Payment Gateway Seamless Integration S2S

Project Guide –V 1.1

www.signetpay.com
Submitted By: Dasshpe Team
Version: Version 1.1
Date: July 1, 2020

Signet Payments Solution Pvt Ltd

Contents

ABOUT THIS GUIDE

Objectives and target audience

This guide is designed to provide detailed information on Dasshpe Payments Gateway Complete code architecture and design as well as how to connect and use Dasshpe Payment Gateway and Dasshpe ecommerce service by integrating on Merchant Website. The guide covers the steps in the payment process and the information that needs to be passed from Merchant web servers to Dasshpe, to enable Dasshpe to process payments.

Additional gateway integration options are also described.

Test Sever Transaction Url:
https://uat.dasshpe.com/crm/jsp/merchantpay
Production Server Transaction Url:
https://secure.dasshpe.com/crm/jsp/merchantpay

Related documentation

You should use this guide together with the additional Dasshpe Payment Gateway documents described below.

Guide Description	Integration Document Payment Gateway Integration Document V1.0

Conventions used in this guide

Convention Description
The table below lists some of the conventions used in this guide.
Table 1: List of conventions

Reference	Indicates a reference to another section in this guide. For Example, refer to the **Introduction**
File path	Used to indicate a file path or folder structure.
Glossary	Glossary term

INTRODUCTION

The Dasshpe Payment Gateway is a secured website, where you redirect customers from your Website/Ecommerce/Mobile platform to make a payment using Credit Card/Debit Card/Internet Banking other payment options. The gateway collects customer payment details in a secured manner using standard HTML forms and processes the payment transaction.
After the payment is complete, the customer is returned to your website and you receive a real-time notification of the payment, which include details of the transaction

PRE-REQUISITE

It is expected that the users may go through the entire guide to understand the Integration Requirements though it is fairly easy for people with technical understanding.

It is assumed that the Merchant doesn't have any specific business need for capturing the Customer's Card Information on their website as additional regulatory requirements of having PCI DSS certification is mandatory for capturing Customer's Credit/Debit/Net banking information on Merchant websites.

All Card/Net banking information is captured seamlessly on Dasshpe Payment Gateway Page in a secured manner and transaction response is returned back to the Merchant real time post processing of the transaction.

Contact us for queries

For all support queries, contact the Merchant Services department:
Email: care@Dasshpe.com

Dasshpe Payment Gateway in a Glance

Connecting to the Dasshpe Payment Gateway requires adding Dasshpe as a payment method on your website’s checkout or

payment page. When your customer selects the payment option, you should ensure that they are redirected to the Dasshpe Payment Gateway. At the same time you will need to submit information about the payment, such as your Merchant ID, amount to be paid and few other required parameters.
You can use a standard Post form to collect and pass payment and customer details to Dasshpe.

Integration with Dasshpe Payment Gateway – PG Hosted integration

Request Format

Please review the table below for details of the required and optional parameters that need to be included in your form. An example of a simple HTML form is provided.

Table 1: Dasshpe Gateway parameters(* These fields are required)

Field name	Description	Required	Type*	Min	Max	Example
Merchant Details						
PAY_ID*	Pay ID is provided by Dasshpe (Mandatory)	YES	NU	16	16	1602345784521 78
ORDER_ID*	Merchant reference number (Mandatory)	YES	AN	1	50	ESN78452
RETURN_URL*	Url of merchant website to get the response back after transaction is done (Mandatory)	YES	CH	5	1024	http://www.domain.com/response_url
SALT*	Provided by Dasshpe on activation (Mandatory)	YES	AN	16	16	2a1e47c4fba401c
HASH*	Unique value generated by SHA 256 hashing algorithm (Mandatory)	YES	AN	64	64	7995156CE4C40C44C41BECA3B9CE09B9
PAYMENT_TYPE*	(Mandatory)	YES	A	2	2	CC – CREDIT CARD DC- DEBIT

						CARD NB-NETBANKING WL- WALLET UP-UPI
MOP_TYPE*	(Mandatory)	YES	A	2	2	VI –VISA,MS-MASTER,RU-RUPAY
CARD_NUMBER*	(Mandatory)	YES	N	14	16	4000000000000002
CARD_EXP_DT*	(Mandatory)	YES	N	6	6	DDYYYY
CVV	(Mandatory)	YES	N	3	4	CVV -123 AMEX-1234
Customer Details						
CUST_NAME	Customer name	NO	CH	1	150	John Pal
CUST_FIRST_NAME	Customer first name	NO	CH	2	150	John
CUST_LAST_NAME	Customer last name	NO	CH	2	150	Pal
CUST_STREET_ADDRESS1	Customer address	NO	CH	2	250	House no-101
CUST_CITY	Customer city	NO	CH	2	50	Gurgaon
CUST_STATE	Customer state	NO	CH	2	100	Haryana
CUST_COUNTRY	Customer country	NO	CH	2	100	India
CUST_ZIP	Customer zip	NO	AN	6	9	TWQ 123
CUST_PHONE	Customer phone	NO	NU	8	15	07417456565
CUST_EMAIL*	Customer email (Mandatory)	NO	CH	6	120	john@gmail.com
CUST_SHIP_	Customer	NO	CH	2	150	Pal

LAST_NAME	Shipping last name					
CUST_SHIP_FIRST_NAME	Customer shipping first name	NO	CH	2	150	John
CUST_SHIP_NAME	Customer shipping name	NO	CH	2	150	John Pal
CUST_SHIP_STREET_ADDRESS1	Customer shipping address	NO	CH	2	250	House no-101
CUST_SHIP_STREET_ADDRESS2	Customer shipping address	NO	CH	2	250	Block A
CUST_SHIP_CITY	Customer shipping city	NO	CH	2	50	Gurgaon
CUST_SHIP_STATE	Customer shipping state	NO	CH	2	100	Haryana
CUST_SHIP_COUNTRY	Customer shipping country	NO	CH	2	100	India
CUST_SHIP_ZIP	Customer shipping zip	NO	AN	6	9	TWQ 123
CUST_SHIP_PHONE	Customer shipping phone	NO	NU	8	15	07417456565
MERCHANT_PAYMENT_TYPE	payment option type	YES	AL	2	2	CC - credit Card DC - Debit Card NB - Net-Banking WL - WALLLET UP - UPI
Payment Details						
AMOUNT*	Total Sale Amount (Mandatory)	YES	NU	3	12	100
TXNTYPE*	Merchant Transaction Type AUTH/	YES	CH	4	50	SALE

	SALE (Mandatory)					
CURRENCY_CODE*	3-digit code of the currency (Mandatory)	YES	NU	3	3	356
Item Level Details						
PRODUCT_DESC	Description of product	NO	CH	1	1024	xyz

*Abbreviation NU - Numeric, CH – Character, AN – Alphanumeric

*Refer Amount format for Amount

HASH: Generate SHA-256

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.EmptyStackException;
import java.util.Map;
import java.util.Stack;
import java.util.TreeMap;
import org.apache.commons.codec.binary.Hex;
```

```
public class ChecksumUtils {
	private static Stack<MessageDigest> stack = new
Stack<MessageDigest>();
	private final static String separator = "~";
	private final static String equator = "=";
	private final static String hashingAlgo = "SHA-256";

	// Hash calculation from request map
	public static String generateCheckSum(Map<String, String> parameters,
String secretKey)
			throws NoSuchAlgorithmException {
		Map<String, String> treeMap = new TreeMap<String,
String>(parameters);

		StringBuilder allFields = new StringBuilder();
		for (String key : treeMap.keySet()) {
			allFields.append(separator);
			allFields.append(key);
			allFields.append(equator);
			allFields.append(treeMap.get(key));
		}

		allFields.deleteCharAt(0); // Remove first FIELD_SEPARATOR
		allFields.append(secretKey);

		// Calculate hash
		return getHash(allFields.toString());
	}

	// Response hash validation
	public static boolean validateResponseChecksum(Map<String, String>
responseParameters, String secretKey,
			String responseHash) throws NoSuchAlgorithmException {
		boolean flag = false;
		String generatedHash = generateCheckSum(responseParameters,
secretKey);
		if (generatedHash.equals(responseHash)) {
			flag = true;
		}
		return flag;
	}

	// Generate hash from the supplied string
	public static String getHash(String input) throws
NoSuchAlgorithmException {
		String response = null;

		MessageDigest messageDigest = provide();
		messageDigest.update(input.getBytes());
		consume(messageDigest);

		response = new String(Hex.encodeHex(messageDigest.digest()));

		return response.toUpperCase();
```

```
	}// getSHA256Hex()

	private static MessageDigest provide() throws
NoSuchAlgorithmException {
		MessageDigest digest = null;

		try {
			digest = stack.pop();
		} catch (EmptyStackException emptyStackException) {
			digest = MessageDigest.getInstance(hashingAlgo);
		}
		return digest;
	}

	private static void consume(MessageDigest digest) {
		stack.push(digest);
	}
}
```

Amount format

The amount of the transaction, expressed in the smallest currency unit. The amount must not contain any decimal points, thousands

separators or currency symbols. This value cannot be negative or zero.

For example, INR 12.50 is expressed as 1250.
INR 1 is expressed as 100.
Note: Transactions in currency IDR (Indonesian Rupiah) will use an exponent of 0 (zero). This means an amount expressed as 1250 will be treated as IDR Rp1, 250 and not IDR Rp12.50 (with exponent 2) unlike other currencies.

Dasshpe Response Format

When the payment process is complete Dasshpe sends the details of the transaction to the Response_url. This is done with a standard HTTP POST request. The Dasshpe server continues to post the status until a response of HTTP OK (200) is received from your server or the number of posts exceeds 10.

Table 2: Success And Failure Response
Success Response Code:

{
"POST": {
"RESPONSE_DATE_TIME": "2019-12-17 15:29:30",
"RESPONSE_CODE": "000",
"CUST_PHONE": "9911889966",
"MOP_TYPE": "VI",
"CARD_MASK": "400000******0119",
"CURRENCY_CODE": "356",
"RRN": "935109113589",
"STATUS": "Captured",
"PRODUCT_DESC": "Demo Transaction",
"AMOUNT": "100",
"RESPONSE_MESSAGE":
"SUCCESS",
"CUST_EMAIL": "rohit.singh@Dasshpe.com",
"TXN_ID": "1912171529291040",
"ACQ_ID": "5765767697276303703009",
"TXNTYPE": "SALE",

"HASH":
"BCB4FBD716AB3E82C380B1C1835759A3CF781D691E32524537
E99718E2A64FDB",
"PAYMENT_TYPE": "CC",
"RETURN_URL":
"http:\/\/localhost\/projects\/php_kit_custom\/response-json.php",
"PAY_ID": "1903260434111000",
"ORDER_ID": "DASSHPED1712191059",
"ORIG_TXN_ID": "1912171529271039",
"CUST_NAME": "test"
},
"GET": [],
"IS_VALID": true

Field name	Description	Example value
CUST_NAME	Customer name	John Pal
TXNTYPE	Type of transaction	SALE/AUTH
AMOUNT	Total Sale Amount	100
CURRENCY_CODE *	3-digit code of the currency	826
ORDER_ID	Merchant reference number	ESN78452
PAY_ID	Pay ID is given by Dasshpe	160234578452178
TXN_ID	Transaction Id generated by Dasshpe	150611417421130
RESPONSE_CODE *	Code for transaction status	000
RESPONSE_MESSAGE *	Response message for transaction status	SUCCESS
HASH*	Unique value generated by SHA 256 hashing algorithm	7995156CE4C40C44C41BECA3B9CE09B9
AUTH_CODE	Authorization code	123456
RRN	Bank reference number	789456132
STATUS	Transaction status	Approved/ Captured/Declined

CUST_EMAIL	Customer email	john@gmail.com
RESPONSE_DATE	Date of response	TBD
RESPONSE_TIME	Time of response	TBD
PRODUCT_DESC	Description of product	xyz

* Refer Table 2 for Currency Code

* Refer Table 4 & Table 5 for Response Code

* Refer Table 4 & Table 5 for Response Message

* Refer Generate secure hash

Validating the status response

We recommend that you validate the transaction details in the status response. This can be done as follows:

1. Create a pending transaction or order for a fixed amount on your website.

2. Redirect the customer to the Dasshpe Payment Gateway, where they complete the transaction.

3. Dasshpe will post the transaction confirmation to your ‘Response_url’ page. This will include the 'Amount' (Amount) parameter.

4. Your website should validate the parameters received by calculating the SHA2 signature. If successful, it should compare the value in the confirmation post(amount parameter) to the one from the pending transaction or order on your website. You can also compare other parameters such as ‘order id’ etc.

5. Once you have validated the transaction data you can process the transaction, for example, by dispatching the goods ordered.

Generating a secure hash

SHA-256 signature

The merchant code creates the Secure Hash value on the Transaction Request data. The Payment Server creates another Secure Hash value and sends it back to the merchant in the Transaction Response.
The Secure Hash is a hexadecimal encoded SHA-256 HMAC of a concatenation of VPC and User Defined parameters. The concatenation of parameters takes the form of a set of name-value pairs, similar to the parameter string for an HTTP GET call.

Method of generating hash

To generate a hash you need to make a request of all the required parameters For example, if you want to pass the following name value pairs in your request
{PAY_ID=1507281443471000&ORDER_ID=ORD220920151610&TXNTYPE=SALE&AMOUNT=100&CURRENCY_CODE=356&CUST_NAME=Demo+Merchant&CUST_STREET_ADDRESS1=Demo+Address1&CUST_STREET_ADDRESS2=Demo+Address2&CUST_CITY=Demo+City&CUST_STATE=Demo+State&CUST_COUNTRY=Demo+Country&CUST_ZIP=Demo+Zip+Code&CUST_EMAIL=demo%40Dasshpe.com&CUST_PHONE=1234567890&CUST_SHIP_NAME=Demo+Ship+Customer&CUST_SHIP_STREET_ADDRESS1=Demo+Ship+Address1&CUST_SHIP_STREET_ADDRESS2=Demo+Ship+Address2&CUST_SHIP_CITY=Demo+Ship+City&CUST_SHIP_STATE=Demo+Ship+State&CUST_SHIP_COUNTRY=Demo+Ship+Country&CUST_SHIP_ZIP=Demo+Ship+Zip+Code&CUST_SHIP_EMAIL=demoship%40Dasshpe.com&CUST_SHIP_PHONE=0123456789&RETURN_URL=http%3a%2f%2flocalhost%3a8080%2fMerchantSimulator%2fresponse.jsp&PRODUCT_DESC=Demo+Product}
Then you need to sort all the parameters in ascending order and add “Tiled” symbol as separator.
The Output will be as follows

{AMOUNT=100~CURRENCY_CODE=356~CUST_CITY=Demo City~CUST_COUNTRY=Demo Country~CUST_EMAIL=demo@Dasshpe.com~CUST_NAME=Demo Merchant~CUST_PHONE=1234567890~CUST_SHIP_CITY=Demo ShipCity~CUST_SHIP_COUNTRY=DemoShipCountry~CUST_SHIP_EMAIL=demo ship@Dasshpe.com~CUST_SHIP_NAME=Demo ShipCustomer~CUST_SHIP_PHONE=0123456789~CUST_SHIP_STATE=Demo Ship State~CUST_SHIP_STREET_ADDRESS1=Demo Ship Address1~CUST_SHIP_STREET_ADDRESS2=Demo Ship Address2~CUST_SHIP_ZIP=Demo Ship Zip Code~CUST_STATE=Demo State~CUST_STREET_ADDRESS1=Demo Address1~CUST_STREET_ADDRESS2=Demo Address2~CUST_ZIP=Demo Zip Code~ORDER_ID=SIGORD220920151610~PAY_ID=1507281443471000~PRODUCT_DESC=DemoProduct~RETURN_URL=http://localhost:8080/MerchantSimulator/response.jsp~TXNTYPE=SALE}

Next step is to append the Secret Key at the end of the parameter string given by Dasshpe Payment Gateway to you. After adding you will get the following output

{AMOUNT=100~CURRENCY_CODE=356~CUST_CITY=Demo City~CUST_COUNTRY=DemoCountry~CUST_EMAIL=demo@Dasshpe.com~CUST_NAME=DemoMerchant~CUST_PHONE=1234567890~CUST_SHIP_CITY=Demo Ship City~CUST_SHIP_COUNTRY=Demo ShipCountry~CUST_SHIP_EMAIL=demoship@Dasshpe.com~CUST_SHIP_NAME=DemoShipCustomer~CUST_SHIP_PHONE=0123456789~CUST_SHIP_STATE=Demo Ship State~CUST_SHIP_STREET_ADDRESS1=Demo Ship Address1~CUST_SHIP_STREET_ADDRESS2=Demo Ship Address2~CUST_SHIP_ZIP=Demo Ship Zip Code~CUST_STATE=Demo State~CUST_STREET_ADDRESS1=Demo Address1~CUST_STREET_ADDRESS2=Demo Address2~CUST_ZIP=Demo Zip

Code~ORDER_ID=PAYORD220920151610~PAY_ID=1507281443471000~PRODUCT_DESC=Demo Product~RETURN_URL=http://localhost:8080/MerchantSimulator/response.jsp~TXNTYPE=SALEb6200e78557ee}

Next step is to append the Secret Key at the end of the parameter string given by Dasshpe Payment Gateway to you. After adding you will get the following output

{AMOUNT=100~CURRENCY_CODE=356~CUST_CITY=Demo City~CUST_COUNTRY=Demo Country~CUST_EMAIL=demo@Dasshpe.com~CUST_NAME=Demo Merchant~CUST_PHONE=1234567890~CUST_SHIP_CITY=Demo Ship City~CUST_SHIP_COUNTRY=Demo Ship Country~CUST_SHIP_EMAIL=demoship@Dasshpe.com~CUST_SHIP_NAME=Demo ShipCustomer~CUST_SHIP_PHONE=0123456789~CUST_SHIP_STATE=Demo Ship State~CUST_SHIP_STREET_ADDRESS1=Demo Ship Address1~CUST_SHIP_STREET_ADDRESS2=Demo Ship Address2~CUST_SHIP_ZIP=Demo Ship Zip Code~CUST_STATE=Demo State~CUST_STREET_ADDRESS1=Demo Address1~CUST_STREET_ADDRESS2=Demo Address2~CUST_ZIP=Demo Zip Code~ORDER_ID=PAYORD220920151610~PAY_ID=1507281443471000~PRODUCT_DESC=Demo Product~RETURN_URL=http://localhost:8080/MerchantSimulator/response.jsp~TXNTYPE=SALEb6200e78557ee}

After completing the above mentioned process you will have to call SHA 256 algorithm and pass the parameter string to the same and the SHA will return you the desired result as below

Hash value={6797f1842deb4f3ebaead53e1bafd5a535d322b9fa3893f201fdb03933eeae09}

Now you have to convert the generated value to the Upper Case and you will get the final result as hash value

Hash value = 6797F1842DEB4F3EBAEAD53E1BAFD5A535D322B9FA3893F201FDB03933EEAE09

The purpose of the **SHA2signature**field is to ensure the integrity of the data posted back to your server. You should always compare the **SHA2signature**field's value posted by Dasshpe's servers with the one you calculated.
To calculate the **SHA2sig**, you need to take the values of the fields listed above exactly as they were posted back to you, concatenate them and perform a **SHA2** calculation on this string.

Code integration examples
You can use the examples below to generate your session ID from Dasshpe, which is the recommended method for connecting to the Dasshpe Payment Gateway.

```
<formaction=https://www.merchant.Dasshpe.com/crm/jsp/paymentrequest method=post>
<input type="text" name="PAY_ID" value="1507231702331001"/>
<input type="text" name="MERCHANTNAME" value="Demo Merchant"/>
<input type="text" name="ORDER_ID" value="ORDID2234"/>
<input type="text" name="AMOUNT" value="100"/>
<input type="text" name="TXNTYPE" value="SALE"/>
<input type="text" name="CUST_NAME" value="Demo"/>
<input type="text" name="CUST_STREET_ADDRESS1" value="Gurgaon"/>
<input type="text" name="CUST_ZIP" value="123456"/>
<input type="text" name="CUST_PHONE" value="9999999999"/>
<input type="text" name="CUST_EMAIL" value="test@gmail.com"/>
<input type="text" name="PRODUCT_DESC" value="CD Player"/>
<input type="text" name="CURRENCY_CODE" value="356"/>
<input type="text" name="RETURN_URL"
value="http://www.yourwebsite.com/response.php"/>
<input type="text" name="HASH"
value="12345678901234567890123456789012345678901234567890123456789012345678901234"/>
<input type="text" name="PAY_ID" value="1507231702331001"/>
<input type="submit" value="Click to Pay" name="submit"/>
</form>
```

Transaction Refund API Test Refund URL:
https://uat.dasshpe.com/crm/services/paymentServices/transact

Production Refund URL :
https://secure.dasshpe.com/crm/services/paymentServices/transact

Request Format For Refund

{
"PG_DATE_TIME":"2019-05-03 16:53:47",
"PAY_ID":"19021413120710004", "AMOUNT":"100",
"ORIG_TXN_ID":"1905031653461002",
"CUST_EMAIL":"customer@gmail.com", "TXNTYPE":"REFUND",
"CURRENCY_CODE":"356",
"*RETURN_URL*": "http://www.yourwebsite.com/response.php",
"HASH":"12345678901234567890123456789012345678901234567890123456789012345678901234"
}

Response Format For Refund

{
"RESPONSE_DATE_TIME": "2019-05-03 17:10:11",
"RESPONSE_CODE": "000",

"TXN_ID": "19050317071410000", "MOP_TYPE": "VI",
"CARD_MASK": "400000******0002", "ACQ_ID":
"5568831111886666004010", "TXNTYPE": "REFUND",
"CURRENCY_CODE": "356",
"HASH":"EC8AECD2E7A429707042465E26AF1A6C921EF0097076
8E0285ADA60EDE5", "PAYMENT_TYPE": "CC",
"STATUS": "Captured", "PAY_ID": "1902141312071004",
"ORDER_ID": "676776",
"AMOUNT": "100", "RESPONSE_MESSAGE": "SUCCESS",
"ORIG_TXN_ID": "1905031653461002",
"CUST_EMAIL": "customer@gmail.com"
}

Transaction Status API:

Method:GET

http://uat.Dasshpe.com/crm/services/paymentServices/getStatus?
PAY_ID=180718215751 1000&ORDER_ID=DASSHPE0000155

GATEWAY OPTIONS AND RESPONSES

Response Code for Valid Transaction

Table 3: Response Codes for Valid Transactions

Response Code	Response message
000	Success
001	Acquirer Error
002	Denied
003	Timeout
004	Declined
005	Authentication not available
006	Transaction processing
007	Rejected by acquirer
008	Duplicate
009	Response signature did not match
010	Cancelled by user

011	Invalid request not available

Response Code for Invalid Transaction

Table 4: Response Codes for Invalid Transactions

Response Code	Response Code
300	Invalid Request
317	Invalid currency code
318	Invalid Amount
319	Invalid Order Id
320	Invalid Txn Type
321	Invalid Hash
900	Internal System Error

Supported Currency and Currency Codes

Table 5: Supported Currency with Currency Codes

Payment Method		
Name	**Abbreviation**	**Code**
Indian Rupee	INR	356
Pound	GBP	826
Dollar	USD	840
Euro	EUR	978