TPAY™ MOBILE

TPAY – Subscription with Parking Period – v 2.0

1.1 Integration Scope

To Integrate “Subscription with Parking Period flow” you need to implement:

1. Add Subscription Contract: Which Creates a Subscription at TPAY side, forming the relation between Subscription Plan, Catalog and Product.
2. Verify Subscription Contract: Which verifies the Subscription created from the first call (by default verification using Pin Code, discussed below and its mandatory to be implemented).

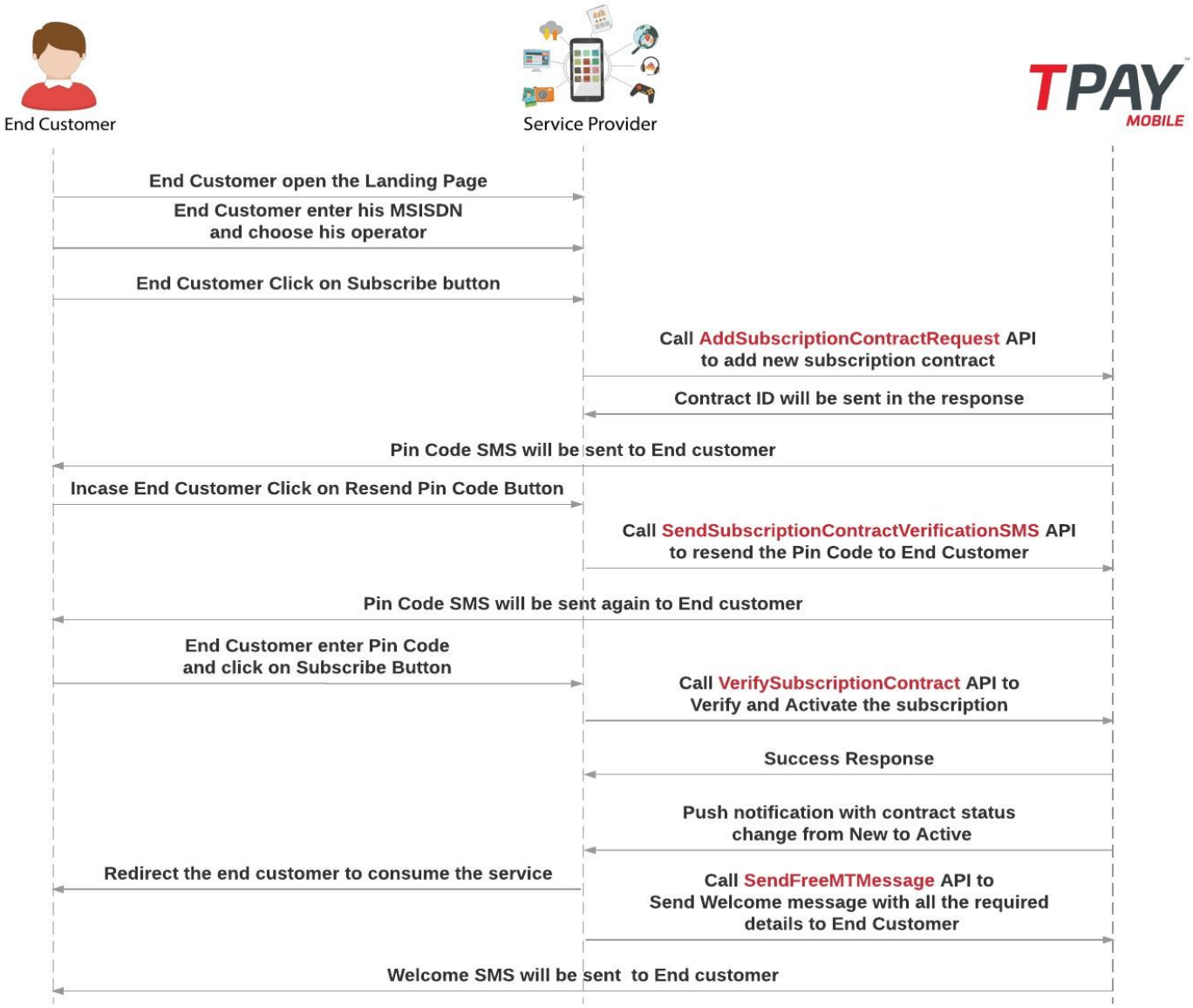
1.2 Integration Estimates

To Integrate the above-mentioned flow, Integration should take maximum 5 working days

2 Subscription with Parking Period – Verification Ways

2.1 Default Flow (Pin Code Verification flow)

End customer will enter his MSISDN and receive Pin Code to subscribe to the service. Partner will verify the subscription contract using Pin Code by providing the below when calling “*AddSubscriptionContractRequest”*:



3 APIs Call Flow

3.1 Add Subscription Contract

Web Reference

REST: http://live.TPAY.me/api/TPAYSubscription.svc/Json/AddSubscriptionContractRequest

Operation name

AddSubscriptionContractRequest

Description

This function initializes a new subscription contract at TPAY engine which is a contract between subscription plan and a certain product in a catalog over a specific time span.

Note: All fields in the Request are Case Sensitive (first letter is small and all other following words are starting with capital letter (for ex: customerAccountNumber).

Fields

Order	Field Name	Data Type	Mandatory / Optional	Description
0	signature	string	Mandatory	How to Calculate Add Subscription Contract Digest?
1	customerAccountNumber	string	Mandatory	The Customer Identifier shared between Partner and TPAY. Its value could be anything.
2	msisdn	string	Mandatory	The MSISDN of the customer
3	operatorCode	String	Mandatory	The mobile network operator code (mobile country code (mcc) + mobile network code (mnc)). http://en.wikipedia.org/wiki/Mobile_Network_Code
4	subscriptionPlanId	long	Mandatory	The Id of the subscription plan that will be linked to this contract
5	initialPaymentproductId	string	Mandatory	The SKU of the product id that defines the initial payment amount. Products and product catalogs can be managed using the TPAY selfmanagement portal.
				Should be set to initiate the recurring in the same day.
6	initialPaymentDate	string	Mandatory	The specific date time stamp that the initial payment should take place in. It should be “contractStartDate+30 days” It’s GMT time zone and should be in this Format “yyyy-MM-dd hh:mm:ssZ”

7	executeInitialPaymentNow	bool	Mandatory	Indicates whether TPAY should execute the initial payment charging request immediately or not. Its value should be “false”.
8	recurringPaymentproductId	string	Mandatory	The SKU of the product id that defines the recurring payment amount. Products and product catalogs can be managed using the TPAY selfmanagement portal.
9	productCatalogName	string	Mandatory	The name product catalog in which all the mentioned products in this contract should be part of it.
10	executeRecurringPaymentNow	bool	Mandatory	Indicates whether TPAY should execute the Recurring payment charging request immediately or not. Its value should be “false”.
11	contractStartDate	string	Mandatory	Defines the contract activation date. It should be in the same day of the subscription. The time is GMT and it should be in this Format “yyyy-MM-dd hh:mm:ssZ”
12	contractEndDate	string	Mandatory	Defines the contract Expiration date.
13	autoRenewContract	bool	Mandatory	Indicates whether the contract should be renewed automatically
				after the end contract date is passed or the contract status should be changed to “Expired”. It’s a Boolean should be “true or false”

14	Language	MerchantLanguage	Mandatory	Indicates the contract language which affects SMS notifications to be sent to the customer. It's type is Enum, use the below value for the language: `Auto = 0, English = 1, Arabic = 2, French = 3`
15	sendVerificationSMS	bool?	Mandatory	Should be true in case of pin code verification so the pin code will be sent to the customer
16	allowMultipleFreeStartPeriods	bool?	Mandatory	If set to false, it will reject any new subscriptions for an old active contract with free start period with the given MSISDN. (contracts whose initial payment date after contract start date) Should be "true".
17	headerEnrichmentReferenceCode	string	Optional	Indicate the reference code of a verified customer within 4 minutes from the request time. The contract will be activated as soon as the operation succeeded. Note: You can verify the msisdn using header enrichment verification. "Check TPAY -Header Enrichment " Integration Guide
18	smsId	string	Optional	It's the Id of the SMS when the customer is verified through MO Verification

In case of success, the operation directly returns SubscriptionContractResponse object containing the following data:

Response Fields

Field Name	Mandatory / Optional	Description
operationStatusCode	Mandatory	In case of success, this should have the value of zero which indicates "Success", otherwise it will contain the value of 51 which
		indicates "Error" and the error details will be listed in "ErrorMessage" parameter.
subscriptionContractId	Mandatory	The ID of the newly created subscription contract.

paymentTransactionStatusCode	Optional	In case of success, this should have the value of "PaymentCompletedSuccessfully " all Status Codes
transactionId	Optional	The Transaction Id that can be used to confirm the payment transaction in case of sending verification pin code to the customer, this parameter will contain value in case of a successful transaction for Initial Payment or after executing the recurring charging while creating the contract. If the customer requires resending verification pin code, you can call ResendVerificationPin It will be null.
nextPaymentDate	Mandatory	The Date of the next recurring payment transaction as planned by the linked subscription plan configuration.
errorMessage	Optional	This parameter will contain the error details if any.

Signature / Digest Calculation

Signature is a security code that is calculated, by hashing all the request fields mentioned above, by concatenating all of them in the same order (In Order's column, which we refer to it as "message"). And using the "Private Key" shared by TPAY Team in Hashing all these data as the Hash key. And adding to it the "Public Key" also shared by TPAY Team.

To calculate the message:

message = customerAccountNumber + msisdn + operatorCode + subscriptionPlanId +

initialPaymentproductId + initialPaymentDate + executeInitialPaymentNow.ToString().ToLower() +

recurringPaymentproductId + productCatalogName +

executeRecurringPaymentNow.ToString().ToLower() + contractStartDate + contractEndDate +

autoRenewContract.ToString().ToLower()+

(language.HasValue ? ((int)language.Value).ToString() : string.Empty) +

(sendVerificationSMS.HasValue ? sendVerificationSMS.Value.ToString().ToLower() : string.Empty)+
(allowMultipleFreeStartPeriods.HasValue ? allowMultipleFreeStartPeriods.Value.ToString().ToLower() : string.Empty) +
headerEnrichmentReferenceCode + smsId;

signature = Public Key + ":" + HexString(HMACSHA256(Private Key, message))

Note: any of the Optional fields can be neglected when calculating the digest, but you should keep the same order.

You can try calculating Signature from this URL: https://dotnetfiddle.net/uGfdWY

AddSubscriptionContractRequest API Response – error message

Error Message	Case
Invalid Digest	Public key is not exist or digest is not calculated correctly
Invalid Start Or End Date	Invalid Start Or End Date, year should be greater than 2000
Contract Start Date Can't Be Before Today	Contract Start Date Can't Be Before Today, contract start date should be in UTC format or UTC+2
Contract Start Date Can't Be After End Date	Contract Start Date Can't Be After End Date
The difference between contract start and end dates can't be less than a single recurring cycle	The difference between contract start and end dates can't be less than a single recurring cycle, contract end date should be a year or more from contract start date
Initial Payment Date Can't Be Before Contract Start Date	Initial payment date should not be before Contract start date, it should be the same or after it
Invalid Operator	operatorCode(MCC+MNC) is invalid and if operator code is null
Invalid Product Id	Product SKU is not correct or null in both (initial or recurring)
Invalid Catalog Or Product	invalid Catalog Name or Product Id, Which means that this catalog doesn't have price on this product
Please enter your phone number.	No Msisdn
Please enter valid phone number for the selected mobile operator.	Msisdn contains non-digit values or regex not match

AddSubscriptionContractRequest Sample

Request:

{"signature":"ISxESI0TsIjxDTR7yXMB:06ad851b8639b5322bb9a547ab32bd75a12cc1377fed08bfd4aa04a8fb2145ca","cust

omerAccountNumber":"testcustomer","msisdn":"201069409370","operatorCode":"60202","subscriptionPlanId":40453,"i nitialPaymentproductId":"Puzzle_game","initialPaymentDate":"2017-06-21 16:18:42Z","executeInitialPaymentNow":false,"recurringPaymentproductId":"Puzzle_game","productCatalogName":"Gam esZone","executeRecurringPaymentNow":false,"contractStartDate":"2017-06-21 16:18:42Z","contractEndDate":"2018-06-21 16:18:42Z","autoRenewContract":true,"language":2,"sendVerificationSMS":true,"allowMultipleFreeStartPeriods":null,"he aderEnrichmentReferenceCode":"","smsId":""}

Reponses:

{"operationStatusCode":0,"subscriptionContractId":340510,"paymentTransactionStatusCode":-1,"transactionId":null,"nextPaymentDate":"2017-06-21 16:18:42Z","errorMessage":null,"subscriptionContractStatus":1}

3.2 Verify Subscription contract

Web Reference

REST: http://live.TPAY.me/api/TPAYSubscription.svc/Json/VerifySubscriptionContract

Operation name

VerifySubscriptionContract

Description

This call should follow Add Subscription Contract Call, as Add Subscription Contract sends Pin code to end customer, and initiates subscription contract at TPAY system

Note: All fields in the Request are Case Sensitive (first letter is small and all other following words are starting with capital letter (for ex: customerAccountNumber)

Fields

Order	Field Name	Data Type	Mandatory / Optional	Description
0	Signature	string	Mandatory	A security code that is calculated by hashing all the request parameters using Partner's private key. signature = merchantPublicKey + ":" + HexString(HMACSHA256(merchantPrivateKey, message)) message = subscriptionContractId + pinCode
1	subscriptionContractId	long	Mandatory	The ID of the customer's subscription contract.

2	pinCode	string	Mandatory	The PIN code entered by the customer.

In case of success, the operation directly returns SubscriptionContractVerificationSMSResponse object containing the following properties:

Response Fields

Field Name	Mandatory / Optional	Description
operationStatusCode	Mandatory	In case of success, this should have the value of zero which indicates "Success", otherwise it will contain the value of 51 which indicates “Error” and the error details will be listed in ErrorMessage parameter.
subscriptionContractId	Mandatory	The ID of the subscription contract.
errorMessage	Optional	This parameter will contain the error details if any.

Signature / Digest Calculation

Signature is a security code that is calculated, by hashing all the request fields mentioned above, by concatenating all of them in the same order (In Order’s column, which we refer to it as “message”). And using the “Private Key” shared by TPAY Team in Hashing all these data as the Hash key. And adding to it the “Public Key” also shared by TPAY Team.

To calculate the message: message = subscriptionContractId + pinCode;

signature = Public Key + ":" + HexString(HMACSHA256(Private Key, message))

Note: any of the Optional fields can be neglected when calculating the digest, but you should keep the same order.

You can try calculating Signature from this URL: https://dotnetfiddle.net/M6MIid

VerifySubscriptionContract API Response – error message

Error Message	Case
Invalid Digest	Public key is not correct/exist or digest is not calculated correctly(ResponseCode: 51), Also when one the parameters are not correct or missing
Invalid Subscription Contract Id	Invalid Subscription Contract Id (ResponseCode: 201), when subscription contract Id is null
SMS Not Sent	SMS Not Sent (ResponseCode: 303), When pin code SMS is not sent to customer
Subscription Contract Is Already Verified	Subscription Contract Is Already Verified (ResponseCode: 202), When the subscription contract status is Active

Exceed Max Attempt Reached Of Invalid Pin	Exceed Max Attempt Reached Of Invalid Pin (ResponseCode: 305), When customer exceed the limit of entering wrong pin code which is 5 times
Subscription Contract Can't Be Activated	Subscription Contract Status is not "New" (ResponseCode: 204)
Subscription Contract Is Already Active	Subscription Contract Status is Active
You have exceeded maximum invalid PIN trials attempt. Please close the widget, refresh the provider's page then relaunch the T-Pay widget again.	invalid pin & Exceed max number of invalid pin retrials for payment transaction
Invalid Pincode	Invalid Pincode (ResponseCode: 302)
Error	Exception (ResponseCode: 51)

VerifySubscriptionContract Sample

Request:

{"signature":"ISxESI0TsIjxDTR7yXMB:30bae38de4a1e2fe0f31a6b43c04d4536bc751482712245ed8bf18ba634dbe42","sub scriptionContractId":"340510","pinCode":"786340","transactionId":""}

Response:

{"operationStatusCode":0,"subscriptionContractId":340510,"errorMessage":null,"responseCode":0}

5 How to Calculate Digest (Using Hashing Algorithms)

- Digital signature techniques are used to avoid data manipulation & web parameters tampering for the communicated web messages over internet
- Create a message containing all your parameters concatenated to each other
- Apply hashing using the private key to generate the digital signature using a proved hashing algorithm (SHA 256, SHA 512)
- Public key is added to the hashed message to identify the sender of the request

C# Example

Message: concatenating all parameters in correct order

Digest function:

```
public static string CalculateDigest(string publicKey, string privateKey, string message)
{
var digest = "";
var hash = new System.Security.Cryptography.HMACSHA256(System.Text.Encoding.UTF8.GetBytes(privateKey));
```

```
var correctHash = string.Join(string.Empty, hash.ComputeHash(System.Text.Encoding.UTF8.GetBytes(message)).Select(b
=> b.ToString("x2")));
digest = publicKey + ":" + correctHash;
return digest;
}
```