Strictly Confidential

BillDesk Payment Gateway

SKODA AUTO

-- EmailPay API Interface Document v1.1

IndiaIdeas.com Limited

Table of Contents

1.0. Introduction

BillDesk EmailPay is a solution, which is used to send out the payment link to the Customer via e-mail/SMS for a specific product / service.

EmailPay solution enables a merchant to create and send the Customer's the Payment Link in the form of an e-mail/SMS into the Customer's inbox.

Key aspects of the solution –

- Merchant can initiate a call to BillDesk to initiate an email/SMS to customer for making the payment.
- The e-mail/SMS Link is sent via secure channel to the Customer's email account/ phone by BillDesk
- Enables Customer to make online, real time payment towards their payments to the merchant using BillDesk payment gateway.

2.0. Merchant Requirement

- Merchant would like to interface with the BillDesk platform, via APIs, so that the Email/SMS with the payment link is sent by BillDesk to the customer.

3.0. Solution

- For this specific requirement, merchant can integrate with the **API based EmailPay** workflow.
- Merchant can fill in specific details from customer on their page and then invoke the API (containing the relevant details) to BillDesk.
- Based on the API request received BillDesk will generate the payment link and send an email/SMS to the end customer with a unique payment link.
- Using this payment link, Customer can make a payment for a specific amount using BillDesk Payment Gateway.

4.0. Process Flow

- Merchant defines the fields required on the form, Email Template, Preview Page template and the Response page templates.

- BillDesk creates a set up basis these inputs and provides the UDF mapping and a redirection link i.e. Post URL to the merchant.

- Merchant will pass the required value/s as required in the API request to the BillDesk URL.

- If BillDesk receives a valid request, then it will send an email/SMS to the end customer with a unique payment link.

- When the Customer clicks on the 'Proceed to Pay' button / Payment link in the email received, BillDesk system will check if a successful payment is already made against this order or not. If found that the payment is still pending, the next page will display the details (non-editable) related to this transaction including the transaction amount. These details will be same details as entered by the dealer on

the Skoda's internal platform which in turn triggers the API to BillDesk with information on these fields.

- Customer proceeds and is routed to the BillDesk payment options page where s/he chooses the preferred payment option and proceeds to complete the payment.

- After completing the Payment Process, Customer will be routed to the Transaction Acknowledgement page (displayed by BillDesk) with Success/Failure message and other details such as Transaction Reference number, Transaction Date, Amount etc.

- Also BillDesk will send a Server to Server ('S2S') response [for every completed transaction] to a Skoda designated URL specified upfront at the time of the integration. Please refer below section for S2S response parameter specifications.

- The transaction reference number is the unique reference number generated by BillDesk for each case and can be referred to for any queries with BillDesk.

- BillDesk Transaction Reference number will also be a part of the MIS, which will be shared on the subsequent day for all the Successful transactions that will be done by the Customers using the Payment link received via email.

5.0. API Definition

This API is used by the merchant to initiate the request to BillDesk to trigger SMS/email to customer.

REQUEST:

URL: https://uat.billdesk.com/MercOnline/EmailPayController [UAT URL]
https://payments.billdesk.com/MercOnline/EmailPayController [Production URL]

- Merchant will initiate request to BillDesk with three parameters to above URL

Parameter	Values	Details
reqid	preSaveCampaignForm	This will be a fixed value
campaignValues	Merchant needs to pass this parameter as Base64 encoded JSON	Details explained below
sendmail	True	This will be a fixed value

Details for campaignValues:

- Merchant needs to pass this parameter as Base64 encoded JSON.
- The following steps will be involved for creating this parameter.

Step 1: Generate Checksum for UDF parameters shared by customer.

- Based on the information provided by customer on merchant website, the data will be mapped to specific UDF in a pre-decided format as agreed between Merchant and BillDesk.

- Please note that the values of UDF while computing checksum have to be in order [UDF1 – 15 whichever is applicable].

- To generate checksum for the UDF fields, use the fields in below format: fieldname|value|fieldname|value|...|Key

  i.e. the fields should be "pipe (|)" separated when generating checksum.

Step 2: Creating API request in JSON format along with checksum

- The checksum generated in step 1, needs to be entered in JSON request as given below as parameter "checksum"

Details of API Request field specification:

Parameter	Sample Value	Description	Datatype	Field Type
MessageCode	1023	Will be provided by BillDesk. It will be a fixed value.		Mandatory
TraceID	ABCD00001	Request Unique Reference No. of the merchant for each request	Alphanumeric	Mandatory
Time stamp	20160624154123	Request timestamp in yyyymmddhhmmss format	Numeric	Mandatory
campaignId	590	Will be provided by BillDesk. It will be a fixed value		Mandatory
campaignCode	JSKDHFSJ213244KJH45JK3N5M3	Will be provided by BillDesk. It will be a fixed value		Mandatory
fieldName	UDF 1	Mapping will be provided by BillDesk basis fields defined for the Form		Mandatory
value	500.00	Value as required to be prepopulated against the UDF on the frontend	As defined at the time of form creation	Mandatory
display	fixed	Value should be passed as 'fixed'		Mandatory
checksum	DJKFSDHFK348239408	To be computed dynamically for every request as per the description below	Alphanumeric	Mandatory

A sample JSON is given below:

```
{
"messageCode": "1023",
"traceId": "ABC0001",
```

```
"timestamp": "20160624154123",
"campaignId": "123",
"campaignCode": "JSKDHFSJ213244KJH45JK3N5M3",
"fields": [{
    "fieldName": "UDF 1",
    "value": "value1",
    "display": "display value"
}, {
    "fieldName": "UDF 2",
    "value": "value2",
    "display": "display value"
}, {
    "fieldName": "UDF 3",
    "value": "value3",
    "display": "display value"
}, {
    "fieldName": "UDF 4",
    "value": "value4",
    "display": "display value"
}, {
    "fieldName": "UDF 5",
    "value": "value5",
    "display": "display value"
}, {
   ],
     "checksum":"PQRSM123K3482AHY08"
     }
```


Step 3: Convert the JSON string to Base64 encoded format.

Example

The fields entered by customer on merchant website is as follows:

a. Vehicle Identification number – MX12K1234 (Mapped to UDF1)
b. Plan – Service1 (Mapped to UDF2)
c. Amount – 5000.00 (Mapped to UDF3)

Step 1: Generate Checksum for UDF parameters shared by customer

- The Checksum would be generated using UDF values as given below:
  UDF 1| MX12K1234|UDF 2| Service1|UDF 3|5000.00|[Key]

- Suppose the checksum output generated above has value:
  DJKFSDHFK348239408

Step 2: The JSON request will be created as given below:

```
{
    "messageCode": "1023",
    "traceId": "ABC0001",
    "timestamp": "20160624154123",
    "campaignId": "123",
    "campaignCode": "JSKDHFSJ213244KJH45JK3N5M3",
    "fields": [{
        "fieldName": "UDF 1",
        "value": "MX12K1234",
        "display": "fixed"
    }, {
        "fieldName": "UDF 2",
        "value": "Service1",
        "display": "fixed"
    }, {
        "fieldName": "UDF 3",
        "value": "5000.00",
        "display": "fixed"
    }],
    "checksum":"DJKFSDHFK348239408"
}
```

Step 3: Convert the JSON string to Base64 encoded format.

The output of the file will be as given below:

eyJtZXNzYWdlQ29kZSI6ICIxMDAyIiwidHJhY2VJZCI6ICJBQkMwMDAxIiwidGltZXN0YW1wIjoIj
IwMTYwNjI0MTU0MTIzIiwiY2FtcGFpZ25JZCI6ICIyNDEiLCJjYW1wYWlnbkNvZGUiOiAicnE1Z1
U3QXZPVENMMHJ4RzQzQ1pxZ2RHRi9GZ2RibFBoVUdDR0pXanhkTT0iLCJmaWVsZHMiOiBbe
yJmaWVsZE5hbWUiOiAiVURGIDEiLCJ2YWx1ZSI6ICJSYWNoYW5hIFdhZ2hlbGEiLCJkaXNwbG
F5IjogImZpeGVkIn0seyJmaWVsZE5hbWUiOiAiVURGIDMiLCJ2YWx1ZSI6ICIxMjM0NTY3ODk
wIiwiZGlzcGxheSI6ICJmaXhlZCJ9LHsiZmllbGROYW1lIjogIlVERiA0IiwidmFsdWUiOiAiMSIsIm
Rpc3BsYXkiOiAiZml4ZWQifV0sImNoZWNrc3VtIjoiOWYzMzY4OTM4ZTdiM2RmYWNkNWIyM2
RjY2I2NWViNWMifQ=

Request to be sent to BillDesk:

https://uat.billdesk.com/MercOnline/EmailPayController?reqid=preSaveCampaignForm&sen
dmail=True&campaignValues=eyJtZXNzYWdlQ29kZSI6ICIxMDAyIiwidHJhY2VJZCI6ICJBQk
MwMDAxIiwidGltZXN0YW1wIjogIjIwMTYwNjI0MTU0MTIzIiwiY2FtcGFpZ25JZCI6ICIyNDEiLCJ
jYW1wYWlnbkNvZGUiOiAicnE1Z1U3QXZPVENMMHJ4RzQzQ1pxZ2RHRi9GZ2RibFBoVUdDR0p
XanhkTT0iLCJmaWVsZHMiOiBbeyJmaWVsZE5hbWUiOiAiVURGIDEiLCJ2YWx1ZSI6ICJSYWNo
YW5hIFdhZ2hlbGEiLCJkaXNwbGF5IjogImZpeGVkIn0seyJmaWVsZE5hbWUiOiAiVURGIDMiLC
J2YWx1ZSI6ICIxMjM0NTY3ODkwIiwiZGlzcGxheSI6ICJmaXhlZCJ9LHsiZmllbGROYW1lIjogIlV
ERiA0IiwidmFsdWUiOiAiMSIsImRpc3BsYXkiOiAiZml4ZWQifV0sImNoZWNrc3VtIjoiOWYzMzY4
OTM4ZTdiM2RmYWNkNWIyM2RjY2I2NWViNWMifQ=

RESPONSE:

IF REQUEST IS VALID:

Response:
<Payment link>

Sample:
https://uat.billdesk.com/MercOnline/EmailPayController?reqid=makePayment&campaignId=1107&merchantId=241&customerId=2921

IF REQUEST IS INVALID:

Response:
<Error Message>

Sample:
Json validation failed

6.0. S2S response to Merchant:

As per merchant request, BillDesk will send an S2S response to the Merchant designated URL as a parameter – msg

Payment Updation process at Merchant end

The following process should be followed at Merchant end for receiving and processing the payment response:

(a) Receive and Read the Payment Response message – msg in S2S response.
(b) Generate the 'checksum value' for the Payment Response and validate it with the 'checksum value' received in the Payment Response. If they match; proceed to step (c) below; else reject the payment response.
(c) Update the original record in the merchant system based on the 'AuthStatus' field received in the Payment Response. Refer the table below for various values that are received in the AuthStatus field, and the related Transaction Status. The updation to the original record must be done as follows:

Successful transaction [AuthStatus – 0300]
Update <record> set STATUS = 'SUCCESS' where ORIGINALSTATUS='PENDING' and ORDERNUMBER='1073234' and TRANSACTIONAMOUNT='100.00'

Failure transaction [AuthStatus – other than 0300]
Update <record> set STATUS = 'FAILURE' where ORIGINALSTATUS='PENDING' and ORDERNUMBER='1073234' and TRANSACTIONAMOUNT='100.00'

(d) The above updation process ensures the following:
- ☐ Only the original record is updated [through the Unique Order Number]
- ☐ The record is updated only once [for original status=PENDING]

- ☐ The record is updated for the same 'Transaction Amount' that was initiated by the merchant.

Response Message description:

MerchantID|CustomerID|TxnReferenceNo|BankReferenceNo|TxnAmount|BankID|BankMerchantID|TxnType|CurrencyName|ItemCode|SecurityType|SecurityID|SecurityPassword|TxnDate|AuthStatus|SettlementType|AdditionalInfo1|AdditionalInfo2|AdditionalInfo3|AdditionalInfo4|AdditionalInfo5|AdditionalInfo6|AdditionalInfo7|ErrorStatus|ErrorDescription|CheckSum

Sample Response Message

ABCD|PS123456789|MSBI0412001668|NA|94.00|SBI|NA|01|INR|DIRECT|NA|NA|NA|29-05-2017 16:08:56|0300|NA|NA|NA|NA|NA|NA|NA|NA|NA|NA|
AB6VN3245B66FE9511DB2A854AAA32ADC563E789CF213CA19E274F18F330G547

Refer 'AuthStatus' field received in the Payment Response to check the status of the transaction. Refer the table below for various values that are received in the AuthStatus field, and the related Transaction Status:

Authorization status

AuthStatus	Status Reason	Proposed Transaction Status
"0300"	Success	Successful Transaction
"0399"	Invalid Authentication at Bank	Cancel Transaction
"NA"	Invalid Input in the Request Message	Cancel Transaction
"0002"	BillDesk is waiting for Response from Bank	Pending Transaction
"0001"	Error at BillDesk	Cancel Transaction

For all AuthStatus that is not a Success, an ErrorDescription will be provided in the Payment Response.

However, it is important to note:

1. The Server-to-Server response handling must be agnostic of the HTTP GET/POST method at Merchant's end.

2. There should be no prefixed parameter appended to this URL that Merchant will provide BillDesk for setting up for the server-to-server direct response

To be able to setup the Server-to-Server direct response mechanism BillDesk would require the following information from Merchant:

1. Merchant Server to Server Direct Response URL

2. Underlying static Public IP address [based on the direct response URL] for setting up of network/ firewall rule at BillDesk end.

If need be, Merchant may allow the following BillDesk IP address at its end so that the Server to Server direct response sent by BillDesk could be accepted.

BillDesk IP Address: 210.210.24.74

It is highly recommended that the server-to-server responses sent by BillDesk are logged for about a week and are checked against the transaction status updated in the Merchant system.

7.0. Next Steps:

To be able to setup the Email Pay service we would require the following details:

- Fields and its specifications which will be passed in the API and are required as a part of Email / Reports
- Email Template – for sending Payment Link
  - Email Sender Name
  - Email Subject Line
  - Email Body and Footer
- Response page Templates - Success and Failure both