# sslcommerz-sdk

*Release 1.0.4*

**Munim Munna**

**Jun 01, 2021**

# CONTENTS:


SSLCOMMERZ
SDK

CHAPTER

ONE

# GETTING STARTED

Install it via pip (python>=3.0).

```
pip install sslcommerz-sdk
```

CHAPTER

TWO

# HOW TO USE

Create the views below depending on framework you are using.

```
from sslcommerz_sdk.enums import TransactionStatus

# TODO: create payment_handler.py file
from .payment_handler import payment_handler, store


def payment_init_view():
    # TODO: Freeze the cart, see what cart freezing is
    session, created = payment_handler.get_or_create_session(
        store=store,
        tran_id="test",
        currency="BDT",
        total_amount=100,
        cus_name="test",
        cus_email="test@test.com",
        cus_add1="test",
        cus_city="test",
        cus_postcode="1234",
        cus_country="test",
        cus_phone="123456",
        success_url="<URL to redirect cutomer when transaction is successful>",
        fail_url="<URL to redirect cutomer when transaction is failed>",
        cancel_url="<URL to redirect cutomer when transaction is cancelled>",
        ipn_url="<URL of ipn_view>",
    )
    # TODO: Redirect customer to session.redirect_url


def ipn_view():
    # TODO: Make this URL public, i.e accessible without logging in
    # TODO: Disable CSRF protection for this view
    # TODO: post_dict = {dict of request POST values}
    session, verified_right_now = payment_handler.verify_transaction(
        payload=post_dict,
    )
    if verified_right_now:
        if session.status == TransactionStatus.VALID:
            print(f"Tran ID: {session.tran_id} successful...")
```

(continues on next page)

(continued from previous page)

```
            # TODO: Update order payment status in your database
        else:
            print("Transaction failed/cancelled!")
            # TODO: Unfreeze the cart sothat customer can modify/delete the cart
```

## 2.1 Setup Payment Handler

### 2.1.1 Django ORM

Add "sslcommerz_sdk.contrib.django_app" to INSTALLED_APPS, and migrate.

```
python manage.py migrate
```

Get started with following payment_handler.py file for Django ORM. If you need to configure multiple store checkout Next Steps.

```
from sslcommerz_sdk.contrib.django_app.models import SslcommerzSession
from sslcommerz_sdk.handlers import PaymentHandler
from sslcommerz_sdk.orm_adapters.django import DjangoORMAdapter
from sslcommerz_sdk.store import SslcommerzStore
from sslcommerz_sdk.store_providers import SingleStoreProvider

store = SslcommerzStore(
    store_id="YOUR_STORE_ID",
    store_passwd="YOUR_STORE_PASSWORD",
    base_url="https://sandbox.sslcommerz.com",
)
payment_handler = PaymentHandler(
    model=SslcommerzSession,
    orm_adapter=DjangoORMAdapter(),
    store_provider=SingleStoreProvider(store=store),
)
```

When you are done, checkout Next Steps.

### 2.1.2 Sqlalchemy

Get started with following payment_handler.py file for Sqlalchemy and Flask-Sqlalchemy. If you need to configure multiple store checkout Next Steps.

```
from sslcommerz_sdk.handlers import PaymentHandler
from sslcommerz_sdk.orm_adapters.sqlalchemy import (
    SqlalchemyORMAdapter,
    sslcommerz_session_sqlalchemy_model_factory,
)
from sslcommerz_sdk.store import SslcommerzStore
from sslcommerz_sdk.store_providers import SingleStoreProvider

# TODO: import your declarative base or db.Model as BaseModel
```

(continues on next page)

(continued from previous page)

```
# TODO: import sqlalchemy session or db.session as db_session

SslcommerzSession = sslcommerz_session_sqlalchemy_model_factory(BaseModel)
store = SslcommerzStore(
    store_id="YOUR_STORE_ID",
    store_passwd="YOUR_STORE_PASSWORD",
    base_url="https://sandbox.sslcommerz.com",
)
payment_handler = PaymentHandler(
    model=SslcommerzSession,
    orm_adapter=SqlalchemyORMAdapter(db_session=db_session),
    store_provider=SingleStoreProvider(store=store),
)
```

Then you can create the model generating alembic migration files or directly from a python shell prompt.

```
# Run in python shell
from app.db import engine
from .payment_handler import SslcommerzSession
SslcommerzSession.__table__.create(engine)
```

When you are done, checkout Next Steps.

### 2.1.3 PynamoDB

Get started with following `payment_handler.py` file for PynamoDB. If you need to configure multiple store checkout Next Steps.

```
from sslcommerz_sdk.handlers import PaymentHandler
from sslcommerz_sdk.orm_adapters.pynamodb import (
    PynamodbORMAdapter,
    sslcommerz_session_pynamodb_model_factory,
)
from sslcommerz_sdk.store import SslcommerzStore
from sslcommerz_sdk.store_providers import SingleStoreProvider

SslcommerzSession = sslcommerz_session_pynamodb_model_factory(region="us-east-1")
store = SslcommerzStore(
    store_id="YOUR_STORE_ID",
    store_passwd="YOUR_STORE_PASSWORD",
    base_url="https://sandbox.sslcommerz.com",
)
payment_handler = PaymentHandler(
    model=SslcommerzSession,
    orm_adapter=PynamodbORMAdapter(),
    store_provider=SingleStoreProvider(store=store),
)
```

You can customize the model with additional parameters to model factory.

```
SslcommerzSession = sslcommerz_session_pynamodb_model_factory(
    region="us-east-1",
```

(continues on next page)

(continued from previous page)

```
    table_name="sslcommerz_sdk_session",
    read_capacity_units=1,
    write_capacity_units=1,
)
```

Then you can create the model directly from a python shell prompt.

```
# Run in python shell
from .payment_handler import SslcommerzSession
SslcommerzSession.create_table()
```

When you are done, checkout Next Steps.

## 2.2 Setup Store

Register a sandbox account for development purpose and set it up as below.

```
from sslcommerz_sdk.store import SslcommerzStore

store = SslcommerzStore(
    store_id="YOUR_STORE_ID",
    store_passwd="YOUR_STORE_PASSWORD",
    base_url="https://sandbox.sslcommerz.com",
)
```

For production create a marchant account on SSLCOMMERZ and set it up as below.

```
from sslcommerz_sdk.store import SslcommerzStore

store = SslcommerzStore(
    store_id="YOUR_STORE_ID",
    store_passwd="YOUR_STORE_PASSWORD",
    base_url="https://securepay.sslcommerz.com",
)
```

### 2.2.1 Customize Store

You can customize the store with additional parameters for testing purpose.

```
from sslcommerz_sdk.store import SslcommerzStore

store = SslcommerzStore(
    store_id="YOUR_STORE_ID",
    store_passwd="YOUR_STORE_PASSWORD",
    base_url="https://securepay.sslcommerz.com",
    session_url="/gwprocess/v4/api.php",
    validation_url="/validator/api/validationserverAPI.php",
    transaction_url="/validator/api/merchantTransIDvalidationAPI.php",
)
```

### 2.2.2 Setup Multiple Store

To handle more than one store use `MultpleStoreProvider` instead in your payment handler.

```
from sslcommerz_sdk.store_providers import MultpleStoreProvider

def get_store_by_id(store_id):
    # TODO: Retrieve password of the store by store_id
    return SslcommerzStore(
        store_id=store_id,
        store_passwd="PASSWORD of that store",
        base_url="https://sandbox.sslcommerz.com",
    )

payment_handler = PaymentHandler(
    model="YOUR_MODEL",
    orm_adapter="YOUR_ORM_ADAPTER",
    store_provider=MultpleStoreProvider(get_store_by_id=get_store_by_id),
)
```

When you are done, checkout Next Steps.

## 2.3 Best Practices

### 2.3.1 Freeze your cart

A shopping cart should be freezed when a payment session is initialized sothat the customer can't empty his cart or modify the cart which changes the total amount payable. A cutomer can do so if he inadvertedly gets into the cart page pressing browser's back button (trust me they do that a lot). Letting the customer modify the cart during an ongoing transaction may result in catestrophic results. SSLCOMMERZ SDK prevents such catestrophys by raising a `TotalAmountTamperedException` exception and thus stopping the customer from proceeding to checkout when payable amount is altered during an ongoing transaction which will present a 500 server error to the customer.

The shopping cart should be unfreezed when the payment fails or the customer cancels the transaction so that he can modify the cart.

Checkout Next Steps.

## 2.4 Contributing

PR should pass the tests and lint commands, checkout the following to get started.

- CONTRIBUTING.md.
- CODE_OF_CONDUCT.md.

## 2.5 License

This project is published under MIT LICENSE.

CHAPTER

THREE

# INDICES AND TABLES

- genindex
- modindex
- search