

---

# SSLCommerz Client

*Release 0.4.0*

**Utsob Roy**

**Mar 19, 2024**



# CONTENTS

<b>1</b>	<b>Overview</b>	<b>1</b>
1.1	Warning	1
1.2	Why?	1
1.3	Features	1
1.4	Installation	1
1.5	Documentation	2
1.6	Development	2
<b>2</b>	<b>Installation and Usage</b>	<b>3</b>
2.1	Installation	3
2.2	Usage	3
<b>3</b>	<b>Reference</b>	<b>7</b>
3.1	sslcommerz_client package	7
<b>4</b>	<b>Contributing</b>	<b>33</b>
4.1	Bug reports	33
4.2	Documentation improvements	33
4.3	Feature requests and feedback	33
4.4	Development	34
<b>5</b>	<b>Authors</b>	<b>35</b>
<b>6</b>	<b>Changelog</b>	<b>37</b>
6.1	0.4.0 (2021-08-05)	37
6.2	0.3.4 (2021-05-08)	37
6.3	0.3.4 (2021-05-08)	37
6.4	0.3.2 (2021-04-19)	37
6.5	0.3.1 (2021-04-19)	37
6.6	0.2.1 (2021-04-18)	38
6.7	0.2.0 (2021-04-18)	38
6.8	0.1.0 (2021-04-18)	38
6.9	0.0.2 (2021-04-16)	38
6.10	0.0.1 (2021-04-16)	38
6.11	0.0.0 (2021-04-16)	38
<b>7</b>	<b>Indices and tables</b>	<b>39</b>
	<b>Python Module Index</b>	<b>41</b>
	<b>Index</b>	<b>43</b>



## 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:

Win-  
dows

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

Other

```
PYTEST_ADDOPTS=--cov-append tox
```

## 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*.



## 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 [Config-
    Dict][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 [Field-
    Info][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

## 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
```

## **AUTHORS**

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





## 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.

## INDICES AND TABLES

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



## PYTHON MODULE INDEX

### S

`sslcommerz_client`, [32](#)  
`sslcommerz_client.client`, [7](#)  
`sslcommerz_client.dataclasses`, [8](#)



## INDEX

### A

**ABBANK** (*sslcommerz\_client.dataclasses.MultiCardNamesEnum* attribute), 15  
**AIRLINE\_TICKETS** (*sslcommerz\_client.dataclasses.ProductProfileEnum* attribute), 25  
**allowed\_bin** (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 19  
**AMEX\_CARD** (*sslcommerz\_client.dataclasses.MultiCardNamesEnum* attribute), 15  
**amount** (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 9  
**amount** (*sslcommerz\_client.dataclasses.CartItem* attribute), 11  
**amount** (*sslcommerz\_client.dataclasses.Session* attribute), 28  
**APIConnect** (*sslcommerz\_client.dataclasses.RefundInitiateResponse* attribute), 25  
**APIConnect** (*sslcommerz\_client.dataclasses.TransactionBySessionResponse* attribute), 30  
**APIConnect** (*sslcommerz\_client.dataclasses.TransactionsBySessionResponse* attribute), 31  
**APIConnectEnum** (class in *sslcommerz\_client.dataclasses*), 8  
**APIResponse** (class in *sslcommerz\_client.dataclasses*), 8  
**base\_fair** (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 9  
**BaseOrderResponse** (class in *sslcommerz\_client.dataclasses*), 8  
**baseURL** (*sslcommerz\_client.client.SSLCommerzClient* property), 7  
**BKASH** (*sslcommerz\_client.dataclasses.MultiCardNamesEnum* attribute), 15  
**BooleanIntEnum** (class in *sslcommerz\_client.dataclasses*), 10  
**BRAC\_MASTER** (*sslcommerz\_client.dataclasses.MultiCardNamesEnum* attribute), 15  
**BRAC\_VISA** (*sslcommerz\_client.dataclasses.MultiCardNamesEnum* attribute), 15

### C

**cancel\_url** (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 19  
**CANCELLED** (*sslcommerz\_client.dataclasses.IPNOrderStatusEnum* attribute), 12  
**card\_brand** (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 9  
**card\_brand** (*sslcommerz\_client.dataclasses.Session* attribute), 28  
**card\_issuer** (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 9  
**card\_issuer** (*sslcommerz\_client.dataclasses.Session* attribute), 28  
**card\_issuer\_country** (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 9  
**card\_issuer\_country** (*sslcommerz\_client.dataclasses.Session* attribute), 28  
**card\_issuer\_country\_code** (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 9  
**card\_issuer\_country\_code** (*sslcommerz\_client.dataclasses.Session* attribute), 28

### B

**BANK\_ASIA** (*sslcommerz\_client.dataclasses.MultiCardNamesEnum* attribute), 15  
**bank\_tran\_id** (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 9  
**bank\_tran\_id** (*sslcommerz\_client.dataclasses.RefundInitiateResponse* attribute), 25  
**bank\_tran\_id** (*sslcommerz\_client.dataclasses.RefundRequestPostData* attribute), 26  
**bank\_tran\_id** (*sslcommerz\_client.dataclasses.Session* attribute), 28

card\_no (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 9  
card\_no (sslcommerz\_client.dataclasses.Session attribute), 28  
card\_sub\_brand (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 9  
card\_type (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 9  
card\_type (sslcommerz\_client.dataclasses.Session attribute), 28  
cart (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 19  
CartItem (class in sslcommerz\_client.dataclasses), 10  
check\_cart\_items() (sslcommerz\_client.dataclasses.PaymentInitPostData class method), 20  
check\_in\_time (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
CITY (sslcommerz\_client.dataclasses.MultiCardNamesEnum attribute), 15  
CITY\_AMEX (sslcommerz\_client.dataclasses.MultiCardNamesEnum attribute), 15  
CITY\_MASTER (sslcommerz\_client.dataclasses.MultiCardNamesEnum attribute), 15  
CITY\_VISA (sslcommerz\_client.dataclasses.MultiCardNamesEnum attribute), 15  
convenience\_fee (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
country\_topup (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
COURIER (sslcommerz\_client.dataclasses.ShippingMethodEnum attribute), 30  
Credential (class in sslcommerz\_client.dataclasses), 11  
credential (sslcommerz\_client.client.SSLCommerzClient property), 7  
currency (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 9  
currency (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
currency (sslcommerz\_client.dataclasses.Session attribute), 28  
currency\_amount (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 9  
currency\_amount (sslcommerz\_client.dataclasses.Session attribute), 28  
currency\_rate (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 9  
currency\_type (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 9  
currency\_type (sslcommerz\_client.dataclasses.Session attribute), 28  
cus\_add1 (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
cus\_add2 (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
cus\_city (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
cus\_country (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
cus\_email (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
cus\_fax (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
cus\_name (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
cus\_phone (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
cus\_postcode (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
cus\_state (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 20  
**D**  
DBBL\_MASTER (sslcommerz\_client.dataclasses.MultiCardNamesEnum attribute), 15  
DBBL\_MOBILE\_BANKING (sslcommerz\_client.dataclasses.MultiCardNamesEnum attribute), 15  
DBBL\_NEXUS (sslcommerz\_client.dataclasses.MultiCardNamesEnum attribute), 15  
DBBL\_VISA (sslcommerz\_client.dataclasses.MultiCardNamesEnum attribute), 15  
desc (sslcommerz\_client.dataclasses.PaymentInitResponse attribute), 24  
directPaymentURL (sslcommerz\_client.dataclasses.PaymentInitResponse attribute), 24  
directPaymentURLBank (sslcommerz\_client.dataclasses.PaymentInitResponse attribute), 24  
directPaymentURLCard (sslcommerz\_client.dataclasses.PaymentInitResponse attribute), 24  
discount\_amount (sslcommerz\_client.dataclasses.OrderValidationResponse attribute), 17



discount_amount	(sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20	EXPIRED (sslcommerz_client.dataclasses.IPNOrderStatusEnum attribute), 12
discount_percentage	(sslcommerz_client.dataclasses.OrderValidationResponse attribute), 17	
discount_percentage	(sslcommerz_client.dataclasses.Session attribute), 28	fail_url (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20
discount_remarks	(sslcommerz_client.dataclasses.OrderValidationResponse attribute), 17	FAILED (sslcommerz_client.dataclasses.APICConnectEnum attribute), 8
discount_remarks	(sslcommerz_client.dataclasses.Session attribute), 28	FAILED (sslcommerz_client.dataclasses.IPNOrderStatusEnum attribute), 13
		FAILED (sslcommerz_client.dataclasses.RefundStatusEnum attribute), 27
		FAILED (sslcommerz_client.dataclasses.ResponseStatusEnum attribute), 27
DONE	(sslcommerz_client.dataclasses.APICConnectEnum attribute), 8	failedreason (sslcommerz_client.dataclasses.PaymentInitResponse attribute), 24
<b>E</b>		
EBL_MASTER	(sslcommerz_client.dataclasses.MultiCardNamesEnum attribute), 15	FALSE (sslcommerz_client.dataclasses.BooleanIntEnum attribute), 10
EBL_VISA	(sslcommerz_client.dataclasses.MultiCardNamesEnum attribute), 15	flight_type (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20
element	(sslcommerz_client.dataclasses.TransactionsByIDResponse attribute), 31	
<b>G</b>		
emi_allow_only	(sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20	Gateway (class in sslcommerz_client.dataclasses), 12
emi_amount	(sslcommerz_client.dataclasses.Session attribute), 28	GatewayPageURL (sslcommerz_client.dataclasses.PaymentInitResponse attribute), 24
emi_instalment	(sslcommerz_client.dataclasses.OrderValidationResponse attribute), 17	GENERAL (sslcommerz_client.dataclasses.ProductProfileEnum attribute), 25
emi_instalment	(sslcommerz_client.dataclasses.Session attribute), 28	get_hash() (sslcommerz_client.dataclasses.IPNResponse method), 13
emi_max_inst_option	(sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20	get_order_validation_data() (sslcommerz_client.client.SSLCommerzClient method), 7
emi_option	(sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20	get_refund_data() (sslcommerz_client.client.SSLCommerzClient method), 7
emi_selected_inst	(sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20	get_transaction_by_id() (sslcommerz_client.client.SSLCommerzClient method), 7
EMIOptionsEnum	(class in sslcommerz_client.dataclasses), 11	get_transaction_by_session() (sslcommerz_client.client.SSLCommerzClient method), 7
EMIOptionsResponseEnum	(class in sslcommerz_client.dataclasses), 12	gw (sslcommerz_client.dataclasses.Gateway attribute), 12
error	(sslcommerz_client.dataclasses.BaseOrderResponse attribute), 9	gw (sslcommerz_client.dataclasses.PaymentInitResponse attribute), 24
<b>H</b>		
error	(sslcommerz_client.dataclasses.Session attribute), 28	HIGH (sslcommerz_client.dataclasses.RiskLevelEnum attribute), 28
errorReason	(sslcommerz_client.dataclasses.RefundInitiateResponse attribute), 20	hotel_city (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20

hotel_name (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20	merz_client.dataclasses.PaymentInitPostData class method), 20
hours_till_departure (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20	mandatory_if_telecom_vertical() (sslcommerz_client.dataclasses.PaymentInitPostData class method), 20
I	mandatory_if_travel_vertical() (sslcommerz_client.dataclasses.PaymentInitPostData class method), 20
IBBL (sslcommerz_client.dataclasses.MultiCardNamesEnum attribute), 15	MASTER_CARD (sslcommerz_client.dataclasses.MultiCardNamesEnum attribute), 15
INACTIVE (sslcommerz_client.dataclasses.APICConnectEnum attribute), 8	MOBILE_BANK (sslcommerz_client.dataclasses.MultiCardNamesEnum attribute), 15
initiate_refund() (sslcommerz_client.client.SSLCommerzClient method), 7	model_computed_fields (sslcommerz_client.dataclasses.APIResponse attribute), 8
initiate_session() (sslcommerz_client.client.SSLCommerzClient method), 7	model_computed_fields (sslcommerz_client.dataclasses.BaseOrderResponse attribute), 9
initiated_on (sslcommerz_client.dataclasses.RefundResponse attribute), 27	model_computed_fields (sslcommerz_client.dataclasses.CartItem attribute), 11
INTERNET_BANK (sslcommerz_client.dataclasses.MultiCardNamesEnum attribute), 15	model_computed_fields (sslcommerz_client.dataclasses.Credential attribute), 11
INVALID_REQUEST (sslcommerz_client.dataclasses.APICConnectEnum attribute), 8	model_computed_fields (sslcommerz_client.dataclasses.Gateway attribute), 12
INVALID_TRANSACTION (sslcommerz_client.dataclasses.OrderStatusEnum attribute), 16	model_computed_fields (sslcommerz_client.dataclasses.IPNResponse attribute), 13
ipn_url (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20	model_computed_fields (sslcommerz_client.dataclasses.IPNValidationStatus attribute), 14
IPNOrderStatusEnum (class in sslcommerz_client.dataclasses), 12	model_computed_fields (sslcommerz_client.dataclasses.OrderValidationPostData attribute), 16
IPNResponse (class in sslcommerz_client.dataclasses), 13	model_computed_fields (sslcommerz_client.dataclasses.OrderValidationResponse attribute), 17
IPNValidationStatus (class in sslcommerz_client.dataclasses), 14	model_computed_fields (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 21
J	model_computed_fields (sslcommerz_client.dataclasses.PaymentInitResponse attribute), 24
journey_from_to (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20	model_computed_fields (sslcommerz_client.dataclasses.RefundInitiateResponse attribute), 26
L	model_computed_fields (sslcommerz_client.dataclasses.RefundRequestPostData attribute), 26
length_of_stay (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 20	model_computed_fields (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 21
logo (sslcommerz_client.dataclasses.Gateway attribute), 12	model_computed_fields (sslcommerz_client.dataclasses.PaymentInitResponse attribute), 24
LOW (sslcommerz_client.dataclasses.RiskLevelEnum attribute), 28	model_computed_fields (sslcommerz_client.dataclasses.RefundInitiateResponse attribute), 26
M	model_computed_fields (sslcommerz_client.dataclasses.RefundRequestPostData attribute), 26
mandatory_if_airline_tickets() (sslcommerz_client.client.SSLCommerzClient method), 7	model_computed_fields (sslcommerz_client.dataclasses.PaymentInitPostData attribute), 21

<i>merz_client.dataclasses.RefundResponse</i> attribute), 27		<i>merz_client.dataclasses.TransactionBySessionResponse</i> attribute), 30	
model_computed_fields <i>merz_client.dataclasses.Session</i> 29	(sslcom- attribute),	model_config <i>merz_client.dataclasses.TransactionsByIDResponse</i> attribute), 31	(sslcom- attribute),
model_computed_fields <i>merz_client.dataclasses.TransactionBySessionResponse</i> attribute), 30	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.APIResponse</i> attribute), 8	(sslcom- attribute),
model_computed_fields <i>merz_client.dataclasses.TransactionsByIDResponse</i> attribute), 31	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.BaseOrderResponse</i> attribute), 9	(sslcom- attribute),
model_config <i>merz_client.dataclasses.APIResponse</i> attribute), 8	(sslcom- attribute),	model_fields ( <i>sslcommerz_client.dataclasses.CartItem</i> attribute), 11	(sslcom- attribute),
model_config <i>merz_client.dataclasses.BaseOrderResponse</i> attribute), 9	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.Credential</i> 11	(sslcom- attribute),
model_config ( <i>sslcommerz_client.dataclasses.CartItem</i> attribute), 11	(sslcom- attribute),	model_fields ( <i>sslcommerz_client.dataclasses.Gateway</i> attribute), 12	(sslcom- attribute),
model_config <i>merz_client.dataclasses.Credential</i> 11	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.IPNResponse</i> attribute), 13	(sslcom- attribute),
model_config ( <i>sslcommerz_client.dataclasses.Gateway</i> attribute), 12	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.IPNValidationStatus</i> attribute), 14	(sslcom- attribute),
model_config <i>merz_client.dataclasses.IPNResponse</i> attribute), 13	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.OrderValidationPostData</i> attribute), 16	(sslcom- attribute),
model_config <i>merz_client.dataclasses.IPNValidationStatus</i> attribute), 14	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.OrderValidationResponse</i> attribute), 17	(sslcom- attribute),
model_config <i>merz_client.dataclasses.OrderValidationPostData</i> attribute), 16	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.PaymentInitPostData</i> attribute), 21	(sslcom- attribute),
model_config <i>merz_client.dataclasses.OrderValidationResponse</i> attribute), 17	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.PaymentInitResponse</i> attribute), 24	(sslcom- attribute),
model_config <i>merz_client.dataclasses.PaymentInitPostData</i> attribute), 21	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.RefundInitiateResponse</i> attribute), 26	(sslcom- attribute),
model_config <i>merz_client.dataclasses.PaymentInitResponse</i> attribute), 24	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.RefundRequestPostData</i> attribute), 26	(sslcom- attribute),
model_config <i>merz_client.dataclasses.RefundInitiateResponse</i> attribute), 26	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.RefundResponse</i> attribute), 27	(sslcom- attribute),
model_config <i>merz_client.dataclasses.RefundRequestPostData</i> attribute), 26	(sslcom- attribute),	model_fields ( <i>sslcommerz_client.dataclasses.Session</i> attribute), 29	(sslcom- attribute),
model_config <i>merz_client.dataclasses.RefundResponse</i> attribute), 27	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.TransactionBySessionResponse</i> attribute), 31	(sslcom- attribute),
model_config ( <i>sslcommerz_client.dataclasses.Session</i> attribute), 29	(sslcom- attribute),	model_fields <i>merz_client.dataclasses.TransactionsByIDResponse</i> attribute), 31	(sslcom- attribute),
model_config	(sslcom- attribute),	module	

[sslcommerz\\_client](#), 32  
[sslcommerz\\_client.client](#), 7  
[sslcommerz\\_client.dataclasses](#), 8  
MTBL ([sslcommerz\\_client.dataclasses.MultiCardNamesEnum](#) attribute), 15  
multi\_card\_name ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) attribute), 23  
MultiCardNamesEnum (class in [sslcommerz\\_client.dataclasses](#)), 15

## N

name ([sslcommerz\\_client.dataclasses.Gateway](#) attribute), 12  
NINE\_MONTHS ([sslcommerz\\_client.dataclasses.EMIOptionsEnum](#) attribute), 11  
NINE\_MONTHS ([sslcommerz\\_client.dataclasses.EMIOptionsResponseEnum](#) attribute), 12  
NO ([sslcommerz\\_client.dataclasses.ShippingMethodEnum](#) attribute), 30  
no\_of\_trans\_found ([sslcommerz\\_client.dataclasses.TransactionsByIDResponse](#) attribute), 32  
NON\_PHYSICAL\_GOODS ([sslcommerz\\_client.dataclasses.ProductProfileEnum](#) attribute), 25  
NONE ([sslcommerz\\_client.dataclasses.EMIOptionsResponseEnum](#) attribute), 12  
not\_more\_than\_255() ([sslcommerz\\_client.dataclasses.CartItem](#) class method), 11  
not\_more\_than\_255() ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) class method), 23  
not\_more\_than\_255() ([sslcommerz\\_client.dataclasses.RefundRequestPostData](#) class method), 26  
not\_more\_than\_eighty() ([sslcommerz\\_client.dataclasses.RefundRequestPostData](#) class method), 26  
not\_more\_than\_fifty() ([sslcommerz\\_client.dataclasses.OrderValidationPostData](#) class method), 16  
not\_more\_than\_fifty() ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) class method), 23  
not\_more\_than\_fifty() ([sslcommerz\\_client.dataclasses.RefundRequestPostData](#) class method), 26  
not\_more\_than\_hundred() ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) class method), 23

not\_more\_than\_hundred\_fifty() ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) class method), 23  
not\_more\_than\_thirty() ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) class method), 23  
not\_more\_than\_three() ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) class method), 23  
num\_of\_item ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) attribute), 23

## O

OrderStatusEnum (class in [sslcommerz\\_client.dataclasses](#)), 16  
OrderValidationPostData (class in [sslcommerz\\_client.dataclasses](#)), 16  
OrderValidationResponse (class in [sslcommerz\\_client.dataclasses](#)), 16  
OTHER\_CARD ([sslcommerz\\_client.dataclasses.MultiCardNamesEnum](#) attribute), 15

## P

PaymentInitPostData (class in [sslcommerz\\_client.dataclasses](#)), 18  
PaymentInitResponse (class in [sslcommerz\\_client.dataclasses](#)), 24  
PHYSICAL\_GOODS ([sslcommerz\\_client.dataclasses.ProductProfileEnum](#) attribute), 25  
pnr ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) attribute), 23  
PROCESSING ([sslcommerz\\_client.dataclasses.RefundStatusEnum](#) attribute), 27  
product ([sslcommerz\\_client.dataclasses.CartItem](#) attribute), 11  
product\_amount ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) attribute), 23  
product\_category ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) attribute), 23  
product\_name ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) attribute), 23  
product\_profile ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) attribute), 23  
product\_type ([sslcommerz\\_client.dataclasses.PaymentInitPostData](#) attribute), 23  
ProductProfileEnum (class in [sslcommerz\\_client.dataclasses](#)), 25

## Q

QCASH(*sslcommerz\_client.dataclasses.MultiCardNamesEnum* attribute), 15

quantity (*sslcommerz\_client.dataclasses.CartItem* attribute), 11

## R

r\_flag (*sslcommerz\_client.dataclasses.Gateway* attribute), 12

raw\_data (*sslcommerz\_client.dataclasses.APIResponse* attribute), 8

redirectGatewayURL (*sslcommerz\_client.dataclasses.Gateway* attribute), 12

redirectGatewayURL (*sslcommerz\_client.dataclasses.PaymentInitResponse* attribute), 25

redirectGatewayURLFailed (*sslcommerz\_client.dataclasses.PaymentInitResponse* attribute), 25

refe\_id(*sslcommerz\_client.dataclasses.RefundRequestPostData* attribute), 26

refund\_amount (*sslcommerz\_client.dataclasses.RefundRequestPostData* attribute), 27

refund\_ref\_id (*sslcommerz\_client.dataclasses.RefundInitiateResponse* attribute), 26

refund\_remarks (*sslcommerz\_client.dataclasses.RefundRequestPostData* attribute), 27

refunded\_on (*sslcommerz\_client.dataclasses.RefundResponse* attribute), 27

RefundInitiateResponse (class in *sslcommerz\_client.dataclasses*), 25

RefundRequestPostData (class in *sslcommerz\_client.dataclasses*), 26

RefundResponse (class in *sslcommerz\_client.dataclasses*), 27

RefundStatusEnum (class in *sslcommerz\_client.dataclasses*), 27

response (*sslcommerz\_client.dataclasses.APIResponse* attribute), 8

response (*sslcommerz\_client.dataclasses.IPNValidationStatus* attribute), 15

ResponseStatusEnum (class in *sslcommerz\_client.dataclasses*), 27

risk\_level(*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 10

risk\_level (*sslcommerz\_client.dataclasses.Session* attribute), 29

risk\_title(*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 10

risk\_title (*sslcommerz\_client.dataclasses.Session* attribute), 29

RiskLevelEnum (class in *sslcommerz\_client.dataclasses*), 28

## S

SBL\_MASTER(*sslcommerz\_client.dataclasses.MultiCardNamesEnum* attribute), 16

SBL\_VISA(*sslcommerz\_client.dataclasses.MultiCardNamesEnum* attribute), 16

Session (class in *sslcommerz\_client.dataclasses*), 28

sessionkey (*sslcommerz\_client.dataclasses.PaymentInitResponse* attribute), 25

sessionkey (*sslcommerz\_client.dataclasses.Session* attribute), 29

ship\_add1 (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 23

ship\_add2 (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 23

ship\_city (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 23

ship\_country (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 23

ship\_name (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 23

ship\_phone (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 23

ship\_postcode (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 23

ship\_state (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 23

shipping\_method (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 23

ShippingMethodEnum (class in *sslcommerz\_client.dataclasses*), 30

SIX\_MONTHS (*sslcommerz\_client.dataclasses.EMIOptionsEnum* attribute), 11

SIX\_MONTHS (*sslcommerz\_client.dataclasses.EMIOptionsResponseEnum* attribute), 12

sslcommerz\_client module, 32

sslcommerz\_client.client module, 7

sslcommerz\_client.dataclasses module, 8

SSLCommerzClient (class in *sslcommerz\_client.client*), 7

status (*sslcommerz\_client.dataclasses.IPNResponse* attribute), 14

status (*sslcommerz\_client.dataclasses.IPNValidationStatus* attribute), 15



status (sslcommerz\_client.dataclasses.OrderValidationResponse attribute), 18

status (sslcommerz\_client.dataclasses.PaymentInitResponse attribute), 25

status (sslcommerz\_client.dataclasses.RefundInitiateResponse attribute), 26

status (sslcommerz\_client.dataclasses.Session attribute), 29

status\_code (sslcommerz\_client.dataclasses.APIResponse attribute), 8

store\_amount (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 10

store\_amount (sslcommerz\_client.dataclasses.Session attribute), 29

store\_id (sslcommerz\_client.dataclasses.Credential attribute), 11

store\_id (sslcommerz\_client.dataclasses.IPNResponse attribute), 14

store\_passwd (sslcommerz\_client.dataclasses.Credential attribute), 11

storeBanner (sslcommerz\_client.dataclasses.PaymentInitResponse attribute), 25

storeLogo (sslcommerz\_client.dataclasses.PaymentInitResponse attribute), 25

SUCCESS (sslcommerz\_client.dataclasses.RefundStatusEnum attribute), 27

SUCCESS (sslcommerz\_client.dataclasses.ResponseStatusEnum attribute), 28

success\_url (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 23

**T**

TAPNPAY (sslcommerz\_client.dataclasses.MultiCardNamesEnum attribute), 16

TELECOM\_VERTICAL (sslcommerz\_client.dataclasses.ProductProfileEnum attribute), 25

third\_party\_booking (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 23

THREE\_MONTHS (sslcommerz\_client.dataclasses.EMIOptionsEnum attribute), 12

THREE\_MONTHS (sslcommerz\_client.dataclasses.EMIOptionsResponseEnum attribute), 12

topup\_number (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 23

total\_amount (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 23

tran\_date (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 10

tran\_date (sslcommerz\_client.dataclasses.Session attribute), 29

tran\_id (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 10

tran\_id (sslcommerz\_client.dataclasses.PaymentInitPostData attribute), 23

tran\_id (sslcommerz\_client.dataclasses.Session attribute), 29

trans\_id (sslcommerz\_client.dataclasses.RefundInitiateResponse attribute), 26

TransactionBySessionResponse (class in sslcommerz\_client.dataclasses), 30

TransactionsByIDResponse (class in sslcommerz\_client.dataclasses), 31

TRAVEL\_VERTICAL (sslcommerz\_client.dataclasses.ProductProfileEnum attribute), 25

TRUE (sslcommerz\_client.dataclasses.BooleanIntEnum attribute), 10

type (sslcommerz\_client.dataclasses.Gateway attribute), 12

**U**

UNATTEMPTED (sslcommerz\_client.dataclasses.IPNOrderStatusEnum attribute), 13

UPAY (sslcommerz\_client.dataclasses.MultiCardNamesEnum attribute), 16

**V**

v (sslcommerz\_client.dataclasses.OrderValidationPostData attribute), 16

val\_id (sslcommerz\_client.dataclasses.BaseOrderResponse attribute), 10

val\_id (sslcommerz\_client.dataclasses.OrderValidationPostData attribute), 16

val\_id (sslcommerz\_client.dataclasses.Session attribute), 29

VALID (sslcommerz\_client.dataclasses.IPNOrderStatusEnum attribute), 13

VALID (sslcommerz\_client.dataclasses.OrderStatusEnum attribute), 16

valid\_decimal() (sslcommerz\_client.dataclasses.CartItem class method), 11

valid\_decimal() (sslcommerz\_client.dataclasses.PaymentInitPostData class method), 23

---

`valid_decimal()` (*sslcommerz\_client.dataclasses.RefundRequestPostData* class method), 27

`valid_emi_allow_only()` (*sslcommerz\_client.dataclasses.PaymentInitPostData* class method), 24

`validate_against_credential()` (*sslcommerz\_client.dataclasses.IPNResponse* method), 14

`validate_based_on_shipping_method()` (*sslcommerz\_client.dataclasses.PaymentInitPostData* class method), 24

`validate_IPN()` (*sslcommerz\_client.client.SSLCommerzClient* method), 7

`validate_num_of_item()` (*sslcommerz\_client.dataclasses.PaymentInitPostData* class method), 24

`validate_v()` (*sslcommerz\_client.dataclasses.OrderValidationPostData* class method), 16

`VALIDATED` (*sslcommerz\_client.dataclasses.OrderStatusEnum* attribute), 16

`value_a` (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 10

`value_a` (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 24

`value_a` (*sslcommerz\_client.dataclasses.Session* attribute), 29

`value_b` (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 10

`value_b` (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 24

`value_b` (*sslcommerz\_client.dataclasses.Session* attribute), 29

`value_c` (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 10

`value_c` (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 24

`value_c` (*sslcommerz\_client.dataclasses.Session* attribute), 30

`value_d` (*sslcommerz\_client.dataclasses.BaseOrderResponse* attribute), 10

`value_d` (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 24

`value_d` (*sslcommerz\_client.dataclasses.Session* attribute), 30

`vat` (*sslcommerz\_client.dataclasses.PaymentInitPostData* attribute), 24

`verify_key` (*sslcommerz\_client.dataclasses.IPNResponse* attribute), 14

`verify_sign` (*sslcommerz\_client.dataclasses.IPNResponse* attribute), 14

`verify_sign_sha2` (*sslcommerz\_client.dataclasses.IPNResponse* attribute), 14

`VISA_CARD` (*sslcommerz\_client.dataclasses.MultiCardNamesEnum* attribute), 16

**Y**

**YES** (*sslcommerz\_client.dataclasses.ShippingMethodEnum* attribute), 30