1. Technical Requirements for SDK Libraries Development

Here’s an outline of the core components for SDK library development in

Python languages.

Core Components of SDK Libraries

1. Initialization and Configuration

o Purpose: Set up the SDK and configure options (e.g., authentication

credentials, API endpoints, logging).

o Common Features:

▪ Environment configuration (development, production).

▪ Setting authentication keys/tokens.

▪ Customizable parameters like timeouts, retries, etc

2. API Client

o Purpose: The interface for interacting with the underlying APIs (REST,

SOAP, etc.).

o Common Features:

▪ HTTP request handling (GET, POST, PUT, DELETE).

▪ Pagination support.

▪ Error handling and retries.

3. Authentication Module

o Purpose: Enable secure communication with the API or backend.

o Common Authentication Methods:

▪ API Keys.

▪ OAuth 2.0 (Bearer tokens).

▪ Basic Authentication

4. Models or Data Structures

o Purpose: Represent API responses as objects or classes for easier

manipulation.

o Features:

▪ Mapping JSON responses to language-specific objects.

▪ Handling nested data structures

5. Utilities

o Purpose: Provide helper methods for common tasks.

o Examples:

▪ Logging.

▪ Input validation.

▪ Utility functions for date/time or string manipulation.

6. Error Handling

o Purpose: Standardize error management across the library.

o Features:

▪ Custom error classes or exceptions.

▪ Categorizing errors (e.g., network issues, authentication errors, APIspecific errors).

▪ Detailed error messages.

7. Testing Utilities

o Purpose: Support testing of apps using the SDK.

o Features:

▪ Mock server or API stubs for simulating backend response

▪ Test hooks for simulating SDK behaviors.

8. Documentation and Code Samples

o Purpose: Make the SDK easy to use for developers.

o Examples:

▪ Inline comments and docstrings.

▪ Sample integration code.

3. Python

• Core Libraries:

o requests for HTTP requests.

o pydantic for data validation and parsing.

• Key Components:

o Classes or modules for grouping related API endpoints.

o Extensive use of Python's exception hierarchy for error handling.

• Packaging:

o Distribute via PyPI with setuptools or poetry.

**Key Components**

**o The building blocks the SDK, defining its structure and functionality.**

**a. Configuration**

**o Centralizes settings like API base URL, authentication tokens, and timeout**

**values.**

**b. API Client**

**o Manages HTTP requests and integrates with the configuration.**

**c. Authentication**

**o Handles API key, token management, or OAuth**

**d. Data Models**

**o Defines Go structs that represent API requests and responses.**

**e. API Endpoints**

**o Encapsulates API logic and endpoints for CRUD operations.**

**f. Utilities**

**o Helper functions for repetitive tasks like query building or pagination.**

**g. Error Handling**

**o Provides structured errors to make debugging easier.**

**h. Testing Utilities**

**o Mocks for HTTP requests and testing helpers.**

Packaging

o Proper packaging ensures usability and ease of distribution.

a. Directory Structure

o Organize the SDK into logical directories for scalability:

b. Go Modules

o Enable dependency management with Go modules (go.mod).

c. Semantic Versioning

o Tag releases using semantic versioning (v1.0.0).

o Publish tags for version control.

d. Documentation

o Use godoc for API documentation

o Provide a README.md file for usage examples

SDK Non-Functional Requirements

1. Modularity:

o Break the SDK into modules based on API features or endpoints.

o Example: Authentication module, Data module, etc.

2. Compatibility:

o The SDK works seamlessly with supported language versions/platforms (e.g.,

Python 3.7+).

o Integration tests pass for all officially supported platforms and environments.

3. Scalability:

o The SDK can handle up to 1,000 concurrent API calls without failures or

significant degradation.

4. Consistency:

o Follow the idiomatic coding style of the target language.

o Example: Python's PEP 8 callback patterns.

5. Versioning:

o Use semantic versioning (e.g., 1.2.3 where major.minor.patch).

6. Testing:

o Ensure 100% test coverage with unit and integration tests.

o Use CI/CD pipelines for automated testing.

7. Documentation:

o Provide detailed developer documentation, including setup, API references,

and examples.

8. Error Reporting:

o Ensure consistent error codes and messages across all SDKs.

9. Ease of Use:

o Create a simple, intuitive API that abstracts complexity

o Follow naming conventions and standards for the target platform.

10.Security:

o All data transmitted via the SDK is encrypted (e.g., HTTPS).

o token-based authentication).

o Sensitive data like API keys is never written to logs within the SDK.

11.Performance:

o Optimize the SDK for minimal impact on the host application’s performance

(e.g., memory and CPU usage).

12.Backward Compatibility:

o Ensure new SDK versions do not break existing implementations.

Key Team Roles:

SDK Developers: Build and maintain the SDK (Gebeya Team)

QA Engineers: Test the SDK across platforms and scenarios. (STEP QA Team)

Technical Writers: Create and maintain documentation. (STEP Tech Team)

Support Engineers: Handle developer queries and issues. (STEP Operation Team)

API’s Specifications for four REST API’s required for the SDK libraries

development (Authentication, C2B, B2C, Stkpush(NI Push)

1. Authentication

Request:

• Endpoint: https://apisandbox.safaricom.et/v1/token/generate?grant\_type=client\_cr

edentials

• Method: GET

• Authorization Header: Basic Auth

• Body:

`// HTTP Basic authentication

Username

password

// HEADERS`,

`// Authorization: Basic `

// \*\*Credential = Base64 Encode (Consumer Key: Consumer Secret)\*\*

{

Authorization: Basic

Q2RtTmJkdDBpQk4xb3FEZkthc200ZGFiZHBLbXRhTm46RExLRzdQQnVuNzIwR1ppbQ==;

}

// PARAMS

grant\_type;

client\_credentials;`

Result:

Body

{

"access\_token": "05RpFfThkohCr4K1FAtSjXNDAz1a",

"token\_type": "Bearer",

"expires\_in" : "3599"

}

Error Result Codes:

Body

{

"resultCode": "999991",

"resultDesc": "Invalid client id passed."

}

Possible Response Code and Description and Cause and Mitigation

1. Response code: 999991, Invalid client id and Description: passed Cause:

Incorrect basic Authorization username, Mitigation: Input the correct username

2. Response code: 999996 , Description: Invalid Authentication passed Cause:

Incorrect authorization type Mitigation: Select type as Basic Auth

3. Response code:999997 Description: Invalid Authorization Header Cause:

Incorrect basic authorization password, Mitigation: Input the correct password

4. Response code: 999998, Description: Required parameters [grant type] is

invalid, or empty Incorrect grant type Mitigation: Select grant type as client

credentials

!Note this Authentication Api will be used across all the Api’s listed below

2. The M-PESA online checkout API (STK push)

Also known as NI push, facilitates Merchant/Business and initiated Customer to Business

(C2B) Payments. Upon integrating with the API, businesses can promptly send payment

requests to customers' M-PESA registered phone numbers, triggering a USSD push prompt

on their mobile devices. Customers are then prompted to enter their M-PESA PIN to

authorize and complete the payment.

This API simplifies transactions for customers, allowing them to confirm payments

effortlessly by entering their M-PESA PIN directly on their mobile phones. With the ability to

preset all necessary payment information beforehand, businesses using this API can

significantly reduce the risk of erroneous payments and ensure accurate transaction

processing.

Request

Endpoint: https://apisandbox.safaricom.et/mpesa/stkpush/v3/processrequest

Method: POST

Asynchronous API

Authorization Header: Pass your bearer token in the request header to authorize this call.

Request Body

{

"MerchantRequestID": "SFC-Testing-9146-4216-9455-e3947ac570fc"

"BusinessShortCode": "554433",

"Password": "123",

"Timestamp": "20160216165627",

"TransactionType": "CustomerPayBillOnline",

"Amount": "10.00",

"PartyA": "251700404789",

"PartyB": "554433",

"PhoneNumber": "251700404789",

"TransactionDesc": "Monthly Unlimited Package via Chatbot",

"CallBackURL": "https://apigee-listener.oat.mpesa.safaricomet.net/api/ussdpush/result",

"AccountReference": "DATA",

"ReferenceData": [

{

"Key": "BundleName",

"Value": "Monthly Unlimited Bundle"

},

{

"Key": "BundleType",

"Value": "Self"

},

{

"Key": "TINNumber",

"Value": "89234093223"

}

]

}

Response

{

"MerchantRequestID": "9cae-431a-9bb5-0e58fd6aced6",

"CheckoutRequestID": "ws\_CO\_1202202404292020468057",

"ResponseCode": "0",

"ResponseDescription": "Success. Request accepted for processing",

"CustomerMessage": "Success. Request accepted for processing"

}

Result

{

"Body": {

"stkCallback": {

"MerchantRequestID": "29115-34620561-1",

"CheckoutRequestID": "ws\_CO\_191220191020363925",

"ResultCode": 0,

"ResultDesc": "The service request is processed successfully.",

"CallbackMetadata": {

"Item": [{

"Name": "Amount",

"Value": 1.00

},

{

"Name": "MpesaReceiptNumber",

"Value": "NLJ7RT61SV"

},

{

"Name": "TransactionDate",

"Value": 20191219102115

},

{

"Name": "PhoneNumber",

"Value": 254708374149

}]

}

}

}

}

Result

{

"Body": {

"stkCallback": {

"MerchantRequestID": "29115-34620561-1",

"CheckoutRequestID": "ws\_CO\_191220191020363925",

"ResultCode": 0,

"ResultDesc": "The service request is processed successfully.",

"CallbackMetadata": {

"Item": [{

"Name": "Amount",

"Value": 1.00

},

{

"Name": "MpesaReceiptNumber",

"Value": "NLJ7RT61SV"

},

{

"Name": "TransactionDate",

"Value": 20191219102115

},

{

"Name": "PhoneNumber",

"Value": 254708374149

}]

}

}

}

}

Possible Result Code (Error):

{

"Body": {

"stkCallback": {

"MerchantRequestID": "29115-34620561-1",

"CheckoutRequestID": "ws\_CO\_191220191020363925",

"ResultCode": 1032,

"ResultDesc": "Request canceled by user."

}

}

}

3. Customer to Business API’s (C2B)

The M-PESA Customer to Business (C2B) APIs for Validation and Confirmation are essential

tools for businesses to verify and confirm transactions initiated by customers through all

M-PESA mobile money channels.

A. Register URL - Receive Payment Notification

The M-PESA Register URL API serves as a crucial tool for businesses and developers

seeking to integrate their systems with the M-PESA mobile money platform. This API

facilitates the registration of validation and confirmation URLs, which are essential for

handling transaction notifications and communication between the M-PESA platform and

external systems when customers pay for the business.

Wanna try out ?

• Make sure you have publicly exposed endpoints that are accessible over the

Internet.

• You are free to register your URLs multiple times on sandbox or even overwrite the

existing ones.

What do you need before you register URL on production ?

• Use publicly available (Internet-accessible) IP addresses or domain names.

• All Production URLs must be HTTPS, on Sandbox you're allowed to simulate using

HTTP.

Request

Endpoint: https://apisandbox.safaricom.et/v1/c2b-registerurl/register?apikey={username}

Method: POST

No Auth: use apikey as a query parameter on the URL

Request Body:

{

"ShortCode": "101010",

"ResponseType": "Completed",

"CommandID": "RegisterURL",

"ConfirmationURL": "http://mydomain.com/c2b/confirmation",

"ValidationURL": "http://mydomai.com/c2b/validation"

}

Response

{

"header": {

"responseCode": 200,

"responseMessage": "Request processed successfully",

"customerMessage": "Request processed successfully",

"timestamp": "2024-02-12T02:20:31.390"

}

}

Error

200 Success (Request processed successfully)

400 Short Code already Registered

B. Endpoint: https://apisandbox.safaricom.et/v1/c2b/payments

Method: POST

Request:

{

"RequestRefID": "{{$guid}}",

"CommandID": "CustomerPayBillOnline"

"Remark": "Here is a remark",

"ChannelSessionID": "10100000037656400042",

"SourceSystem": "USSD",

"Timestamp": "2014-09-30T11:03:19.111+03:00",

"Parameters": [

{

"Key": "Amount",

"Value": "500"

},

{

"Key": "AccountReference",

"Value": "TU781RE"

},

{

"Key": "Currency",

"Value": "ETB"

}

],

"ReferenceData": [

{

"Key": "AppVersion",

"Value": "v0.2"

}

],

"Initiator": {

"IdentifierType": 1,

"Identifier": "251799100026",

"SecurityCredential":

"INjHSGNd8GT5bUyYUKKZKsD/jqvCYHR3Ld88xMnEgtM3kje6Fqf9JihIgeHHqN1sdQ8co8a

wajdD2gKPVbFWDklR90XWg/c1XqMhrKJUL4wagQxWREiFGgT79PMa7u6S1Pa/e/7IpwyoHn

t6Scawat5Q64TU6qKO1cuaJM/hN/udjec3+9TnqGPIsGZC/lnDX+F7N5UiNuq1gF98ta5UU9e

y9v4x80KPlQ7clyjQPdoi0qVGPtCMQjwPojRJfrvYFMEHDx/h64CFE+02/qoFqmjE0UsBe0tbA

ZXN8X18pGjfwpjh+ZpvfkRJ+HpqmbC45MHuj5wrcTWk31av59vWYszOG3cIMzJqT+iiiFLiOIfn

yMyqWV6/huRzgINVRmAcenucbPJf+GEH0Rrf1MSA5JmnsfymDQXPC/AMrO1jc2QPhRE4Oi

xabEQrWN6fx6sZj2cV4Cq1CTI8zoWluugweg==",

"SecretKey":

"NqroGXabob/7Nu5ELALwSOk9YbFEYSC7CmecEPQesG4jLLhVF9tKpH5VzCmy2cNw4v06

XXOPnv2UnaNAAx2Oo1PjW9B02Xt/y5eNcbgoWVhyfknebf8jM8UcsjHZUN/8vwZSa2dN6N

UO/6Zy2HuabZTjLDCqK6TEHyfngDMaMjoBYlxiIot9KWjRUsr97D/m/PY1CAuLq0tnvve5r0Sai

PxH1+PKMTTdPx8Sp1pOACJbM1CXZ+RjYn/YHQLumcMcHADX9n15FvYYGJ/zf5K18GIpw/Q

AIHIy80t8ZpZFZ4n3H61ajQlRfPPDy+IxJ7vIsyRK1g+SEtiknpEVtxx8Lw=="

},

"PrimaryParty": {

"IdentifierType": 1,

"Identifier": "251799100026"

},

"ReceiverParty": {

"IdentifierType": 4,

"Identifier": "370360",

"ShortCode": "370360"

}

}

Response:

{

"RequestRefID": "29900fe1-ac90-45ce-9443-19eec5f31234",

"ResponseCode": "0"

"ResponseDesc": "The service request is processed successfully.",

"TransactionID": "",

"AdditionalInfo": []

}

Possible response type (Error):

{

"RequestRefID": "17b0ca4b-e721-4e54-9e17-315d3f968c78",

"ResponseCode": "2001",

"ResponseDesc": "The initiator information is invalid.",

"TransactionID": null,

"AdditionalInfo": []

}

4. Pay Out B2C)

The Business to Customer (Pay out) API facilitates seamless payments from businesses to

individual customers. It caters to various business needs, including salary payments,

betting wining payouts, bulk disbursements, cashback rewards, promotional payouts, and

loan disbursements. This API operates asynchronously, meaning that the transaction's

completion status is determined by the callback result.

Request

• Endpoint: https://apisandbox.safaricom.et/mpesa/b2c/v1/paymentrequest

• Method: POST

• Authorization Header: Pass your bearer token in the request header to authorize

this call.

• Request Body:

{

"InitiatorName": "testapi",

"SecurityCredential":

"iSHJEgQYt3xidNVJ7lbXZqRXUlBqpM/ytL5incRQISaAYX/daObQopdHWiSVXJvexSoYCt9mm

b6+TiikmTrGZm5fbaT1BeuPKDF9NFpOLG3n3hUZE2s5wNJvFxD3sM62cBdCQulFquFBc0C

wHpq/K2cU1MN8lahvYp+vHnmGODogMBDk8/5Q+5CuRRFNRIt50xM0r10kUHVeWgUa71

H6oK2RqXnog4EPTnanMlswz7N3J8jeIKzgGUwnJA8va5CvuNWu2B2L1cAm9g6pGribcgFZ2s

gzByJpRWBkfntjGgzsYXh+K3fPZmxWyTQi7TscSvujH1EaS7JxvCIWMM3K0Q==",

"Occassion": "Disbursement",

"CommandID": "BusinessPayment",

"PartyA": "101010",

"PartyB": "251700100100",

"Remarks": "Test B2C",

"Amount": 12,

"QueueTimeOutURL": "https://mydomain.com/b2c/timeout",

"ResultURL": "https://mydomain.com/b2c/result"

}

Response

Success

{

"ConversationID": "AG\_20240209\_70205ca849aecd7fbd7f",

"OriginatorConversationID": "77fd-4542-b4f4-1748deeeb48f",

"ResponseCode": "0",

"ResponseDescription": "Accept the service request successfully."

}

Error

{

"requestId": "fc21-42cd-af6b-59613899fed5",

"errorCode": "400.002.02",

"errorMessage": "Bad Request - Invalid CommandID"

}

Result

Success

{

"Result": {

"ResultType": 0,

"ResultCode": 0,

"ResultDesc": "The service request is processed successfully.",

"OriginatorConversationID": "93de-494d-a96b-8dba903c6e59",

"ConversationID": "AG\_20240209\_702031abc8afeb07c769",

"TransactionID": "SB96BUP8I4",

"ResultParameters": {

"ResultParameter": [

{

"Key": "TransactionAmount",

"Value": 12

},

{

"Key": "TransactionReceipt",

"Value": "SB96BUP8I4"

},

{

"Key": "ReceiverPartyPublicName",

"Value": "251700100100 - Abebe Bekele"

},

{

"Key": "TransactionCompletedDateTime",

"Value": "09.02.2024 05:33:37"

},

{

"Key": "B2CUtilityAccountAvailableFunds",

"Value": 4531.00

},

{

"Key": "B2CWorkingAccountAvailableFunds",

"Value": 32233.00

},

{

"Key": "B2CRecipientIsRegisteredCustomer",

"Value": "Y"

},

{

"Key": "B2CChargesPaidAccountAvailableFunds",

"Value": 0.00

}

]

},

"ReferenceData":

"ReferenceItem": {

"Key": "QueueTimeoutURL",

"Value": "https:\/\/apisandbox.safaricom.et\/mpesa\/b2cresults\/v1\/submit"

}

}

}

}

Error

{

"Result": {

"ResultType": 0,

"ResultCode": 2001,

"ResultDesc": "The initiator information is invalid.",

"OriginatorConversationID": "b539-498a-9d8b-09d34973570c",

"ConversationID": "AG\_20240209\_7020389af8e641219a2c",

"TransactionID": "SB90000000",

"ReferenceData": {

"ReferenceItem": {

"Key": "QueueTimeoutURL",

"Value": "https:\/\/apisandbox.safaricom.et\/mpesa\/b2cresults\/v1\/submit"

}

}

}

}