8cyjyL7
↪"
          "e4abOqfH0B39CXL3EZVlFWJq8vlil+pV7+ufwHY6ZNHPP7OLw28ZJR47b1k/hLFr2+xEc/
↪9vEAhCGOE4uIprAopDGMWvg0PQ/x6"
          "GXJ7KARt2/OACUoFCepyfgj5nIK+Q1jkJ/TCbsJU7/slowfXlhFWxIowFCUQX5MmSJuFH5JE/
↪icxDHAIM85g/xauLs+Ru1EdMii"
          "3H6Tyn0zOkuod3kv9kC3i2UPW2554lKAn9amWYH8qM/dCtjLaXfnarYXV/
↪dqu0d52882wSnSKcAgz9rHu9S3DqXn8twfdOpfTqaf"
          "qa4f1XpxJG1Xy3/R/J//fkQvLRcLTizc3aSLq0rzqvpqxD1u6dqYlk4nn6UbO1T2occZ/
↪1a2m4GHONO/F2ZAbSHoZy1FrZBV8dBW"
          "gOeIcrGV8eVfYULSiXvlxbkOlsFscbyZbBjptNbaom9pJbnYBjm/
↪R6JAVxzRchr2H9fecfBpMkisAN0mwzD3ZpvhCrORlmLup07N"

↪"92G30IF5XBJrMDkqXWpUPv241XDolt8Lv4fOdtgasvJOF4ptw6sDih0bmRD5Q4b+prKiO4FZFjJjbH+8YwnUiGyjDjJQcZhjLH
↪"

↪"Dqjl4ratXNR2eXlZrorTf62mx7AwQhmOdnAz2Tzv0SEFq9GB17JAh003JnbbrgXzUU08vzGRtTA+1uR/
↪ipXJBfdz7k/R7464b49r"
          "J7e24+XiMfn6H/AvBh19Q="
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Authorisation with 3D Secure

For more detailed information, please see the section Authorisation with 3D Secure into Addon Payments developer hub.

1. Create the Authorisation with 3D Secure request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Cvn, CardWithCvn, Mpi
from addonpayments.api.client import ApiClient
from addonpayments.api.three_ds.requests import AuthRequestWithThreeDS

# Create API elements
cvn = Cvn(number='123', presind=1)
card = CardWithCvn(
    type='VISA', number='4263970000005262',
    expdate='1220', cvn=cvn, chname='card owner'
)
mpi = Mpi(cavv='AAACBllleHchZTBWIGV4AAAAAAA', xid='crqAeMwkEL9r4POdxpByWJ1', eci=5)
# create request to send
request = AuthRequestWithThreeDS(
    merchantid='my_merchant_id,
    card=card,
    amount=100,
    currency='EUR',
    autosettle='1',
    comments=['comment one', 'comment two'],
    mpi=mpi
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

1.2.4.5 Transaction management

Please see the Transaction management section into Addon Payments developer hub.

Settle

For more detailed information, please see the section Settle into Addon Payments developer hub.

1. Create the Settle request.
2. Send XML request to AddonPayments

```
from addonpayments.api.client import ApiClient
from addonpayments.api.transaction_management.requests import Settle

# create request to send
request = Settle(
    merchantid='my_merchant_id',
    pasref='pasref_from_original_transaction',
    amount=100,
```

(continues on next page)

(continued from previous page)

```
    comments=['comment one', 'comment two']
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Rebate

For more detailed information, please see the section Rebate into Addon Payments developer hub.

1. Create the Rebate request.
2. Send XML request to AddonPayments

```
from addonpayments.api.client import ApiClient
from addonpayments.api.transaction_management.requests import Rebate

# create request to send
request = Rebate(
    merchantid='my_merchant_id',
    amount=100,
    currency='EUR',
    pasref='pasref_from_original_transaction',
    authcode='authcode_from_original_transaction',
    refundhash='sha1_hash_of_rebate_password',
    comments=['comment one', 'comment two']
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Void

For more detailed information, please see the section Void into Addon Payments developer hub.

1. Create the Void request.
2. Send XML request to AddonPayments

```
from addonpayments.api.client import ApiClient
from addonpayments.api.transaction_management.requests import Vpid

# create request to send
request = Settle(
    merchantid='my_merchant_id',
    pasref='pasref_from_original_transaction',
    comments=['comment one', 'comment two']
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

1.2.4.6 Dynamic Currency Conversion (DCC)

Please see the Dynamic Currency Conversion (DCC) section into Addon Payments developer hub.

DCC Rate

For more detailed information, please see the section DCC Rate into Addon Payments developer hub.

1. Create the DCC rate request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Card, DccInfoWithRateType
from addonpayments.api.client import ApiClient
from addonpayments.api.dcc.requests import DccRate

# Create API elements
card = Card(type='VISA', number='4263970000005262', expdate='1220', chname='card owner
↪')
dcc_info = DccInfoWithRateType(ccp='ccp', type=1, ratetype='S')

# create request to send
request = DccRate(
    merchantid='my_merchant_id',
    card=card,
    amount=100,
    currency='EUR',
    dccinfo=dcc_info,
    comments=['comment one', 'comment two']
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```

Authorisation with DCC Information

For more detailed information, please see the section DCC Information into Addon Payments developer hub.

1. Create the Authorisation with DCC Information request.
    - Create the required API elements for this request
2. Send XML request to AddonPayments

```
from addonpayments.api.elements import Cvn, CardWithCvn, Mpi
from addonpayments.api.client import ApiClient
from addonpayments.api.dcc.requests import AuthRequestWithDccInfo

# Create API elements
cvn = Cvn(number='123', presind=1)
card = CardWithCvn(
    type='VISA', number='4263970000005262',
    expdate='1220', cvn=cvn, chname='card owner'
)
dcc_info = DccInfoWithAmount(
```

(continues on next page)

(continued from previous page)

```
    ccp='ccp', type=1, ratetype='S',
    rate='1.6728', amount=100, currency='EUR'
)
# create request to send
request = AuthRequestWithDccInfo(
    merchantid=self.merchant_id,
    card=card,
    amount=100,
    currency='EUR',
    autosettle='1',
    comments=['comment one', 'comment two'],
    dccinfo=dcc_info
)
# Send request to Addon Payments
client = ApiClient('my_secrect')
response = client.send(request)
```