SSLCommerz Client

Release 0.4.0

Utsob Roy

Mar 19, 2024

CONTENTS

CHAPTER

ONE

OVERVIEW

A Sane SSLCommerz Client for Python.

- Free software: MIT license

1.1 Warning

Under development. Is not usable. At all.

1.2 Why?

There are at least 5 sdk/library/client for SSLCommerz in PyPI right now including an official one. However, we wanted to create an API client that will take care of a major part of validation (thanks to `pydantic`), will feel intuitive, and allow you to access and inspect data in ease.

1.3 Features

- Pydantic powered dataclasses for every request (in request one can also use `dict` that will be converted to a *dataclass*) and response.
- IPN validation.
- Methods for all official endpoints.

1.4 Installation

```
pip install sslcommerz-client
```

You can also install the in-development version with:

```
pip install https://gitlab.com/codesigntheory/python-sslcommerz-client/-/archive/master/
↪python-sslcommerz-client-master.zip
```

1.5 Documentation

https://python-sslcommerz-client.readthedocs.io/

1.6 Development

To run all the tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER

TWO

INSTALLATION AND USAGE

2.1 Installation

At the command line:

```
pip install sslcommerz-client
```

2.2 Usage

To use SSLCommerz Client in a project:

```
from sslcommerz_client import SSLCommerzClient
```

2.2.1 Initiate Client

To initiate a client:

```
from sslcommerz_client import SSLCommerzClient

client = SSLCommerzClient(
    store_id="YOUR_STORE_ID",
    store_passwd="YOUR_STORE_PASSWORD",
    sandbox=True // default false
)
```

2.2.2 Initiate a Session

To Initiate a Session:

```
post_data = {
    "total_amount": 100,
    "currency": "BDT",
    "tran_id": "221122",
    "product_category": "fashion",
    "success_url": "https://example.com",
    "fail_url": "https://example.com",
```

(continues on next page)

(continued from previous page)

```
    "cancel_url": "https://example.com",
    "cus_name": "Jon Osterman",
    "cus_email": "jon@osterman.com",
    "shipping_method": "NO",
    "num_of_item": 1,
    "product_name": "Fancy Pants",
    "product_category": "Cloth",
    "product_profile": "physical-goods",
    "cus_add1": "Some Address",
    "cus_city": "Dhaka",
    "cus_country": "Bangladesh",
    "cus_phone": "01558221870",
}

response = client.initiateSession(post_data)
```

`response` will be an *APIResponse* object with `raw_data` - the actual response, `status_code` for convenience, and an `response` - a *PaymentInitResponse*. One can use *PaymentInitResponse* as is or create a dict or json from it. For more, consult `pydantic` documentation.

2.2.3 Validate IPN

To validate an IPN response:

```
validation = client.validate_IPN(data) // data: response data as a dict.
```

`validation` will be an *IPNValidationStatus* with validation status as `status` and the response as *IPNResponse*.

2.2.4 Getting Order Validation Data

```
data = {"val_id": "some_val_id"}
validation_response = client.get_order_validation_data(data)
```

`validation_response` will be an *APIResponse* object with `raw_data` - the actual response, `status_code` for convenience, and an `response` - a *OrderValidationResponse*.

2.2.5 Initiate Refund

```
data = {
    "bank_tran_id": "some_tran_id",
    "refund_amount": "100.00",
    "refund_remarks": "faulty product"
}
refund_response = client.initiate_refund(data)
```

`refund_response` will be an *APIResponse* object with `raw_data` - the actual response, `status_code` for convenience, and an `response` - a *RefundInitiateResponse*.

2.2.6 Get Refund Data

```
refund_response = client.get_refund_data("refund_ref_id")
```

`refund_response` will be an *APIResponse* object with `raw_data` - the actual response, `status_code` for convenience, and an `response` - a *RefundResponse*.

2.2.7 Get Transaction by Session

```
transaction_response = client.get_transaction_by_session("sessionkey")
```

`transaction_response` will be an *APIResponse* object with `raw_data` - the actual response, `status_code` for convenience, and an `response` - a *TransactionBySessionResponse*.

2.2.8 Get Transactions by ID

```
transaction_response = client.get_transaction_by_id("tran_id")
```

`transaction_response` will be an *APIResponse* object with `raw_data` - the actual response, `status_code` for convenience, and an `response` - a *TransactionsByIDResponse*.

CHAPTER

THREE

REFERENCE

3.1 sslcommerz_client package

3.1.1 Submodules

3.1.2 sslcommerz_client.client module

class sslcommerz_client.client.**SSLCommerzClient**(*store_id: str*, *store_passwd: str*, *sandbox: bool = False*)

Bases: `object`

property baseURL

property credential

get_order_validation_data(*data: dict | OrderValidationPostData | IPNResponse*)

Get Order validation data from API.

get_refund_data(*refund_ref_id: str*)

Get existing refund data.

get_transaction_by_id(*tran_id: str*)

Get transactions by transaction id.

get_transaction_by_session(*sessionkey: str*)

Get a transaction by sessionkey.

initiate_refund(*data: dict | RefundRequestPostData*)

Initiate a refund.

initiate_session(*postData: PaymentInitPostData | dict*)

Initiates an session.

validate_IPN(*data: dict*)

Validate an IPN response.

3.1.3 sslcommerz_client.dataclasses module

class sslcommerz_client.dataclasses.**APIConnectEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: str, Enum

DONE = 'DONE'

FAILED = 'FAILED'

INACTIVE = 'INACTIVE'

INVALID_REQUEST = 'INVALID_REQUEST'

class sslcommerz_client.dataclasses.**APIResponse**(***, *raw_data: Any = None*, *status_code: int*, *response:* OrderValidationResponse | IPNResponse | PaymentInitResponse | RefundResponse | RefundInitiateResponse | TransactionBySessionResponse | TransactionsByIDResponse | *None = None*)

Bases: BaseModel

dataclass for api response complete with raw response data, status_code and one of response objects for easy introspection.

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {'arbitrary_types_allowed': True}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'raw_data': FieldInfo(annotation=Any, required=False), 'response': FieldInfo(annotation=Union[OrderValidationResponse, IPNResponse, PaymentInitResponse, RefundResponse, RefundInitiateResponse, TransactionBySessionResponse, TransactionsByIDResponse, NoneType], required=False), 'status_code': FieldInfo(annotation=int, required=True)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

raw_data: Any

response: *OrderValidationResponse* | *IPNResponse* | *PaymentInitResponse* | *RefundResponse* | *RefundInitiateResponse* | *TransactionBySessionResponse* | *TransactionsByIDResponse* | None

status_code: int

class sslcommerz_client.dataclasses.**BaseOrderResponse**(*, *tran_date: datetime*, *tran_id: str*, *val_id: str*, *amount: Decimal*, *store_amount: Decimal*, *card_type: str*, *card_no: str*, *currency: str*, *bank_tran_id: str*, *card_issuer: str*, *card_brand: str*, *card_issuer_country: str*, *card_issuer_country_code: str*, *currency_type: str*, *currency_amount: Decimal*, *currency_rate: Decimal*, *risk_level:* RiskLevelEnum, *risk_title: str*, *error: str | None = None*, *base_fair: Decimal | None = None*, *card_sub_brand: str | None = None*, *value_a: str | None = None*, *value_b: str | None = None*, *value_c: str | None = None*, *value_d: str | None = None*)

Bases: BaseModel

Base dataclass for Order and IPN.

amount: Decimal

bank_tran_id: str

base_fair: Decimal | None

card_brand: str

card_issuer: str

card_issuer_country: str

card_issuer_country_code: str

card_no: str

card_sub_brand: str | None

card_type: str

currency: str

currency_amount: Decimal

currency_rate: Decimal

currency_type: str

error: str | None

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'amount': FieldInfo(annotation=Decimal, required=True), 'bank_tran_id': FieldInfo(annotation=str, required=True), 'base_fair': FieldInfo(annotation=Union[Decimal, NoneType], required=False), 'card_brand': FieldInfo(annotation=str, required=True), 'card_issuer': FieldInfo(annotation=str, required=True), 'card_issuer_country': FieldInfo(annotation=str, required=True), 'card_issuer_country_code': FieldInfo(annotation=str, required=True), 'card_no': FieldInfo(annotation=str, required=True), 'card_sub_brand': FieldInfo(annotation=Union[str, NoneType], required=False), 'card_type': FieldInfo(annotation=str, required=True), 'currency': FieldInfo(annotation=str, required=True), 'currency_amount': FieldInfo(annotation=Decimal, required=True), 'currency_rate': FieldInfo(annotation=Decimal, required=True), 'currency_type': FieldInfo(annotation=str, required=True), 'error': FieldInfo(annotation=Union[str, NoneType], required=False), 'risk_level': FieldInfo(annotation=RiskLevelEnum, required=True), 'risk_title': FieldInfo(annotation=str, required=True), 'store_amount': FieldInfo(annotation=Decimal, required=True), 'tran_date': FieldInfo(annotation=datetime, required=True), 'tran_id': FieldInfo(annotation=str, required=True), 'val_id': FieldInfo(annotation=str, required=True), 'value_a': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_b': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_c': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_d': FieldInfo(annotation=Union[str, NoneType], required=False)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

risk_level: *RiskLevelEnum*

risk_title: str

store_amount: Decimal

tran_date: datetime

tran_id: str

val_id: str

value_a: str | None

value_b: str | None

value_c: str | None

value_d: str | None

class sslcommerz_client.dataclasses.**BooleanIntEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: int, Enum

FALSE = 0

TRUE = 1

class sslcommerz_client.dataclasses.**CartItem**(*, *product: str*, *quantity: int*, *amount: Decimal*)

Bases: BaseModel

Dataclass for cart items in PaymentInitPostData.

amount: Decimal

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'amount': FieldInfo(annotation=Decimal, required=True), 'product': FieldInfo(annotation=str, required=True), 'quantity': FieldInfo(annotation=int, required=True)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

classmethod not_more_than_255(*v: str*, *info: ValidationInfo*)

product: str

quantity: int

classmethod valid_decimal(*v*, *info: ValidationInfo*)

class sslcommerz_client.dataclasses.**Credential**(*, *store_id: str*, *store_passwd: str*)

Bases: BaseModel

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'store_id': FieldInfo(annotation=str, required=True), 'store_passwd': FieldInfo(annotation=str, required=True)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

store_id: str

store_passwd: str

class sslcommerz_client.dataclasses.**EMIOptionsEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: int, Enum

NINE_MONTHS = 9

SIX_MONTHS = 6

THREE_MONTHS = 3

class sslcommerz_client.dataclasses.**EMIOptionsResponseEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: str, Enum

NINE_MONTHS = '9'

NONE = '0'

SIX_MONTHS = '6'

THREE_MONTHS = '3'

class sslcommerz_client.dataclasses.**Gateway**(***, *name: str*, *type: str*, *logo: str | None = None*, *gw: str | None = None*, *r_flag: str | None = None*, *redirectGatewayURL: str | None = None*)

Bases: BaseModel

gw: str | None

logo: str | None

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'gw': FieldInfo(annotation=Union[str, NoneType], required=False), 'logo': FieldInfo(annotation=Union[str, NoneType], required=False), 'name': FieldInfo(annotation=str, required=True), 'r_flag': FieldInfo(annotation=Union[str, NoneType], required=False), 'redirectGatewayURL': FieldInfo(annotation=Union[str, NoneType], required=False), 'type': FieldInfo(annotation=str, required=True)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

name: str

r_flag: str | None

redirectGatewayURL: str | None

type: str

class sslcommerz_client.dataclasses.**IPNOrderStatusEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: str, Enum

CANCELLED = 'CANCELLED'

EXPIRED = 'EXPIRED'

FAILED = 'FAILED'

UNATTEMPTED = 'UNATTEMPTED'

VALID = 'VALID'

class sslcommerz_client.dataclasses.**IPNResponse**(*, *tran_date: datetime*, *tran_id: str*, *val_id: str*, *amount: Decimal*, *store_amount: Decimal*, *card_type: str*, *card_no: str*, *currency: str*, *bank_tran_id: str*, *card_issuer: str*, *card_brand: str*, *card_issuer_country: str*, *card_issuer_country_code: str*, *currency_type: str*, *currency_amount: Decimal*, *currency_rate: Decimal*, *risk_level:* RiskLevelEnum, *risk_title: str*, *error: str | None = None*, *base_fair: Decimal | None = None*, *card_sub_brand: str | None = None*, *value_a: str | None = None*, *value_b: str | None = None*, *value_c: str | None = None*, *value_d: str | None = None*, *store_id: str*, *status:* IPNOrderStatusEnum, *verify_sign: str*, *verify_key: str*, *verify_sign_sha2: str | None = None*)

Bases: *BaseOrderResponse*

IPN response dataclass with validation

get_hash(*credential: ~sslcommerz_client.dataclasses.Credential*, *hasher=<built-in function openssl_md5>*)

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'amount': FieldInfo(annotation=Decimal, required=True), 'bank_tran_id': FieldInfo(annotation=str, required=True), 'base_fair': FieldInfo(annotation=Union[Decimal, NoneType], required=False), 'card_brand': FieldInfo(annotation=str, required=True), 'card_issuer': FieldInfo(annotation=str, required=True), 'card_issuer_country': FieldInfo(annotation=str, required=True), 'card_issuer_country_code': FieldInfo(annotation=str, required=True), 'card_no': FieldInfo(annotation=str, required=True), 'card_sub_brand': FieldInfo(annotation=Union[str, NoneType], required=False), 'card_type': FieldInfo(annotation=str, required=True), 'currency': FieldInfo(annotation=str, required=True), 'currency_amount': FieldInfo(annotation=Decimal, required=True), 'currency_rate': FieldInfo(annotation=Decimal, required=True), 'currency_type': FieldInfo(annotation=str, required=True), 'error': FieldInfo(annotation=Union[str, NoneType], required=False), 'risk_level': FieldInfo(annotation=RiskLevelEnum, required=True), 'risk_title': FieldInfo(annotation=str, required=True), 'status': FieldInfo(annotation=IPNOrderStatusEnum, required=True), 'store_amount': FieldInfo(annotation=Decimal, required=True), 'store_id': FieldInfo(annotation=str, required=True), 'tran_date': FieldInfo(annotation=datetime, required=True), 'tran_id': FieldInfo(annotation=str, required=True), 'val_id': FieldInfo(annotation=str, required=True), 'value_a': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_b': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_c': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_d': FieldInfo(annotation=Union[str, NoneType], required=False), 'verify_key': FieldInfo(annotation=str, required=True), 'verify_sign': FieldInfo(annotation=str, required=True), 'verify_sign_sha2': FieldInfo(annotation=Union[str, NoneType], required=False)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

status: *IPNOrderStatusEnum*

store_id: str

validate_against_credential(*credential:* Credential | *dict*)

verify_key: str

verify_sign: str

verify_sign_sha2: str | None

class sslcommerz_client.dataclasses.**IPNValidationStatus**(*, *status: bool*, *response:* IPNResponse)

Bases: BaseModel

IPN validation result's dataclass.

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {'arbitrary_types_allowed': True}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'response': FieldInfo(annotation=IPNResponse, required=True), 'status': FieldInfo(annotation=bool, required=True)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

response: *IPNResponse*

status: bool

class sslcommerz_client.dataclasses.**MultiCardNamesEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: str, Enum

ABBANK = 'abbank'

AMEX_CARD = 'amexcard'

BANK_ASIA = 'bankasia'

BKASH = 'bkash'

BRAC_MASTER = 'brac_master'

BRAC_VISA = 'brac_visa'

CITY = 'city'

CITY_AMEX = 'city_amex'

CITY_MASTER = 'city_master'

CITY_VISA = 'city_visa'

DBBL_MASTER = 'dbbl_master'

DBBL_MOBILE_BANKING = 'dbblmobilebanking'

DBBL_NEXUS = 'dbbl_nexus'

DBBL_VISA = 'dbbl_visa'

EBL_MASTER = 'ebl_master'

EBL_VISA = 'ebl_visa'

IBBL = 'ibbl'

INTERNET_BANK = 'internetbank'

MASTER_CARD = 'mastercard'

MOBILE_BANK = 'mobilebank'

MTBL = 'mtbl'

OTHER_CARD = 'othercard'

QCASH = 'qcash'

SBL_MASTER = 'sbl_master'

SBL_VISA = 'sbl_visa'

TAPNPAY = 'tapnpay'

UPAY = 'upay'

VISA_CARD = 'visacard'

class sslcommerz_client.dataclasses.**OrderStatusEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: str, Enum

INVALID_TRANSACTION = 'INVALID_TRANSACTION'

VALID = 'VALID'

VALIDATED = 'VALIDATED'

class sslcommerz_client.dataclasses.**OrderValidationPostData**(***, *val_id: str*, *v: int | None = None*)

Bases: BaseModel

Dataclass for Order validation API post data.

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'v': FieldInfo(annotation=Union[int, NoneType], required=False), 'val_id': FieldInfo(annotation=str, required=True)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

classmethod not_more_than_fifty(*v*, *info: ValidationInfo*)

v: int | None

val_id: str

classmethod validate_v(*v*)

class sslcommerz_client.dataclasses.**OrderValidationResponse**(*, *tran_date: datetime*, *tran_id: str*, *val_id: str*, *amount: Decimal*, *store_amount: Decimal*, *card_type: str*, *card_no: str*, *currency: str*, *bank_tran_id: str*, *card_issuer: str*, *card_brand: str*, *card_issuer_country: str*, *card_issuer_country_code: str*, *currency_type: str*, *currency_amount: Decimal*, *currency_rate: Decimal*, *risk_level:* RiskLevelEnum, *risk_title: str*, *error: str | None = None*, *base_fair: Decimal | None = None*, *card_sub_brand: str | None = None*, *value_a: str | None = None*, *value_b: str | None = None*, *value_c: str | None = None*, *value_d: str | None = None*, *status:* OrderStatusEnum, *emi_instalment:* EMIOptionsResponseEnum, *discount_amount: Decimal*, *discount_percentage: Decimal*, *discount_remarks: str*)

Bases: *BaseOrderResponse*

Order validation response.

discount_amount: Decimal

discount_percentage: Decimal

discount_remarks: str

emi_instalment: *EMIOptionsResponseEnum*

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

```
model_fields: ClassVar[dict[str, FieldInfo]] = {'amount':
FieldInfo(annotation=Decimal, required=True), 'bank_tran_id':
FieldInfo(annotation=str, required=True), 'base_fair':
FieldInfo(annotation=Union[Decimal, NoneType], required=False), 'card_brand':
FieldInfo(annotation=str, required=True), 'card_issuer': FieldInfo(annotation=str,
required=True), 'card_issuer_country': FieldInfo(annotation=str, required=True),
'card_issuer_country_code': FieldInfo(annotation=str, required=True), 'card_no':
FieldInfo(annotation=str, required=True), 'card_sub_brand':
FieldInfo(annotation=Union[str, NoneType], required=False), 'card_type':
FieldInfo(annotation=str, required=True), 'currency': FieldInfo(annotation=str,
required=True), 'currency_amount': FieldInfo(annotation=Decimal, required=True),
'currency_rate': FieldInfo(annotation=Decimal, required=True), 'currency_type':
FieldInfo(annotation=str, required=True), 'discount_amount':
FieldInfo(annotation=Decimal, required=True), 'discount_percentage':
FieldInfo(annotation=Decimal, required=True), 'discount_remarks':
FieldInfo(annotation=str, required=True), 'emi_instalment':
FieldInfo(annotation=EMIOptionsResponseEnum, required=True), 'error':
FieldInfo(annotation=Union[str, NoneType], required=False), 'risk_level':
FieldInfo(annotation=RiskLevelEnum, required=True), 'risk_title':
FieldInfo(annotation=str, required=True), 'status':
FieldInfo(annotation=OrderStatusEnum, required=True), 'store_amount':
FieldInfo(annotation=Decimal, required=True), 'tran_date':
FieldInfo(annotation=datetime, required=True), 'tran_id': FieldInfo(annotation=str,
required=True), 'val_id': FieldInfo(annotation=str, required=True), 'value_a':
FieldInfo(annotation=Union[str, NoneType], required=False), 'value_b':
FieldInfo(annotation=Union[str, NoneType], required=False), 'value_c':
FieldInfo(annotation=Union[str, NoneType], required=False), 'value_d':
FieldInfo(annotation=Union[str, NoneType], required=False)}
```

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

status: *OrderStatusEnum*

class sslcommerz_client.dataclasses.**PaymentInitPostData**(*, *total_amount: Decimal*, *currency: str*, *tran_id: str*, *product_category: str*, *success_url: Url*, *fail_url: Url*, *cancel_url: Url*, *emi_option:* BooleanIntEnum *= BooleanIntEnum.FALSE*, *cus_name: str*, *cus_email: str*, *cus_add1: str*, *cus_city: str*, *cus_country: str*, *cus_phone: str*, *shipping_method:* ShippingMethodEnum *= ShippingMethodEnum.YES*, *num_of_item: int*, *product_name: str*, *product_profile:* ProductProfileEnum, *ipn_url: str | None = None*, *multi_card_name:* MultiCardNamesEnum *| None = None*, *allowed_bin: str | None = None*, *emi_max_inst_option:* EMIOptionsEnum *| None = None*, *emi_selected_inst:* EMIOptionsEnum *| None = None*, *emi_allow_only: int | None = None*, *cus_add2: str | None = None*, *cus_postcode: str | None = None*, *cus_state: str | None = None*, *cus_fax: str | None = None*, *ship_name: str | None = None*, *ship_add1: str | None = None*, *ship_add2: str | None = None*, *ship_city: str | None = None*, *ship_postcode: str | None = None*, *ship_country: str | None = None*, *ship_phone: str | None = None*, *ship_state: str | None = None*, *hours_till_departure: str | None = None*, *flight_type: str | None = None*, *pnr: str | None = None*, *journey_from_to: str | None = None*, *third_party_booking: str | None = None*, *hotel_name: str | None = None*, *length_of_stay: str | None = None*, *check_in_time: str | None = None*, *hotel_city: str | None = None*, *product_type: str | None = None*, *topup_number: str | None = None*, *country_topup: str | None = None*, *cart: List[*CartItem*] | None = None*, *product_amount: Decimal | None = None*, *vat: Decimal | None = None*, *discount_amount: Decimal | None = None*, *convenience_fee: Decimal | None = None*, *value_a: str | None = None*, *value_b: str | None = None*, *value_c: str | None = None*, *value_d: str | None = None*)

Bases: BaseModel

Dataclass for session initiation post data.

allowed_bin: str | None

cancel_url: Url

cart: List[*CartItem*] | None

classmethod check_cart_items(*v*)

check_in_time: str | None

convenience_fee: Decimal | None

country_topup: str | None

currency: str

cus_add1: str

cus_add2: str | None

cus_city: str

cus_country: str

cus_email: str

cus_fax: str | None

cus_name: str

cus_phone: str

cus_postcode: str | None

cus_state: str | None

discount_amount: Decimal | None

emi_allow_only: int | None

emi_max_inst_option: *EMIOptionsEnum* | None

emi_option: *BooleanIntEnum*

emi_selected_inst: *EMIOptionsEnum* | None

fail_url: Url

flight_type: str | None

hotel_city: str | None

hotel_name: str | None

hours_till_departure: str | None

ipn_url: str | None

journey_from_to: str | None

length_of_stay: str | None

classmethod mandatory_if_airline_tickets(*v*, *info: ValidationInfo*)

classmethod mandatory_if_telecom_vertical(*v*, *info: ValidationInfo*)

classmethod mandatory_if_travel_vertical(*v*, *info: ValidationInfo*)

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {'arbitrary_types_allowed': True}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'allowed_bin': FieldInfo(annotation=Union[str, NoneType], required=False), 'cancel_url': FieldInfo(annotation=Url, required=True, metadata=[UrlConstraints(max_length=None, allowed_schemes=['http', 'https'], host_required=None, default_host=None, default_port=None, default_path=None)]), 'cart': FieldInfo(annotation=Union[List[CartItem], NoneType], required=False), 'check_in_time': FieldInfo(annotation=Union[str, NoneType], required=False), 'convenience_fee': FieldInfo(annotation=Union[Decimal, NoneType], required=False), 'country_topup': FieldInfo(annotation=Union[str, NoneType], required=False), 'currency': FieldInfo(annotation=str, required=True), 'cus_add1': FieldInfo(annotation=str, required=True), 'cus_add2': FieldInfo(annotation=Union[str, NoneType], required=False), 'cus_city': FieldInfo(annotation=str, required=True), 'cus_country': FieldInfo(annotation=str, required=True), 'cus_email': FieldInfo(annotation=str, required=True), 'cus_fax': FieldInfo(annotation=Union[str, NoneType], required=False), 'cus_name': FieldInfo(annotation=str, required=True), 'cus_phone': FieldInfo(annotation=str, required=True), 'cus_postcode': FieldInfo(annotation=Union[str, NoneType], required=False), 'cus_state': FieldInfo(annotation=Union[str, NoneType], required=False), 'discount_amount': FieldInfo(annotation=Union[Decimal, NoneType], required=False), 'emi_allow_only': FieldInfo(annotation=Union[int, NoneType], required=False), 'emi_max_inst_option': FieldInfo(annotation=Union[EMIOptionsEnum, NoneType], required=False), 'emi_option': FieldInfo(annotation=BooleanIntEnum, required=False, default=<BooleanIntEnum.FALSE: 0>), 'emi_selected_inst': FieldInfo(annotation=Union[EMIOptionsEnum, NoneType], required=False), 'fail_url': FieldInfo(annotation=Url, required=True, metadata=[UrlConstraints(max_length=None, allowed_schemes=['http', 'https'], host_required=None, default_host=None, default_port=None, default_path=None)]), 'flight_type': FieldInfo(annotation=Union[str, NoneType], required=False), 'hotel_city': FieldInfo(annotation=Union[str, NoneType], required=False), 'hotel_name': FieldInfo(annotation=Union[str, NoneType], required=False), 'hours_till_departure': FieldInfo(annotation=Union[str, NoneType], required=False), 'ipn_url': FieldInfo(annotation=Union[str, NoneType], required=False), 'journey_from_to': FieldInfo(annotation=Union[str, NoneType], required=False), 'length_of_stay': FieldInfo(annotation=Union[str, NoneType], required=False), 'multi_card_name': FieldInfo(annotation=Union[MultiCardNamesEnum, NoneType], required=False), 'num_of_item': FieldInfo(annotation=int, required=True), 'pnr': FieldInfo(annotation=Union[str, NoneType], required=False), 'product_amount': FieldInfo(annotation=Union[Decimal, NoneType], required=False), 'product_category': FieldInfo(annotation=str, required=True), 'product_name': FieldInfo(annotation=str, required=True), 'product_profile': FieldInfo(annotation=ProductProfileEnum, required=True), 'product_type': FieldInfo(annotation=Union[str, NoneType], required=False), 'ship_add1': FieldInfo(annotation=Union[str, NoneType], required=False), 'ship_add2': FieldInfo(annotation=Union[str, NoneType], required=False), 'ship_city': FieldInfo(annotation=Union[str, NoneType], required=False), 'ship_country': FieldInfo(annotation=Union[str, NoneType], required=False), 'ship_name': FieldInfo(annotation=Union[str, NoneType], required=False), 'ship_phone': FieldInfo(annotation=Union[str, NoneType], required=False), 'ship_postcode': FieldInfo(annotation=Union[str, NoneType], required=False), 'ship_state': FieldInfo(annotation=Union[str, NoneType], required=False), 'shipping_method': FieldInfo(annotation=ShippingMethodEnum, required=False, default=<ShippingMethodEnum.YES: 'YES'>), 'success_url': FieldInfo(annotation=Url, required=True, metadata=[UrlConstraints(max_length=None, allowed_schemes=['http', 'https'], host_required=None, default_host=None, default_port=None, default_path=None)]), 'third_party_booking': FieldInfo(annotation=Union[str, NoneType], required=False), 'topup_number': FieldInfo(annotation=Union[str, NoneType], required=False), 'total_amount': FieldInfo(annotation=Decimal, required=True), 'tran_id': FieldInfo(annotation=str, required=True), 'value_a': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_b': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_c': FieldInfo(annotation=Union[str, NoneType],

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

multi_card_name: *MultiCardNamesEnum* **| None**

classmethod not_more_than_255(*v*, *info: ValidationInfo*)

classmethod not_more_than_fifty(*v*, *info: ValidationInfo*)

classmethod not_more_than_hundred(*v*, *info: ValidationInfo*)

classmethod not_more_than_hundred_fifty(*v*, *info: ValidationInfo*)

classmethod not_more_than_thirty(*v*, *info: ValidationInfo*)

classmethod not_more_than_three(*v*, *info: ValidationInfo*)

num_of_item: **int**

pnr: **str | None**

product_amount: **Decimal | None**

product_category: **str**

product_name: **str**

product_profile: *ProductProfileEnum*

product_type: **str | None**

ship_add1: **str | None**

ship_add2: **str | None**

ship_city: **str | None**

ship_country: **str | None**

ship_name: **str | None**

ship_phone: **str | None**

ship_postcode: **str | None**

ship_state: **str | None**

shipping_method: *ShippingMethodEnum*

success_url: **Url**

third_party_booking: **str | None**

topup_number: **str | None**

total_amount: **Decimal**

tran_id: **str**

classmethod valid_decimal(*v*, *info: ValidationInfo*)

classmethod valid_emi_allow_only(*v*, *info: ValidationInfo*)

classmethod validate_based_on_shipping_method(*v*, *info: ValidationInfo*)

classmethod validate_num_of_item(*v*)

value_a: str | None

value_b: str | None

value_c: str | None

value_d: str | None

vat: Decimal | None

class sslcommerz_client.dataclasses.**PaymentInitResponse**(*, *status:* ResponseStatusEnum, *failedreason: str | None = None*, *sessionkey: str | None = None*, *gw: Any | None = None*, *redirectGatewayURL: str | None = None*, *directPaymentURLBank: str | None = None*, *directPaymentURLCard: str | None = None*, *directPaymentURL: str | None = None*, *redirectGatewayURLFailed: str | None = None*, *GatewayPageURL: str | None = None*, *storeBanner: str | None = None*, *storeLogo: str | None = None*, *desc: List[*Gateway*] | None = None*)

Bases: BaseModel

Payment initiation response as a dataclass.

GatewayPageURL: str | None

desc: List[*Gateway*] | None

directPaymentURL: str | None

directPaymentURLBank: str | None

directPaymentURLCard: str | None

failedreason: str | None

gw: Any | None

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {'arbitrary_types_allowed': True}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'GatewayPageURL': FieldInfo(annotation=Union[str, NoneType], required=False), 'desc': FieldInfo(annotation=Union[List[Gateway], NoneType], required=False), 'directPaymentURL': FieldInfo(annotation=Union[str, NoneType], required=False), 'directPaymentURLBank': FieldInfo(annotation=Union[str, NoneType], required=False), 'directPaymentURLCard': FieldInfo(annotation=Union[str, NoneType], required=False), 'failedreason': FieldInfo(annotation=Union[str, NoneType], required=False), 'gw': FieldInfo(annotation=Union[Any, NoneType], required=False), 'redirectGatewayURL': FieldInfo(annotation=Union[str, NoneType], required=False), 'redirectGatewayURLFailed': FieldInfo(annotation=Union[str, NoneType], required=False), 'sessionkey': FieldInfo(annotation=Union[str, NoneType], required=False), 'status': FieldInfo(annotation=ResponseStatusEnum, required=True), 'storeBanner': FieldInfo(annotation=Union[str, NoneType], required=False), 'storeLogo': FieldInfo(annotation=Union[str, NoneType], required=False)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

redirectGatewayURL: str | None

redirectGatewayURLFailed: str | None

sessionkey: str | None

status: *ResponseStatusEnum*

storeBanner: str | None

storeLogo: str | None

class sslcommerz_client.dataclasses.**ProductProfileEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: str, Enum

AIRLINE_TICKETS = 'airline-tickets'

GENERAL = 'general'

NON_PHYSICAL_GOODS = 'non-physical-goods'

PHYSICAL_GOODS = 'physical-goods'

TELECOM_VERTICAL = 'telecom-vertical'

TRAVEL_VERTICAL = 'travel-vertical'

class sslcommerz_client.dataclasses.**RefundInitiateResponse**(*, *APIConnect: APIConnectEnum*, *bank_tran_id: str*, *trans_id: str | None = None*, *refund_ref_id: str | None = None*, *status: RefundStatusEnum*, *errorReason: str | None = None*)

Bases: BaseModel

Refund initiation response.

APIConnect: *APIConnectEnum*

bank_tran_id: str

errorReason: str | None

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'APIConnect': FieldInfo(annotation=APIConnectEnum, required=True), 'bank_tran_id': FieldInfo(annotation=str, required=True), 'errorReason': FieldInfo(annotation=Union[str, NoneType], required=False), 'refund_ref_id': FieldInfo(annotation=Union[str, NoneType], required=False), 'status': FieldInfo(annotation=RefundStatusEnum, required=True), 'trans_id': FieldInfo(annotation=Union[str, NoneType], required=False)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

refund_ref_id: str | None

status: *RefundStatusEnum*

trans_id: str | None

class sslcommerz_client.dataclasses.**RefundRequestPostData**(*, *bank_tran_id: str*, *refund_amount: str*, *refund_remarks: str*, *refe_id: str*)

Bases: BaseModel

Dataclass for Refund API post data.

bank_tran_id: str

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'bank_tran_id': FieldInfo(annotation=str, required=True), 'refe_id': FieldInfo(annotation=str, required=True), 'refund_amount': FieldInfo(annotation=str, required=True), 'refund_remarks': FieldInfo(annotation=str, required=True)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

classmethod not_more_than_255(*v*, *info: ValidationInfo*)

classmethod not_more_than_eighty(*v*, *info: ValidationInfo*)

classmethod not_more_than_fifty(*v*, *info: ValidationInfo*)

refe_id: str

refund_amount: str

refund_remarks: str

classmethod valid_decimal(*v*, *info: ValidationInfo*)

class sslcommerz_client.dataclasses.**RefundResponse**(*, *APIConnect: APIConnectEnum*, *bank_tran_id: str*, *trans_id: str | None = None*, *refund_ref_id: str | None = None*, *status: RefundStatusEnum*, *errorReason: str | None = None*, *initiated_on: datetime*, *refunded_on: datetime*)

Bases: *RefundInitiateResponse*

Refund response.

initiated_on: datetime

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'APIConnect': FieldInfo(annotation=APIConnectEnum, required=True), 'bank_tran_id': FieldInfo(annotation=str, required=True), 'errorReason': FieldInfo(annotation=Union[str, NoneType], required=False), 'initiated_on': FieldInfo(annotation=datetime, required=True), 'refund_ref_id': FieldInfo(annotation=Union[str, NoneType], required=False), 'refunded_on': FieldInfo(annotation=datetime, required=True), 'status': FieldInfo(annotation=RefundStatusEnum, required=True), 'trans_id': FieldInfo(annotation=Union[str, NoneType], required=False)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

refunded_on: datetime

class sslcommerz_client.dataclasses.**RefundStatusEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: `str`, `Enum`

FAILED = 'failed'

PROCESSING = 'processing'

SUCCESS = 'success'

class sslcommerz_client.dataclasses.**ResponseStatusEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: `str`, `Enum`

FAILED = 'FAILED'

SUCCESS = 'SUCCESS'

class sslcommerz_client.dataclasses.**RiskLevelEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: str, Enum

HIGH = '1'

LOW = '0'

class sslcommerz_client.dataclasses.**Session**(***, *status: str*, *tran_date: datetime*, *tran_id: str*, *val_id: str*, *amount: Decimal*, *store_amount: Decimal*, *card_type: str*, *card_no: str*, *bank_tran_id: str*, *card_issuer: str*, *card_brand: str*, *card_issuer_country: str*, *card_issuer_country_code: str*, *currency_type: str*, *currency_amount: Decimal*, *risk_level:* RiskLevelEnum, *risk_title: str*, *sessionkey: str | None = None*, *error: str | None = None*, *currency: str | None = None*, *emi_instalment: Decimal | str | None = None*, *emi_amount: Decimal | str | None = None*, *discount_percentage: Decimal | str | None = None*, *discount_remarks: str | None = None*, *value_a: str | None = None*, *value_b: str | None = None*, *value_c: str | None = None*, *value_d: str | None = None*)

Bases: BaseModel

Dataclass for transaction session.

amount: Decimal

bank_tran_id: str

card_brand: str

card_issuer: str

card_issuer_country: str

card_issuer_country_code: str

card_no: str

card_type: str

currency: str | None

currency_amount: Decimal

currency_type: str

discount_percentage: Decimal | str | None

discount_remarks: str | None

emi_amount: Decimal | str | None

emi_instalment: Decimal | str | None

error: str | None

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'amount': FieldInfo(annotation=Decimal, required=True), 'bank_tran_id': FieldInfo(annotation=str, required=True), 'card_brand': FieldInfo(annotation=str, required=True), 'card_issuer': FieldInfo(annotation=str, required=True), 'card_issuer_country': FieldInfo(annotation=str, required=True), 'card_issuer_country_code': FieldInfo(annotation=str, required=True), 'card_no': FieldInfo(annotation=str, required=True), 'card_type': FieldInfo(annotation=str, required=True), 'currency': FieldInfo(annotation=Union[str, NoneType], required=False), 'currency_amount': FieldInfo(annotation=Decimal, required=True), 'currency_type': FieldInfo(annotation=str, required=True), 'discount_percentage': FieldInfo(annotation=Union[Decimal, str, NoneType], required=False), 'discount_remarks': FieldInfo(annotation=Union[str, NoneType], required=False), 'emi_amount': FieldInfo(annotation=Union[Decimal, str, NoneType], required=False), 'emi_instalment': FieldInfo(annotation=Union[Decimal, str, NoneType], required=False), 'error': FieldInfo(annotation=Union[str, NoneType], required=False), 'risk_level': FieldInfo(annotation=RiskLevelEnum, required=True), 'risk_title': FieldInfo(annotation=str, required=True), 'sessionkey': FieldInfo(annotation=Union[str, NoneType], required=False), 'status': FieldInfo(annotation=str, required=True), 'store_amount': FieldInfo(annotation=Decimal, required=True), 'tran_date': FieldInfo(annotation=datetime, required=True), 'tran_id': FieldInfo(annotation=str, required=True), 'val_id': FieldInfo(annotation=str, required=True), 'value_a': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_b': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_c': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_d': FieldInfo(annotation=Union[str, NoneType], required=False)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

risk_level: *RiskLevelEnum*

risk_title: str

sessionkey: str | None

status: str

store_amount: Decimal

tran_date: datetime

tran_id: str

val_id: str

value_a: str | None

value_b: str | None

value_c: str | None

value_d: str | None

class sslcommerz_client.dataclasses.**ShippingMethodEnum**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: str, Enum

COURIER = 'Courier'

NO = 'NO'

YES = 'YES'

class sslcommerz_client.dataclasses.**TransactionBySessionResponse**(***, *status: str*, *tran_date: datetime*, *tran_id: str*, *val_id: str*, *amount: Decimal*, *store_amount: Decimal*, *card_type: str*, *card_no: str*, *bank_tran_id: str*, *card_issuer: str*, *card_brand: str*, *card_issuer_country: str*, *card_issuer_country_code: str*, *currency_type: str*, *currency_amount: Decimal*, *risk_level:* RiskLevelEnum, *risk_title: str*, *sessionkey: str | None = None*, *error: str | None = None*, *currency: str | None = None*, *emi_instalment: Decimal | str | None = None*, *emi_amount: Decimal | str | None = None*, *discount_percentage: Decimal | str | None = None*, *discount_remarks: str | None = None*, *value_a: str | None = None*, *value_b: str | None = None*, *value_c: str | None = None*, *value_d: str | None = None*, *APIConnect:* APIConnectEnum)

Bases: *Session*

Dataclass for transaction by session query.

APIConnect: *APIConnectEnum*

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'APIConnect': FieldInfo(annotation=APIConnectEnum, required=True), 'amount': FieldInfo(annotation=Decimal, required=True), 'bank_tran_id': FieldInfo(annotation=str, required=True), 'card_brand': FieldInfo(annotation=str, required=True), 'card_issuer': FieldInfo(annotation=str, required=True), 'card_issuer_country': FieldInfo(annotation=str, required=True), 'card_issuer_country_code': FieldInfo(annotation=str, required=True), 'card_no': FieldInfo(annotation=str, required=True), 'card_type': FieldInfo(annotation=str, required=True), 'currency': FieldInfo(annotation=Union[str, NoneType], required=False), 'currency_amount': FieldInfo(annotation=Decimal, required=True), 'currency_type': FieldInfo(annotation=str, required=True), 'discount_percentage': FieldInfo(annotation=Union[Decimal, str, NoneType], required=False), 'discount_remarks': FieldInfo(annotation=Union[str, NoneType], required=False), 'emi_amount': FieldInfo(annotation=Union[Decimal, str, NoneType], required=False), 'emi_instalment': FieldInfo(annotation=Union[Decimal, str, NoneType], required=False), 'error': FieldInfo(annotation=Union[str, NoneType], required=False), 'risk_level': FieldInfo(annotation=RiskLevelEnum, required=True), 'risk_title': FieldInfo(annotation=str, required=True), 'sessionkey': FieldInfo(annotation=Union[str, NoneType], required=False), 'status': FieldInfo(annotation=str, required=True), 'store_amount': FieldInfo(annotation=Decimal, required=True), 'tran_date': FieldInfo(annotation=datetime, required=True), 'tran_id': FieldInfo(annotation=str, required=True), 'val_id': FieldInfo(annotation=str, required=True), 'value_a': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_b': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_c': FieldInfo(annotation=Union[str, NoneType], required=False), 'value_d': FieldInfo(annotation=Union[str, NoneType], required=False)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

class sslcommerz_client.dataclasses.**TransactionsByIDResponse**(*, *APIConnect: APIConnectEnum*, *no_of_trans_found: int*, *element: List[Session]*)

Bases: `BaseModel`

Dataclass for transactions by ID query.

APIConnect: *APIConnectEnum*

element: List[*Session*]

model_computed_fields: ClassVar[dict[str, ComputedFieldInfo]] = {}

A dictionary of computed field names and their corresponding *ComputedFieldInfo* objects.

model_config: ClassVar[ConfigDict] = {}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

model_fields: ClassVar[dict[str, FieldInfo]] = {'APIConnect': FieldInfo(annotation=APIConnectEnum, required=True), 'element': FieldInfo(annotation=List[Session], required=True), 'no_of_trans_found': FieldInfo(annotation=int, required=True)}

Metadata about the fields defined on the model, mapping of field names to [*FieldInfo*][pydantic.fields.FieldInfo].

This replaces *Model.__fields__* from Pydantic V1.

no_of_trans_found: int

3.1.4 Module contents

CHAPTER

FOUR

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

4.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.2 Documentation improvements

SSLCommerz Client could always use more documentation, whether as part of the official SSLCommerz Client docs, in docstrings, or even on the web in blog posts, articles, and such.

4.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://gitlab.com/codesigntheory/python-sslcommerz-client/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

4.4 Development

To set up *python-sslcommerz-client* for local development:

1. Fork python-sslcommerz-client (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@gitlab.com:codesigntheory/python-sslcommerz-client.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

4.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`).
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

4.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel*:

```
tox -p auto
```

CHAPTER

FIVE

AUTHORS

- Utsob Roy - https://co.design
- Nayan Biswas - https://co.design

CHAPTER

SIX

CHANGELOG

6.1 0.4.0 (2021-08-05)

- Fixed Refund Request

6.2 0.3.4 (2021-05-08)

- Removed CardBrandEnum

6.3 0.3.4 (2021-05-08)

- Fixed Card Brand Enum

6.4 0.3.2 (2021-04-19)

- Added more test.
- Fixed some bugs.

6.5 0.3.1 (2021-04-19)

- Added more test.
- Fixed some bugs.
- Fully working IPN validation, order query.

6.6 0.2.1 (2021-04-18)

- Added None in card brand
- Fixed some bugs.

6.7 0.2.0 (2021-04-18)

- Added a test.
- Fixed some bugs.

6.8 0.1.0 (2021-04-18)

- Added all API methods.
- Updated Docs.

6.9 0.0.2 (2021-04-16)

- Removed a Classifier

6.10 0.0.1 (2021-04-16)

- DataClasses for PaymentInit and APN request and responses.
- Functional Client only with session initiation.

6.11 0.0.0 (2021-04-16)

- First release on PyPI.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

Y