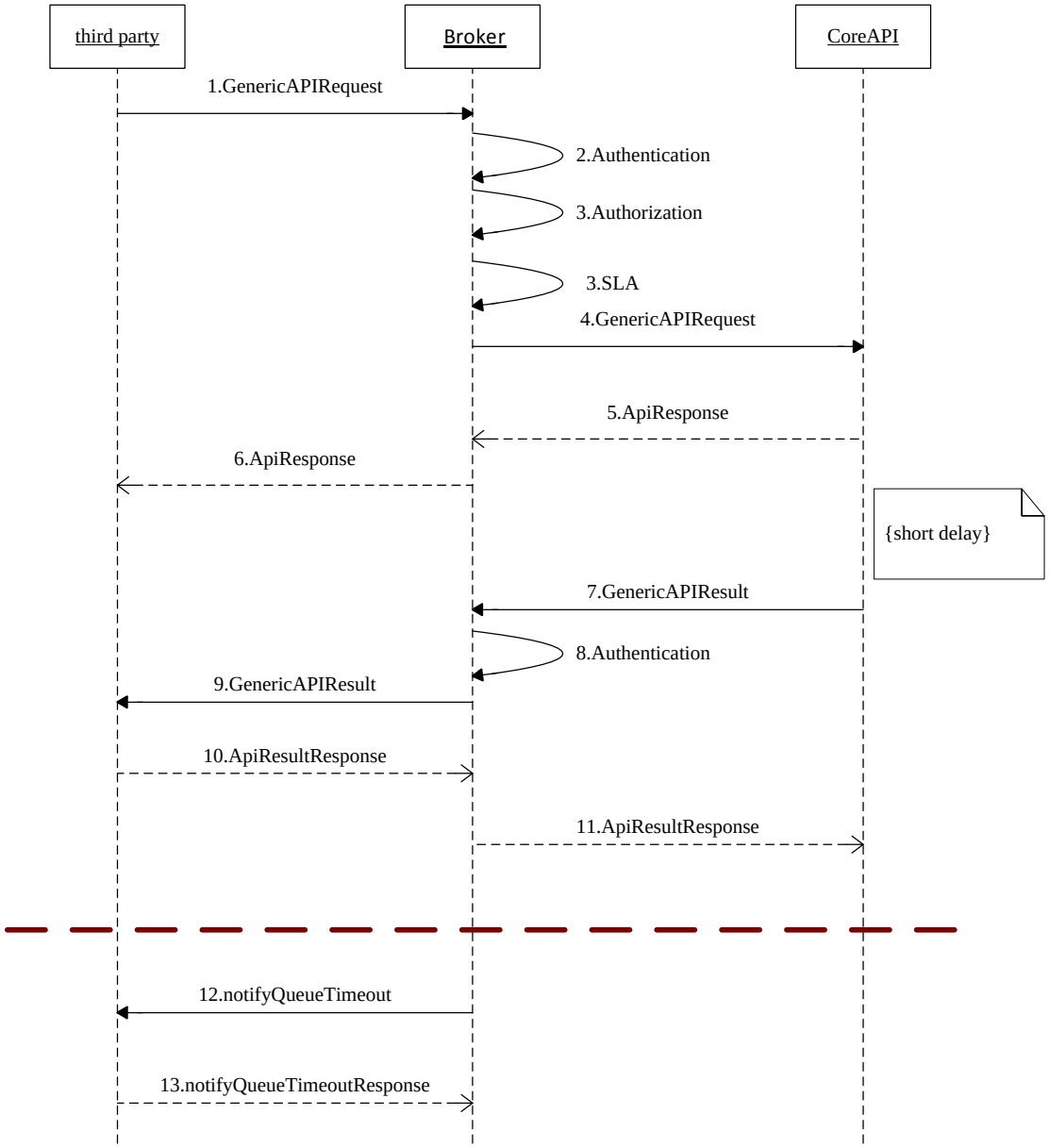
Safaricom Integration Guide B2C – Quick Setup

Contents

Introduction

This document describes the steps to be followed when testing on M-Pesa for a business to customer (B2C) transaction. The following diagram describes the process flow:



Step 1: What you will require for testing

This is an important step as it allows us to gather the requirements for proceeding with the tests. The following will be used for testing and should be installed:

Firstly, you will need to know how you are going to connect to us.

	IT VPN	EXTRANET VPN	DATA (P2P)	ports
Test bed	196.201.214.136	10.26.0.75	192.168.9.48	8310 - http, 18423 – https.
Production Request	196.201.214.137	10.26.0.76	192.168.9.49	8310 - http, 18423 – https.
Production Result	196.201.214.127	10.26.0.77	192.168.9.50	8310 - http, 18423 – https.

For broker end points for testing using the IP and the ports above the general format will be:

http: http://IPaddress:port/mminterface/request

https: https://IPaddress:port/mminterface/request

http: http://IPaddress:port/queryTransactionService/services/transaction

https: https://IPaddress:port/queryTransactionService/services/transaction

- You will be sent a folder called Third Party v3.This will include the WSDL files and documents you will need to use together with SOAP. You can use the link below to download them: https://drive.google.com/folderview?id=0B_BRYv-DOjeTamo3TFVBX2VpcTQ&usp=sharing.
- You will need to have SOAP installed to send request and see responses from broker.
- Ensure you have shared your source IP with us.
- Ensuring you have a Test Paybill, service id, username and password assigned to carry out test transactions, otherwise please contact the FS Planning team on FSPlanning@safaricom.co.ke and it shall be created. However, you will need to send your source IP and the SPid you are going to use to the email above.
- You can use the default testing credentials:

  Test MPESA credentials
  ==========

  Broker

- SPID: 107020
- ServiceID: 107020000
- SpPassword: Abdul123!

Core API/MPESA

- Short code: 015689
- Initiator username: apitest
- Initiator password: Abdul123

- Editing and requests will be sent and done on one particular WSDL called CBP_Interface_Request>GenericAPI>B2C.

Step 2: Setting up the listener

The listener is used to process events as they occur. Thus, for testing purposes to receive responses for successful and erroneous transactions, we will be required to set up a listener:

- You will be required to import the corresponding WSDL for the request (CBPInterface_Result).
- Look up the IP address for your computer using start>run>cmd and type ipconfig.
- Go to CBPInterface_Request>RequestBinding, right click and generate mock service.
- When creating a listener/MockService fill up the port that you want to run this service on.
- This will create a result binding mock service and click play to start the listener.
- In the CBPInterface_Request>GenericAPI>B2C go to the following code and change the <result url> to your IP and port that you had put above. It will be in this format. ip:port/resultfile

```
<CallerType>2</CallerType>
<ThirdPartyID>broker_4</ThirdPartyID>
<Password>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxx==</Password>
<ResultURL>http://172.21.51.95:8087/</ResultURL>
</Caller>
```

Step 3: Testing

After importing all the WSDL's to SOAP, you shall begin with setting up the endpoints. Depending on your connection to us, refer to step 1. Ensure you telnet to the endpoints to confirm connectivity.

3.1 HTTP TEST

Ensure your server configurations and ports are on http. Please inform us via mail on the ports that you will be going to use.

1. Perform a telnet and share output for both production and test bed IP and ports.

2. Send request and check for any errors.

3. Confirm that you are able to post a successful request and check for a successful response on your callback URL.

4. Confirm that you have configured Callback URL (Result URL) in your request to receive GenericAPIResult from broker with the IP and ports that are configured in the firewall on our side.

3.2 MOVING TO HTTPS (2-Way SSL)

Generate the Certificate Signing Request (CSR) depending on the web server platform while strictly adhering to the following guidelines:

The following fields must be populated with valid and verifiable information:
i. Common Name= **FQDN of hosting server using standard notation**
ii. Organization Name= **Business Name as recognized by Safaricom**
iii. Locality= **Region where the organization is located**
iv. State / Province= **Province/County**
v. Country = **Country of registration of the organization**
vi. Minimum Key Size is = **2048bits**

2. Important Considerations for Certificate Requestors:
i. The common name needs to be a Fully Qualified Domain Name (FQDN) and unique for every request made. Even if the name is not verifiable via public DNS, please select a similar name anyway that follows standard DNS notation since other arrangements can be made for the mapping of that name to the corresponding IP address e.g. **endpoint1.3rdpartyname.co.ke**

ii. The private key generated along with the CSR is solely for the Requestor to secure. We do encourage that you protect this key with a password and ensure relevant access controls are put in place for the server that will be hosting the private key. This key should NOT be shared with anyone. Any compromise of the key shall be communicated immediately to Safaricom and a new one generated so that a new certificate can be generated.

iii. Safaricom will issue certificates with a maximum validity of 2 years. The requestor will be responsible for tracking the expiry of the certificate issued and raise a new request at least 2 months before the expiry of the certificate.

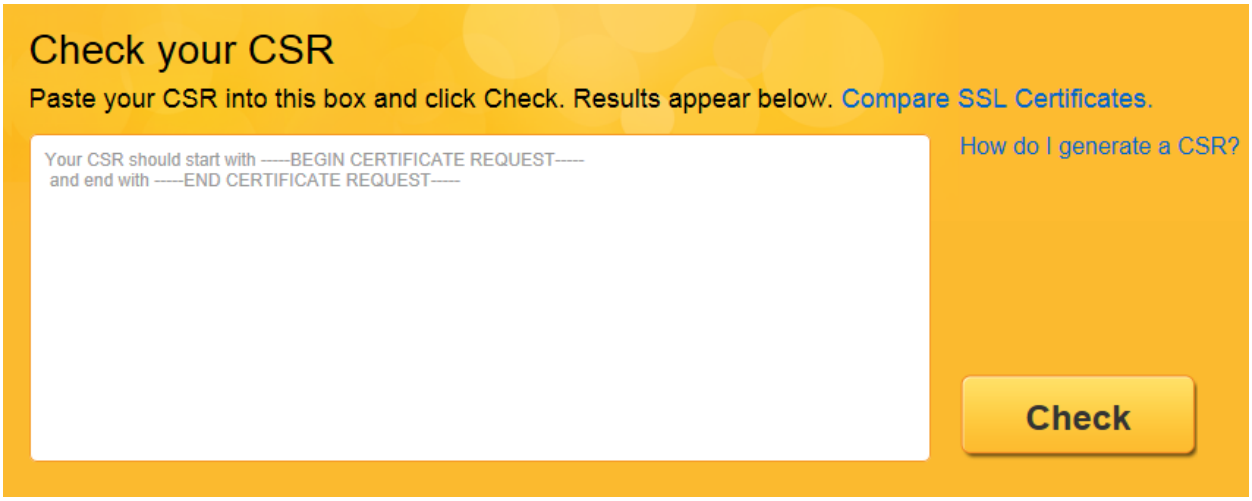
iv. Certificates generated are solely for the private use when connecting to Safaricom systems and for the purpose intended as prescribed by Safaricom. Breach of acceptable usage for the certificate issued will result in immediate revocation and possible legal action if deemed necessary by Safaricom.

3. Checking for Errors on the CSR:

i. Open Symantec CSR checker to make sure the CSR has been generated properly using the link https://cryptoreport.websecurity.symantec.com/checker/views/csrCheck.jsp

ii. Copy and paste the contents of the CSR in the box then click check to see if there are any errors.

iii. Submit your CSR for signing after checking and correcting errors



Steps:

- Change your ports to https and also inform us of the change on our end.

- Once the above is done you can send the .csr to FSPlanning@safaricom.co.ke for signing.

- Once you have the signed certificate, configure testbroker.crt added as a CA in the local trust store, this will be used to authenticate broker.

-You also add the .p7b as client and server certificates.

-Change the endpoint you are sending requests to our https endpoint (same ip: port 18423) and add https in front of the URL.

- If there is any issue, troubleshoot with our technical teams FSPlanning@safaricom.co.ke or VAS-SUPPORT2@safaricom.co.ke

Step 4: Query Transaction and Queue Timeout

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  xmlns:req="http://api-v1.gen.mm.vodafone.com/mminterface/request">
  <soapenv:Header/>
  <soapenv:Body>
   <req:RequestMsg><![CDATA[<?xml  version="1.0" encoding="UTF-8"?>
```

```
<Request>
  <Identity>
      <Caller>
    <CallerType>2</CallerType>   //variable that doesn't change
    <ThirdPartyID>broker_4</ThirdPartyID>  //variable that doesn't change
    <Password>k+JtvqNV3eg=</Password>   //variable that doesn't change
       <ResultURL>http://172.29.81.18:8089/mockResultBinding</ResultURL>  //the listener on your side receiving the results
  </Caller>
 <Initiator>
     <IdentifierType>14</IdentifierType> //variable that doesn't change
    <Identifier>crystal</Identifier>  //will be provided to you
    <SecurityCredential>D6MkFLPIn8df27Iu4V+WsA==</SecurityCredential>  //will be provided to you
  </Initiator>
  <ReceiverParty>
    <IdentifierType>4</IdentifierType> //variable that doesn't change
    <Identifier>12345</Identifier>  //your short code
  </ReceiverParty>
 </Identity>
 <Transaction>
  <CommandID>TransactionStatusQuery</CommandID>  //the command id for querying transaction status
  <OriginatorConversationID>D7866050CE74C4446B9BE97D9FBACC69041</OriginatorConversationID>   //always changes on every request,
should be unique
  <Remark>Christmas Bonus</Remark> //remark
  <Parameters>
    <Parameter>
     <Key>ReceiptNumber</Key>
     <Value>0000000001199549</Value> //the receipt number of the transaction you are querying
    </Parameter>

  </Parameters>

  <ReferenceData>
    <ReferenceItem>
     <Key>Occasion</Key>
     <Value>Christmas</Value>
    </ReferenceItem>
  </ReferenceData>
  <Timestamp>2014-02-12T23:53:19.0000521Z</Timestamp>  //the timestamp
 </Transaction>
 <KeyOwner>1</KeyOwner> //constant variable, does not change
</Request>]]>
</req:RequestMsg>
  </soapenv:Body>
</soapenv:Envelope>
```

Step 5: Security

The initiator security credential tag

The Caller will be required to confirm its authority to act on behalf of the Initiator (in other words, a specific B2C organization) by presenting the user name and password for the Initiator, the latter encrypted with the public key from an X509 certificate issued to the Initiator specifically for this purpose.

The following algorithm must be followed by the Initiator to encrypt passwords:

First, create the block of data to be encrypted:

- Write the unencrypted password value.

- Then, encrypt the block of data created in step 1 with the public portion of the password key certificate. Use the RSA algorithm, and use PKCS #1.5 padding (not OAEP), and add the result to the encrypted stream – this becomes the encrypted password which is submitted via the API.

- Convert the resulting encrypted byte array into a string using base64 encoding. Present this base64 encoded string in the API request as the initiator SecurityCredential value.

The SOAP Header

You will be required to set up the SOAP header, you will require three parameters:

-SpID

-password

-timestamp

The following will be the used to encrypt the password: BASE64(sha256(spId + Password + timeStamp))

N.B: The password is first converted to sha256, and then result is converted to base64.

The end result for the header should be like the sample code below:

```
<soapenv:Header>
    <tns:RequestSOAPHeader xmlns:tns="http://www.huawei.com/schema/osg/common/v2_1">
      <tns:spId>100001</tns:spId> //SPID, provided by us

<tns:spPassword>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx==</tns:spPassword> //Password, provided by us. Encrypted using
base64(sha256(SPid+Password+Timestamp))
      <tns:timeStamp>yyyymmddhhmmss</tns:timeStamp> //Timestamp in format yyyymmddhhmmss
      <tns:serviceId>100000001</tns:serviceId> //Service id, provided by us
    </tns:RequestSOAPHeader>
</soapenv:Header>
```

Step 6: Data Type Definations

IdentityType enumeration

List of IdentityType values.

Enumeration	Description
1000	Customer
2000	SPOperator
3000	OrganizationOperator
5000	Organization
6000	Till
8000	SP

IdentifierType enumeration

List of IdentityType values.

Enumeration	Description
1	MSISDN
2	TillNumber
3	SPShortCode
4	OrganizationShortCode
5	IdentityID
6	O2CLink
9	SPOperatorCode
10	POSNumber
11	OrganizationOperatorUserName
12	OrganizationOperatorCode
13	VoucherCode

ParameterType structure

Element name	Element type	Optional	Description
Key	xsd:string	No	It indicates a parameter name.
Value	xsd:string	No	It indicates a parameter value.

Parameters structure

Element name	Element type	Optional	Description
Parameter	ParameterType[1..unbounded]	No	It is used to carry specific parameters for specific transaction or business operation.

ReferenceData structure

Element name	Element type	Optional	Description
ReferenceItem	ParameterType[1..unbounded]	No	It is used carry some reference data that MM need not analyze but need to record it into transaction log..

Transaction structure

Element name	Element type	Optional	Description
CommandID	xsd:string	No	The unique identifier of transaction/business operation. Max length is 64.eg • *SalaryPayment* • *BusinessPayment* • *BusinessPaymentWithWithdrawalChargePaid* • *SalaryPaymentWithWithdrawalChargePaid* • *PromotionPayment* • *TransferFromBankToCustomer*
LanguageCode	xsd:string	Yes	It indicates language. It's reserved.
OriginatorConversationID	xsd:string	No	The unique identifier of the request message generated by third party. It is used to identify a request between the third party and MM. Max length is 128. Field must start with the B2C organisation short and name of organisation. Eg. 232323_KCBOrg_XXXXXX XXXXX must be unique for every transaction.
ConversationID	xsd:string	Yes	The unique identifier generated by MM for a previous request message. It is used to support communication multi-times between the third party and MM for one operation/transaction.
Remark	xsd:string	Yes	The remark information about this operation. Max length is 255
EncryptedParameters	xsd:string	Yes	It is used to carry the value for the element Parameters which are encrypted. The value for this parameter should be a CDATA and encode with base64
Parameters	Parameters	Yes	It is used to carry specific parameters for specific transaction or business operation. If the element EncryptedParameters presents, this parameter should not present.
ReferenceData	ReferenceData	Yes	It is used carry some reference data that MM need not analyze but need to record it into transaction log.
Timestamp	xsd:string	No	The timestamp generated by the third party.

Caller structure

Element name	Element type	Optional	Description
CallerType	xsd:integer	No	Indicates the type of the caller: 2-APICaller 3-Other(Reserved)
ThirdPartyID	xsd:string	No	The unique identifier of a third party system defined in MM. It indicates the third party which initiates the request. Max length is 20
Password	xsd:string	Yes	This security credential of the ThirdPartyID defined in MM. If the password feature for third party is used in MM, then this parameter must be presented in the request message.
CheckSum	xsd:string	Yes	Currently it is unused. It is reserved for API security.
ResultURL	xsd:string	Yes	It indicates the destination URL where Broker should send the result message to.

Initiator structure

Element name	Element type	Optional	Description
IdentifierType	IdentifierType	No	It indicates the identifier type of the initiator. The value of this parameter must be a valid identifier type supported by MM.
Identifier	xsd:string	No	It indicates the identifier of the initiator. Its value must match the inputted value of the parameter IdentifierType.
SecurityCredential	xsd:string	No	It indicates the security credential of the initiator. Its value must match the inputted value of the parameter IdentifierType.
ShortCode	xsd:string	No	When the initiator is an organization operator, this parameter must be present in the request to indicate which organization the operator belongs to. If the initiator is not an organization operator, this parameter should not be present.

Step 7: The SOAP Request Body

A SOAP request can be made in the following ways:

Element name	Element type	Optional	Description
CommandID	xsd:string	No	The unique identifier of transaction/business operation. Max length is 64.eg • *SalaryPayment* • *BusinessPayment* • *BusinessPaymentWithWithdrawalChargePaid* • *SalaryPaymentWithWithdrawalChargePaid* • *PromotionPayment* • *TransferFromBankToCustomer*
LanguageCode	xsd:string	Yes	It indicates language. It's reserved.
OriginatorConversationID	xsd:string	No	The unique identifier of the request message generated by third party. It is used to identify a request between the third party and MM. Max length is 128. Field must start with the B2C organisation short and name of organisation. Eg. 232323_KCBOrg_XXXXXX XXXXX must be unique for every transaction.
ConversationID	xsd:string	Yes	The unique identifier generated by MM for a previous request message. It is used to support communication multi-times between the third party and MM for one operation/transaction.
Remark	xsd:string	Yes	The remark information about this operation. Max length is 255
EncryptedParameters	xsd:string	Yes	It is used to carry the value for the element Parameters which are encrypted. The value for this parameter should be a CDATA and encode with base64
Parameters	Parameters	Yes	It is used to carry specific parameters for specific transaction or business operation. If the element EncryptedParameters presents, this parameter should not present.
ReferenceData	ReferenceData	Yes	It is used carry some reference data that MM need not analyze but need to record it into transaction log.
Timestamp	xsd:string	No	The timestamp generated by the third party.

This part contains a number of parameters:

This is how the final request should look like:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:req="http://api-
v1.gen.mm.vodafone.com/mminterface/request">
  <soapenv:Header>
    <tns:RequestSOAPHeader xmlns:tns="http://www.huawei.com.cn/schema/common/v2_1">
      <tns:spId>35000011</tns:spId> //SPid, provided by us
      <tns:spPassword>c5216e519a071d601bedd150f3fcd026</tns:spPassword>  //Password, provided by us. Encrypted using
Base64(sha256(SPid+Password+Timestamp))
      <tns:serviceId>35000001000009</tns:serviceId> //Service id, provided by us
      <tns:timeStamp>20120101010101</tns:timeStamp> //Timestamp, in format yyyymmddhhmmss
    </tns:RequestSOAPHeader>
  </soapenv:Header>
  <soapenv:Body>
    <req:RequestMsg><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<request xmlns="http://api-v1.gen.mm.vodafone.com/mminterface/request">
  <Transaction>
    <CommandID>PromotionPayment</CommandID>  //The command id for a b2c. You can use PromotionPayment or BusinessPayment.
    <LanguageCode></LanguageCode>
    <OriginatorConversationID>459xxxxxxxxxxxxxxxxxxxxxxxxxxxxx6</OriginatorConversationID>  //This is unique to every request
    <ConversationID> </ConversationID>
    <Remark> </Remark>
        <Parameters>
      <Parameter>
        <Key>Amount</Key>
        <Value>1</Value> //Amount
      </Parameter>
      <Parameter>
        <Key>Key1</Key> //Constant variable, remains the same
        <Value>Value1</Value> //Constant variable, remains the same
      </Parameter>
    </Parameters>
    <ReferenceData>
      <ReferenceItem>
        <Key>QueueTimeoutURL</Key> // Your queue timeout url
        <Value>http://10.66.49.789:7888/new</Value>
      </ReferenceItem>
      <ReferenceItem>
        <Key>Occasion</Key>
        <Value>Jamuhuri</Value>
      </ReferenceItem>
    </ReferenceData>
    <Timestamp>2013-07-29T18:50:41.2109675Z</Timestamp> //Timestamp
  </Transaction>
  <Identity>
    <Caller>
      <CallerType>2</CallerType>//Constant variable, remains the same
      <ThirdPartyID> </ThirdPartyID>
      <Password>Password0</Password>//Constant variable, remains the same
      <CheckSum>CheckSum0</CheckSum>//Constant variable, remains the same
      <ResultURL>ResultURL0</ResultURL>//Constant variable, remains the same
    </Caller>
    <Initiator>
      <IdentifierType>11</IdentifierType>//Constant variable, remains the same
      <Identifier>username</Identifier>  //Provided by us
<SecurityCredential>SecurityCredential0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx==</SecurityCreden
tial> //Needs to be encrypted using the step 5.
      <ShortCode>100001</ShortCode>  //Shortcode for thebusiness
    </Initiator>
    <PrimaryParty>
      <IdentifierType>4</IdentifierType> //Constant variable, remains the same
      <Identifier>100001</Identifier> //Short code
      <ShortCode>100001</ShortCode>  //Short code
```

```
    </PrimaryParty>
    <ReceiverParty>
      <IdentifierType>1</IdentifierType>
      <Identifier>2547204789659</Identifier>  //MSISDN receiving the money
      <ShortCode>ShortCode1</ShortCode>   //Constant variable, remains the same
    </ReceiverParty>
    <AccessDevice>
      <IdentifierType>1</IdentifierType> //Constant variable, remains the same
      <Identifier>Identifier3</Identifier>//Constant  variable, remains the same
    </AccessDevice>
  </Identity>
  <KeyOwner>1</KeyOwner> //Constant variable, remains the same
</request>]]></req:RequestMsg>
  </soapenv:Body>
</soapenv:Envelope>
```

An expected response would be as below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  xmlns:req="http://api-
v1.gen.mm.vodafone.com/mminterface/request">
  <soapenv:Header/>
  <soapenv:Body>
    <req:ResponseMsg><![CDATA[<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://api-v1.gen.mm.vodafone.com/mminterface/response">
  <ResponseCode>ResponseCode0</ResponseCode>
  <ResponseDesc>ResponseDesc0</ResponseDesc>
  <ConversationID>
  </ConversationID>
  <OriginatorConversationID>
  </OriginatorConversationID>
  <ServiceStatus>0</ServiceStatus> //0 is success, anything else is fail. Check error codes in appendix.
</response>]]></req:ResponseMsg>
  </soapenv:Body>
</soapenv:Envelope>
```

Error Codes

Error code	Error Description	
0	Success	ApiResult
1	Insufficient Funds	ApiResult
2	Less Than Minimum Transaction Value	ApiResult
3	More Than Maximum Transaction Value	ApiResult
4	Would Exceed Daily Transfer Limit	ApiResult
5	Would Exceed Minimum Balance	ApiResult
6	Unresolved Primary Party	ApiResult
7	Unresolved Receiver Party	ApiResult
8	Would Exceed Maxiumum Balance	ApiResult
11	Debit Account Invalid	ApiResult
12	Credit Account Invaliud	ApiResult
13	Unresolved Debit Account	ApiResult
14	Unresolved Credit Account	ApiResult
15	Duplicate Detected	ApiResult
17	Internal Failure	ApiResult
18	Initiator Credential Check Failure	ApiResult
19	Message Sequencing Failure	ApiResult

20	Unresolved Initiator	ApiResult
21	Initiator to Primary Party Permission Failure	ApiResult
22	Initiator to Receiver Party Permission Failure	ApiResult
23	Request schema validation error	ApiResponse
24	MissingRequestParameters	ApiResponse
25	InvalidRequestParameters	ApiResponse
26	SystemTooBusy	ApiResponse
0	Success	ApiResponse
100000000	Request was cached, waiting for resending	ApiResponse
100000001	The system is overload	ApiResponse
100000002	Throttling error	ApiResponse
100000003	Exceed the limitation of the LICENSE	ApiResponse
100000004	Internal Server Error	ApiResponse
100000005	Invalid input value:%1 %1 indicates the parameter's name.	ApiResponse
100000006	SP's status is abnormal	ApiResponse
100000007	Authentication failed	ApiResponse
100000008	Service's status is abnormal	ApiResponse
100000009	API's status is abnormal	ApiResponse
100000010	Insufficient permissions	ApiResponse
100000011	Exceed the limitation of request rate	ApiResponse
100000012	Insufficient balance	ApiResponse
100000013	No route	ApiResponse
100000014	Missing mandatory parameter:%1 %1 indicates the parameter's name.	ApiResponse
28	InitiatorAllowedOperationCheckFailure	ApiResult
29	InvalidCommand	ApiResult
30	ErrorSerializingRequest	ApiResponse
31	InitiatorNotSpecified	ApiResult
32	ErrorSerializingRequest	ApiResult
33	PrimaryPartyNotSpecified	ApiResult
34	PrimaryPartyIdentifierInvalid	ApiResult
35	ReceiverPartyNotSpecified	ApiResult
36	ReceiverPartyIdentifierInvalid	ApiResult
37	MissingApiCommand	ApiResult
38	InvalidConversationId	ApiResult
39	UnknownConversationId	ApiResult
40	InvalidParameterDefinition	ApiResult
41	DuplicateConversationDetected	ApiResult
42	DuplicateStageMessageDetected	ApiResult
43	AwaitingConfirmation	ApiResult
44	InitiatorToReceiverPartyPermissionFailure	ApiResult
45	InternalErrorDuringFinancialTransaction	ApiResult
46	ConfirmationReceived	ApiResult
47	RejectionReceived	ApiResult
48	OperationPermissionFailure	ApiResult
49	NoTransactionFound	ApiResult
50	InitiatorOrgIdentityStatusFailure	ApiResult
51	DebitChargeAccountInvalid	ApiResult
52	WouldExceedMaximumSingleAirtimePurchase	ApiResult

Response Codes

ResponseCode	ResponseDesc
000000000	Success

100000001	The system is overload
100000002	Throttling error
100000003	Exceed the limitation of the LICENSE
100000004	Internal Server Error
100000005	Invalid input value:%1 %1 indicates the parameter's name.
100000006	SP's status is abnormal
100000007	Authentication failed
100000008	Service's status is abnormal
100000010	Insufficient permissions
100000014	Missing mandatory parameter:%1 %1 indicates the parameter's name.

Command ID's

Below is a list of all the command ID's that you can use for the integration:

BusinessPayment
BusinessPaymentWithWithdrawalChargePaid
PromotionPayment
SalaryPayment
SalaryPaymentWithWithdrawalChargePaid
TransferFromBankToCustomer
DisburseFundsToBusiness
BusinessTransferFromMMFToUtility
BusinessTransferFromUtilityToMMF
MerchantToMerchantTransfer
MerchantTransferFromMerchantToWorking
MerchantTransferFromWorkingToMerchant
BusinessToBusinessTransfer
SalaryPaymentUnregisteredUser
PromotionPaymentUnregisteredUser
BusinessPaymentWithWithdrawalChargePaid
SalaryPaymentWithWithdrawalChargePaid
TransferFromBankToCustomer
MMFB2C

Finalizing the integration

1. Once the above is carried out on http (refer to step 3.1)

2. Carry the above out on https (refer to step 3.2), send signed off UAT's

3. Once testing is completed, the person assisting you will help you move to production.

FAQ's

1. Who initiates the transaction on B2C?
   You as the business initiate the transactions and check if you get a response result.
2. Do we require having a https connection?
   Yes, as a compulsory requirement you do require a https after testing on http and before moving to the live platform
3. What is different on the live environment than the testing?
   Everything is replicated on both platforms, the only thing different is you will be creating your own credentials and will have access to an M-Pesa web interface. You will not be provided the API credentials by us.
4. What is the easiest way to ask for support?
   You can get in touch on hangouts or skype with any of our teams that are assisting you.
5. Is there anything that can be changed on the API?
   Unfortunately, since a lot of businesses are connected to the API, changes will be difficult as it will affect all the users.
6. Is there a way I can download all the files that I will require in the tests?
   Yes, all the files that you will require apart from the .p7b's will be available at the following link: https://drive.google.com/folderview?id=0B_BRYv-DOjeTamo3TFVBX2VpcTQ&usp=sharing
7. Does the format of the request and response need to be the same as the .wsdl you sent us?
   Yes, the format should remain the same. Any change will result in an error result.
8. When should we start testing?
   You should start testing a month or more in advance with our teams so that any errors encountered can be quickly dealt with.