Bridge PG

2016

The document can be used as a guide for Merchants ready to integrate with the CSC platform and should be used as a reference document for understanding the usage and process for integration of the CSC Bridge PG service for digital wallet payments.

www.csc.gov.in

Copyright

CSC E-Governance Services India Limited.

Registered office:

CSC E Governance Services India Limited
3rd Floor Electronics Niketan, Lodhi Road, New Delhi

Preface

CSC e-Governance Services India Limited is a **Special Purpose Vehicle** (CSC SPV) incorporated under the Companies Act, 1956 by the Department of Electronics and Information Technology (DeitY), Ministry of Electronics and Information Technology (MEiTY), Government of India to monitor the implementation of the Common Service Center Scheme.

CSC SPV is connecting local population with the Government departments, banks, and insurance companies and with various service providers in private sector using IT-Enabled network of citizen service points.

Bridge payment gateway is a secure hosted payment page, where you can redirect customers from your website to make a payment through Bridge payment gateway. The guide covers the steps involved in a payment process and the information that needs to be passed from your web servers to Bridge, to enable Bridge to process the payments.

This document details out the implementation of a Payment Gateway Integration and related information that is required to understand the functioning of Bridge payment gateway.

Contents

1. Overview

Common Service Center Scheme (CSC) engages a powerful network of more than 2 lakh users who conduct business on our Digital Seva Portal everyday to deliver government & business services to citizens across the length and breadth of the country. It is a pan-India network catering to regional, geographic, linguistic and cultural diversity of the country, thus enabling the Government's mandate of a socially, financially and digitally inclusive society.

CSC enables a digital marketplace for our network of operators arming them with a digital wallet; they can use to make payments for products and service purchased on the portal. CSCs offer assisted access of e-services to citizens with a focus on enhancing governance, delivering essential government and public utility services, social welfare schemes, financial services, education & skill development courses, health products, agriculture services and digital literacy, apart from a host of B2C services.

Bridge PG

Bridge payment gateway is a secure hosted payment page, where you can redirect customers from your website to make a payment through Bridge payment gateway. Bridge PG manages the interactions between Merchants/Service providers and the wallets users for making online payments using the digital wallet including payment reconciliation for settlement services with merchants.

CSC Bridge PG is a next generation payment solution for connecting an online merchant with the CSC digital wallet for online payments and further exposing their service and product to a large customer base for increased business opportunities.

Bridge PG is a powerful payment calculator, cart with a digital wallet and a gateway to e-payment service.

- Access to a vibrant B2B marketplace
- Fully secured with industry standard symmetric and asymmetric encryptions
- Highly scalable for large volume transaction processing
- Faster checkout with negligible failure
- Works like cart & PG or only PG
- Zero cost Integration

1.1. About this guide

1.1.1. Objective and target users

This guide provides details on how to connect your website to the CSC Wallet using the Bridge PG service. It is intended for users who have a working knowledge of HTML. The guide covers the steps in the payment process and the information that needs to be passed from your web servers to Bridge, to enable CSC wallet to process payments.

Access to large network of Customers

CSC has an active network of 200,000 users with more being added each day. You benefit with our large network of users which results in more business opportunities.

1.1.2. Convention

The table below lists some of the conventions used in this guide.

CONVENTION	DESCRIPTION
Reference	Indicates a reference to another section in this guide.

Table 1

1.1.3. Abbreviation

The table below lists some of the abbreviations used in this guide.

ABBREVIATION	DESCRIPTION
MEiTY	Ministry of Electronics & Information Technology
DEiTY	Department of Electronics & Information Technology
CSC	Common Service Center
SPV	Special Purpose Vehicle
PG	Payment Gateway
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
Recon	Reconciliation
AES	Advanced Encryption Standard
RSA	Rivest-Shamir-Adleman algorithm
SOA	Service-Oriented Architecture
API	Application Programming Interface

Table 2

1.1.4. Definitions

The table below describes the keywords used in the guide.

TERM	DEFINITION
Merchant	Businesses ready to sell their product or services online
Wallet	An online prepaid financial instrument used to make payments for products and service purchased on the portal.
Public Key	Key component of RSA keys widely circulated for encryption of data
Private Key	Key that can only decrypt the data that has been encrypted by its public part

Table 3

1.1.5. Who to Contact for queries

Please contact our support department at below mention email and toll free number for all integration related queries.

Email Id	partners@csc.gov.in
Toll Free Number	1800 3000 3468

Table 4

2. Introduction Bridge PG

Bridge PG is CSCs flagship product platform for managing online payments. It features simplified integration procedures and is fully secured with industry standard encryption methodologies. Integration with the CSC Pay platform is free of cost. The Bridge Pay is also equipped with a seamless payment settlement engine which is secure, fully automated and highly flexible to adhere to your complex business rules.

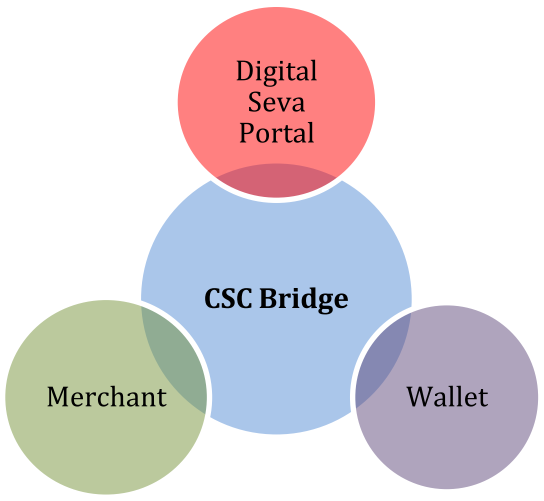


Figure 1

Key Features

- Bridge Payment Gateway enables to seamlessly manage complex transaction rules with ease
- Seamlessly switch between dummy to actual transaction of the production environment by just changing payload parameter to demo/prod
- Seamlessly integrate to banking and recon engine for faster post transaction processing
- End to end data encryption for security using symmetric and asymmetric encryption
- Enables multi mode wallet top up mechanism for faster and secure wallet recharge
- Linear scalable for high performance

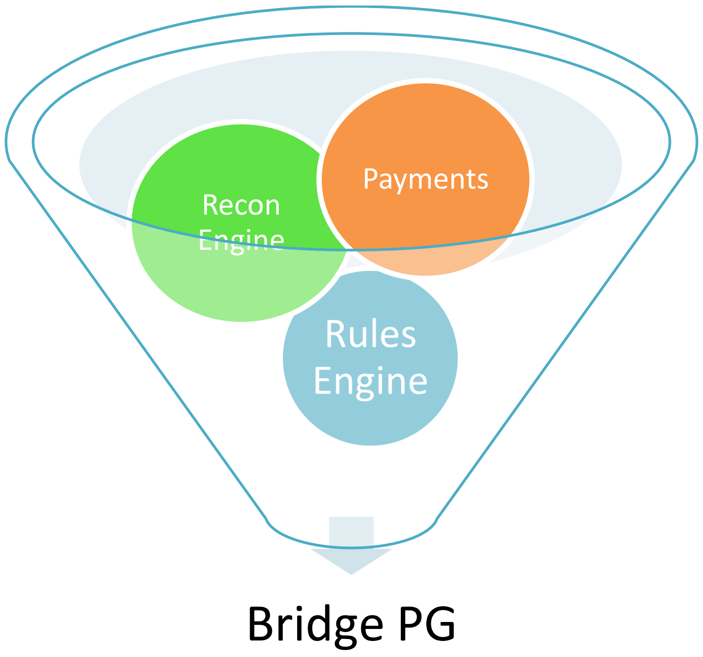


Figure 2

Bridge Platform Components

The bridge platform is designed on a highly modularized SOA comprising of multiple components which functions independently and collaboratively when required.

Payments

This module is core to the platform and interacts with the wallet to enable online payment processing.

Recon Engine

Bridge features industry standard process based payment settlement and transaction reconciliation.

Rules Engine

Rules engine manages the complex business rules associated with the merchant products or services during a transaction.

Event Sync & Warehouse

Sync utility is an event program which is responsible for transaction processing and inter component messaging asynchronously, while the warehouse is the core data store of the system.

2.1. Payment Flow

A simplified illustration of the Bridge transaction flow is shown in the figure below.

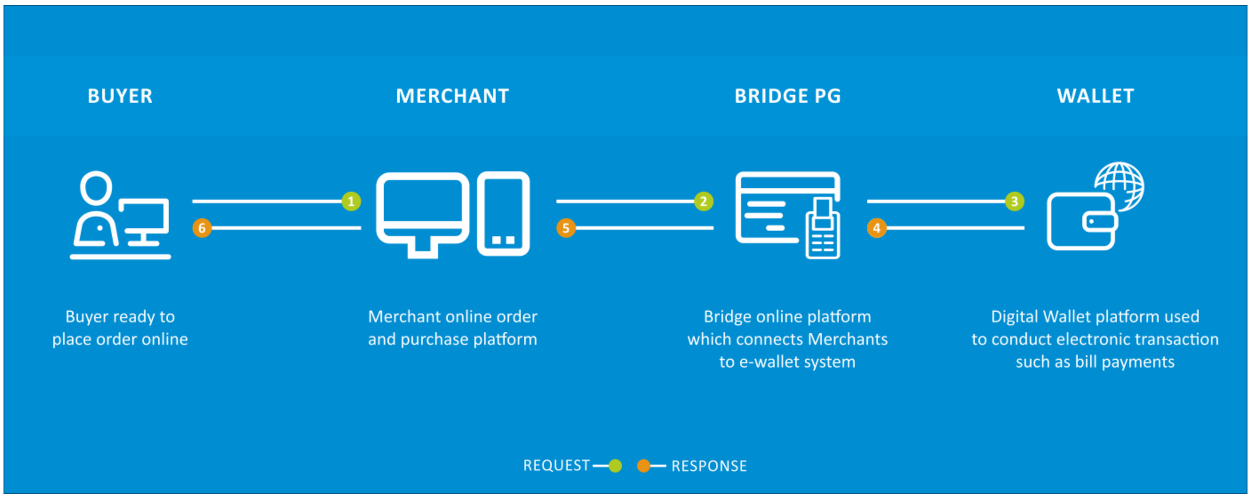

BUYER
MERCHANT
BRIDGE PG
WALLET
1
2
3
4
5
6
Buyer ready to place order online
Merchant online order and purchase platform
Bridge online platform which connects Merchants to e-wallet system
Digital Wallet platform used to conduct electronic transaction such as bill payments
REQUEST
RESPONSE

Figure 3, Payment Flow

The basic steps are mentioned below.

1. When the customer is ready to pay for goods or services on merchant website, they would select the CSC Bridge payment option on your website.
2. Merchant generates the payment payload for bridge payment which includes parameters as referenced as per the process outlined.
3. As the customer clicks on **"CSC Pay"** button on the Merchant Application, which would post the payload on the Bridge PG payment URL, along with the customer being redirected to the pay page.
4. On the pay page the transaction details are displayed as per the product Configuration.
5. Bridge PG would request authorization for the payment from the customer, which uses the CSC Digital Seva Connect credentials.
6. Upon successful authentication the customer is securely redirected to the Wallet page.
7. The Wallet approves or rejects the transaction on the basis of wallet PIN, used for two factor authentication, as entered by the customer.

8. Bridge PG displays the Checkout confirmation page, containing the transaction result and further redirects customer to merchant application with transaction receipt number.

2.2. Two factor authentication

CSC Digital wallet utilizes two factor authorizations for payment completion as an added layer for additional security of the payments being made through the wallet.

CSC Pay enables the Wallet Pin as the second component for authorizing a payment which confirms a user's claimed identity for being utilized for the combination of the alternate secret from his/her Digital Seva Connect credential.

2.3. Tax Calculations

Bridge PG features an online real time solution for online businesses to enable taxation rules as per statutory guidelines on the payment processing platform. Merchants do not need to perform additional functions to obtain applicable taxes rather simply add tax rules to their products during the integration process.

★

Tax calculations are handled as per statutory guide lines and configured for each product and transaction type. Tax rates are also updated regularly as per prevailing rates.

2.4. Connecting to Wallet Checkout

Connecting to the Bridge PG wallet checkout page is simple and can be achieved with minimum efforts by posting the payment payload to the URL generated by the Bridge Library. The bridge library generates a unique access token, which becomes the part of the PG URL as an access segment. The access segment is an additional layer of cover to safeguard users and merchants from abuse.

It is mandatory to utilize the URL generated by the Bridge Library for checkout page display.

★
Merchants are advised to safeguard their Configuration parameters adequately to prevent exploits.

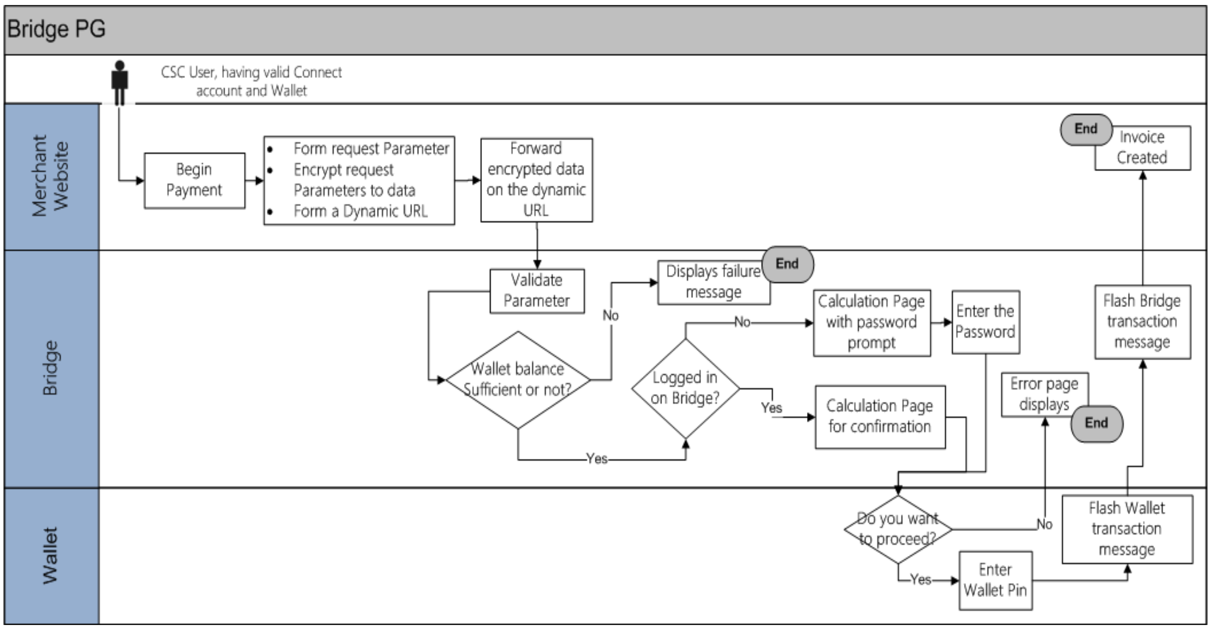

Bridge PG
CSC User, having valid Connect account and Wallet
Merchant Website
Bridge
Wallet
Begin Payment
Form request Parameter
Encrypt request Parameters to data
Form a Dynamic URL
Forward encrypted data on the dynamic URL
End
Invoice Created
Validate Parameter
Wallet balance Sufficient or not?
No
Yes
Displays failure message
End
Logged in on Bridge?
No
Yes
Calculation Page with password prompt
Enter the Password
Calculation Page for confirmation
Flash Bridge transaction message
Error page displays
End
Do you want to proceed?
No
Yes
Enter Wallet Pin
Flash Wallet transaction message

Figure 4

2.5. Status Messages

A transaction can have several states. The different states of a transaction cycle on the Bridge PG are listed below:

Transaction Status

STATUS	CODE
Successful	100
Failure	101
Insufficient balance in Wallet	102
Transaction cancel / abort by customer	103

Table 5

Failure Status Code

The list below represents all possible scenarios of failures and their corresponding meanings.

CODE	URL	STATUS	DESCRIPTION
801	*direct_access*	No Direct Access Allowed	Bridge Pay Page URL being accessed directly.
802	*form_resubmit*	Error: Form Resubmit Detected	Same (potentially same) data from merchant received. Commonly occurs on page refresh or when merchant transaction id used in any previous transaction being sent again.
803	*insufficient_balance*	insufficient_balance	Wallet balance is not adequate to complete the transaction
804	*wallet_bal_trans_failed*	insufficient_balance	Wallet balance is not adequate to complete the transaction
805	*pg_transaction_failed*	PG Transaction Failed	PG Transaction was declined
806	*pg_cancelled_aborted*	Transaction Cancelled Aborted	The transaction was Cancelled by user
807	*pg_error*	PG Error	Generic Error occurred at Wallet pay page.
808	*incorrect_merchant_message*	Incorrect Merchant Data	Data posted to pay page is not as per the specifications. There is mismatch in date of transaction or the time difference is more than 10 minutes.
809	*incorrect_merchant_data*	Error! No Merchant Data	No data was posted on the pay page on merchant site

810	*incorrect_merchant_info*	Error! Incorrect Parameters in Request Data.	Decryption Failed. (Encryptions Keys might be incorrect.)
811	*no_balance*	No Wallet Balance.	User wallet has no balance
812	*no_wallet_data*	Error! No Data From Wallet.	Transaction could not be processed due to zero balance in wallet.
813	*duplicate_wallet_data*	Duplicate wallet data.	Declined due to duplicate data.
814	*no_product*	Product Not Defined in System	Specified product does not exist.
815	*form_resubmit2*	Form resubmit Detected.	Form resubmit Detected.
816	*unknown_aplication*	Unknown Application	Application is not active at bridge.
817	*req_id_mismatch*	Error! Request ID mismatch	Change in request ID during transaction detected.
818	*merchant_time_limit*	Date Difference exceeded limit.	Transaction date and time at merchant is not within the specified threshold with Bridge Pg .The Date Time zone may be different.
819	*incorrect_merchant_date*	Error! Incorrect Date encountered.	The date and time format provided by merchant is not in the accepted format, "merchant_txn_datetime" should only be in YYYY-MM-DD HH:SS:MM format .
820	*merchant_date_different*	Mismatch in Date	The date at Merchant transaction request does not match with bridge pg.
821	*Generic*	Error Occurred!	Generic error may have occurred.
822	*pricing_error*	Price or other parameter is incorrect	Price parameter in transaction request is not valid.

823	*error_post_calculation*	Component Mismatch	There is component mismatch in requested data.
824	*empty_discount_param*	Mandatory Discount Parameter (INR) is required. '0' being a valid value.	Mandatory Discount Parameter is required, '0' should be supplied in case of no discount being offered.
825	*empty_currency_param*	Currency Parameter is required. INR being the valid value.	Currency Parameter is required. INR being the valid value.
826	*merchant_product_mapping*	Incorrect Product	Product Id supplied in request data is incorrect.
827	*error_post_calculations2*	Configuration Error in Price of the Product	Configuration Error in Product price.
828	*param_not_set*	Mandatory parameter is not set.	Mandatory parameter is not set.
829	*logged_data_exception*	Exception Occurred	Internal Server Error.
830	*wallet_data_mismatch*	Internal Server Error	Wallet posted invalid data.
831	*wallet_data_mismatch1*	Internal Server Error	Wallet posted invalid data.
832	*wallet_response_lag1*	Wallet has not responded with in the stipulated time.	Wallet has not responded with in the stipulated time.
833	*wallet_response_lag*	Wallet has not responded with in the stipulated time.	Wallet has not responded with in the stipulated time.
834	*invalid_product_pricing*	Invalid Product Pricing	Product pricing is not valid.
835	*no_mtm_txn*	Internal Server Error	No merchant transaction ID
836	*no_mtm_txn_row*	Internal Server Error	Invalid merchant transaction ID
837	*pgp_no_message*	Invalid Packet to Payment Gateway Page	No data posted at directed post variable / URL.
838	*pgp_no_credentials*	No Credentials	No credentials posted at directed post variable / URL.
839	*pg_incorrect_credentials*	Invalid Credentials	Incorrect data posted at directed post variable / URL.
840	*incorrect_wallet_configuration*	Wallet user may not be configured properly	Wallet user may not be configured properly.
841	*invalid_url*	Invalid Payment Gateway address	The payment address id invalid.
842	*inactive_url*	User Inactive	User is note inactive.

843	*payment_amount*	Invalid Payment	The initiated payment is invalid
845	*incorrect_wallet_configuration*	User id inactive	The user is not active

Table 6

Refund Codes

CODE	STATUS	MESSAGE
120	**Success**	Refund request processed successfully.
121	**Process**	Wallet response not updated.
122	**Wait**	Wallet response not updated.
123	**Fail**	Refund request could not be processed.
124	**Hold**	Refund request put on hold, contact support
125	**Retry**	Refund request is being re-tried.
Default	**Busy**	Unable to answer your query, please try later.

3. Bridge PG API

Bridge PG makes available Rest based APIs which enables platform operators or merchant websites to interact with the Gateway in real time, question at time or after transactions to seek further information about a transaction as well as at the same time submit information useful for wallet and payment reconciliations.

The APIs enable to avoid unwanted operational issues related to payments including refunds, reversals, etc.

Listed below APIs are available with the BridgePG:

a. Transaction Reversal
b. Transaction Status
c. Refund Log
d. Refund Status
e. Recon Log

3.1. Data Formats

The default format for input and output is XML which is what we will use throughout this document. While the API is designed to support XML and JSON formats which are the most popular

Example:

<hostname> bridge / start /format / xml

```
https://csccloud.in ->  Host name
bridge              ->  End point name
start               ->  Resource (endpoint resource name)
format              ->  Format (reserved parameter for data format)
xml                 ->  xml (data format type required in response)

(Format type supported are xml, json )
```

By giving the client application developer the choice of data formats to use, the API is opened up to a much wider audience and can be used with more programming languages and systems. The API will out of the box support the following formats:

- xml - almost any programming language can read XML
- json - useful for JavaScript and increasing adoption on rest api's

3.1.1. Request Methods

As explained in the earlier section on the data formats, the Bridge API supports xml and json data formats for data exchange between the client and the server, this section will give a clear understanding of the usage and consumption methods of the API including the request and response methods for both an XML and a JSON data format.

There are several ways to test & communicate with the API, including Rest Clients, browser plug-in as well as command line tools.

Example to perform a post from command line using Curl:

```
curl http://x.x.x.x/bridgeapi/v1/diagnose
-H "Content-Type: application/json"
-d '{"device_type" : "D"}'
-X POST
```

SAMPLE FORMATS

a) REQUEST DATA FORMATS

XML:

```
<?xml version="1.0" encoding="utf-8"?>

<xml>
<merchant_id>00005</merchant_id>
<request_data>OPXqu9OUJENa+5H4sVipA==</request_data>
</xml>
```

JSON:

```
{"merchant_id":"1234","request_data":"PN005As6eX"}
```

b) RESPONSE DATA FORMATS

XML:

```
<?xml version="1.0" encoding="utf-8"?>
<xml>
<response_code>900</response_code>
<response_status>Fail</response_status>
<response_data>NA</response_data>
<response_message> Unable to locate your Transaction ID.</response_message>
<response_server>server=R|message=OK</response_server>
<response_date>2017-02-19 17:25:24</response_date>
</xml>
```

JSON:

```
{"response_code":"900","response_status":"Success/Fail","response_data":"fs
dfdsfdsfdsf","response_message":"Check back with a valid Merchant
transaction.","response_server":"server=R|message=OK","response_date":"2017
-02-19 21:02:37"}
```

3.2. API Details

3.2.1. Transaction Reversal

Merchants can utilize the service to submit back information for a transaction for which the Merchant could not received any response. Merchants can submit a reversal request for the same so a reversal can be initiated.

Merchant would be able to call reversal API after 120 seconds of initiating the transaction.

Endpoint
transaction/reverse

Method
POST

Input Parameters (for reference only)
```
merchant_id=00001|merchant_txn=1111|merchant_txn_datetime=2017-02-12
20:27:12|
```

Request Parameters

Parameter Name	Description	Type
merchant_id	Merchant ID- 5 Digit Merchant ID E.g. 54321	Integer Fixed Length- 5
merchant _txn	Unique MTM ID- Unique Merchant Transaction ID for which there is no response from CSC Bridge PG E.g. 23476-17022017	String of character Max. Length- 50
merchant_txn_datetime	Date time recorded by merchant- In ISO 8601 Format (yyyy-mm-dd HH:mm:ss) E.g. 2016-08-22 14:00:00	Date - Time

Table 7

Response Parameters

Parameter Name	Description	Type
merchant_id	Merchant ID- 5 Digit Merchant ID E.g. 54321	Integer Fixed Length- 5
reversal_reference	Reversal id- Unique numeric value	Numeric Fixed Length- 24
merchant _txn	Unique MTM ID- Unique Merchant	String of character

	Transaction ID for which there is no response from CSC Bridge PG E.g. 234_76_17022017	Max. Length- 50
txn_status	Status of transaction	String
reversal_response	In Process or Completed	String

Table 8

★
Reversals initiated on the same day will only be accepted and processed.

3.2.2. Transaction Status

The transaction status API can be used by merchants to re-confirm the result of the requested transaction.

Merchant would be able to call transaction status API after 05 seconds of initiating the transaction.

Endpoint

transaction/status

Method

POST

Input Parameters (for reference only)

```
merchant_id=1234|merchant_txn=12|csc_txn=123456|
```

Request Parameters

Parameter Name	Description	Type
merchant_id	Merchant ID- 5 Digit Merchant ID E.g. 54321	Integer Fixed Length- 5
merchant _txn	Unique MTM ID- Unique Merchant Transaction ID E.g. 234_76_17022017	String of character Max. Length- 50
csc_txn	Payment gateway transaction id- numeric unique value	Numeric Fixed Length- 24

Table 9

Response Parameters

Parameter Name	Description	Type
merchant _txn	Unique MTM ID- Unique Merchant Transaction ID E.g. 234_76_17022017	String of character Max. Length- 50
csc_txn	Payment gateway transaction id- numeric unique value	Numeric Fixed Length- 24
txn_status	Status of transaction e.g Success	String
response_code	Code returned by API	Integer Fixed Length- 3
reponse_message	Detailed response message	Variable Max. Length- 100

Table 10

3.2.3. Refund Log

Merchants can utilize the service to submit back information for a transaction which was successfully executed on Bridge but the merchant is not ready to render the service. Merchants can submit a refund request for the same so a refund can be initiated.

Endpoint

refund/log

Method

POST

Input Parameters (for reference only)

```
merchant_id=10025|csc_txn=586997876521322045883304|merchant_txn=MT44
3130423|merchant_txn_param=123456|merchant_txn_status=S|merchant_ref
erence=1212-
2017|refund_deduction=250|refund_mode=F|refund_type=R|refund_trigger
=M|refund_reason=unable to deliver service|
```

Request Parameters

Parameter Name	Description	Type
merchant_id	Merchant ID- 5 Digit Merchant ID E.g. 54321	Integer Fixed Length- 5
merchant_txn	Unique MTM ID- Unique Merchant Transaction ID E.g. 234_76_17022017	String of characters Max. Length- 50
merchant_txn_param	Unique Reference number for transaction in case multiple refunds are required Not applicable(N) or Applicable (E.g. 010205)	String(64)
csc_txn	Payment gateway transaction id-numeric unique value	Numeric Fixed Length- 24
refund_mode	Refund whether Partial (P) or Full (F)	String (P / F only)
refund_type	Refund Type whether Cancellation(C) or Refund(R)	String (C / R only)
refund_trigger	Refund initiated by Customer agent (A) or Merchant (M)	String (A / M only)
merchant_reference	Unique Merchant reference for transaction	Mixed (28)

refund_reason	Description or reason	String (50)
refund_deduction	Refund's Deducted Amount - Value E.g. 599.6	Decimal (6,2)
merchant_txn_status	Status of merchant transaction E.g. Success	String

Table 11

Response Parameters

Parameter Name	Description	Type
refund_status	Status of refund	String
merchant_id	Merchant ID- 5 Digit Merchant ID E.g. 54321	Integer Fixed Length- 5
merchant_txn	Unique MTM ID- Unique Merchant Transaction ID E.g. 234_76_17022017	String of characters Max. Length- 50
merchant_reference	Unique Merchant reference for transaction	Mixed (28)
refund_reference	Reference ID- 24 Digit Refund ID E.g. 725917548839	Integer Fixed Length -24
csc_txn	Payment gateway transaction id- numeric unique value	Numeric Fixed Length- 24

Table 12

3.2.4. Refund Status

For all the refund requests which the merchant has submitted, the confirmation or status of the processing can be requested using the service.

Merchant would be able to check the status of refund on a T+1.

Endpoint
refund/status

Method
POST

Input Parameters (for reference only)
```
merchant_id=10025|csc_txn=586997617021322045883304|merchant_txn=MT44
31304|refund_reference=881045817021823132319662|
```

Request Parameters

Parameter Name	Description	Type
merchant_id	Merchant ID- 5 Digit Merchant ID E.g. 54321	Integer Fixed Length- 5
csc_txn	Payment gateway transaction id-numeric unique value	Numeric Fixed Length- 24
merchant_txn	Unique MTM ID- Unique Merchant Transaction ID E.g. 234_76_17022017	String of character Max. Length- 50
refund_reference	Reference ID- 24 Digit Refund ID E.g. 725917548839	Integer Fixed Length -24

Table 13

Response Parameters

Parameter Name	Description	Type
merchant_txn	Unique MTM ID- Unique Merchant Transaction ID E.g. 234_76_17022017	String of characters Max. Length- 50
csc_id	CSC ID- 12 Digit CSC ID E.g. 725917548839	Integer Fixed Length -12
csc_txn	Payment gateway transaction id-numeric unique value	Numeric Fixed Length- 24
product_id	Product ID- 10 Digit Product ID as assigned by CSC E.g. 1234567890	Integer Fixed Length -10

txn_status	Transaction Status	Numeric Fixed Length- 3
response_status	Status of response	String

Table 14

3.2.5. Recon Log

Transaction reconciliation is an important activity for any financial transaction. Bridge PG features a recon API which can be used by merchants ready to render real time services by submitting the success transaction logs into the system.

Merchant should log every transaction and submit the same through API, post delivery of service.

Endpoint
recon/log

Method
POST

Input Parameters (for reference only)

```
merchant_id=10025|merchant_txn=MT4431304|csc_txn=5869976170213220458
83304|csc_id=500100100013|product_id=7837638392|txn_amount=1230.50|m
erchant_txn_datetime=2017-02-19
13:32:21|merchant_txn_status=S|merchant_receipt_no=MTR2012832-0980|
```

Request Parameters

Parameter Name	Description	Type
merchant_id	Merchant ID- 5 Digit Merchant ID E.g. 54321	Integer Fixed Length- 5
product_id	Product ID- 10 Digit Product ID as assigned by CSC E.g. 1234567890	Integer Fixed Length -10
csc_id	CSC ID- 12 Digit CSC ID E.g. 725917548839	Integer Fixed Length -12
merchant_txn	Unique MTM ID- Unique Merchant Transaction ID E.g. 234_76_17022017	String of characters Max. Length- 50
merchant_txn_datetime	Date time recorded by merchant- In ISO 8601 Format (yyyy-mm-dd HH:mm:ss) E.g. 2016-08-22 14:00:00	Date - Time
txn_amount	Transaction Amount- Value E.g. 599.6	Decimal (10.2)
csc_txn	Payment gateway transaction id- numeric unique value	Numeric Fixed Length- 24
merchant_receipt_no	Invoice / Receipt No	String

	E.g. 1256346ab	Max. Length- 32
merchant_txn_status	Success- "Success" will be display. In case of failure, description of the failure will be display.	Variable Max. Length- 256
merchant_txn_message	Detailed transaction message	Variable

Table 15

Response Parameters

Parameter Name	Description	Type
merchant_id	Merchant ID- 5 Digit Merchant ID E.g. 54321	Integer Fixed Length- 5
response_code	Code returned by API	Integer
recon_reference	Unique Recon Reference ID E.g. 234_76_17022017	Variable Max Length- 32
response_message	Detailed response message	Variable

Table 16

★
Recon log submitted for a period later than 15 days will not be accepted and processed.

4. Getting Started with Integration

This section provides detailed understanding on the requisites steps to be implemented for a successful integration with the Bridge platform. Examples are provided to understand the methods using ASP, Java & PHP.

This section assumes that the developer has a basic understanding of his chosen scripting language.

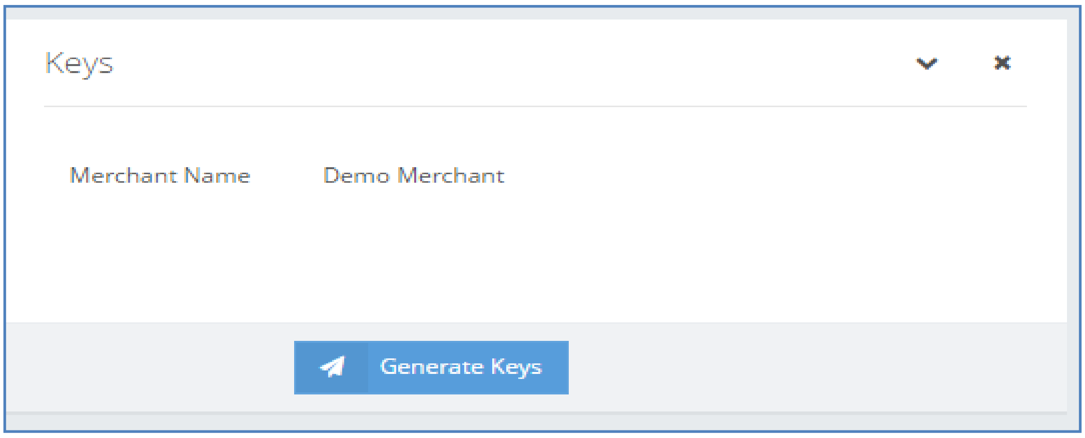
4.1. Merchant On-boarding

Once the Merchant is ready to integrate with CSC Platform, the process is instantiated by providing the Digital Seva Connect credentials, which merchants utilized to access the Merchant Centre in Bridge Portal.

Upon access to the Merchant Center, merchant admin users can generate Configuration files and key pairs to be used for the integration of the payment gateway in self service mode.

1. **Login to the merchant centre of bridge portal.**
2. **Generate Bridge PG Configuration File**

a. On left menu, click on CSC Bridge and select Key Manager Option. The following screen will display:

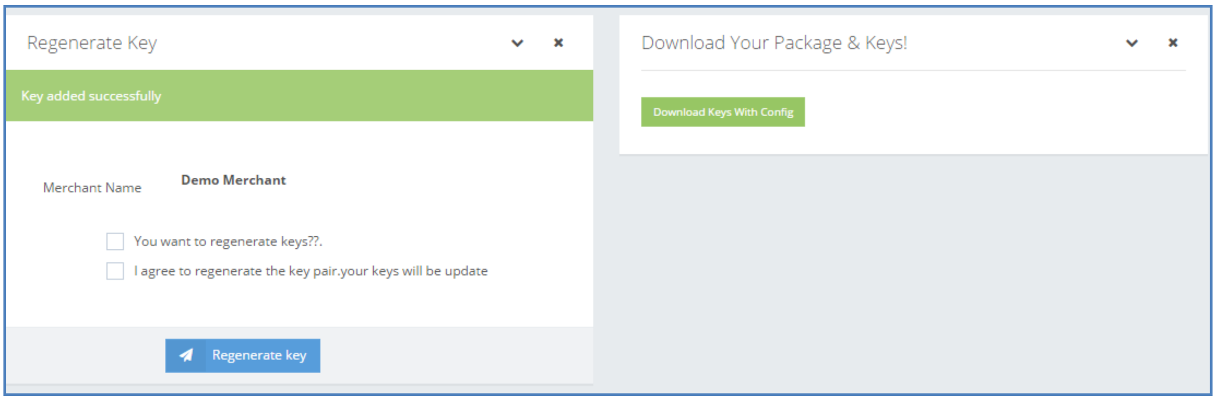


Keys

Merchant Name Demo Merchant

Figure 5

b. Click on **"Generate Keys"** button will generate the Keys and the following screen will be displayed:



Regenerate Key

Key added successfully

Merchant Name Demo Merchant

☐ You want to regenerate keys??.
☐ I agree to regenerate the key pair.your keys will be update

Figure 6

c. Now, you can download your package and keys from "**Download keys with Configuration"** button shown on the right side of the screen

★
Regenerate Keys: Bridge also gives you the facility of regenerating the keys. In case the merchant feels that he can refresh keys, he can refresh the key pair that he has generated.

4.2. Checklist

In order to integrate with the gateway, you must have the following items:

- **Merchant ID**

  Unique 5 digit ID assigned post the merchant on boarding is completed on the Bridge Portal

  For example: **10001 - Demo Merchant**

- **Digital Seva Connect credentials**

  Post the merchant is on boarded; CSCs Digital Seva Connect credentials are auto e-mailed to the administrative user of the merchant registered on the Bridge Portal.

  For example:

  Username: **demo_merchant**
  Password: **demo_pass**

- **Bridge PG Keys**

  Merchants can login to the Bridge portal using the Digital Seva Connect credentials and access the Key generation utility online to generate the RSA key pairs used for encryption and decryption of payment request & response payloads.

4.3. Configuration Process

Please follow the below mentioned steps to complete the Configuration process.

1. Login into your "https://portal.csc.gov.in"
2. Enter your User Id and password.
3. Go to **"CSC Bridge"**.
4. Select **"Key Manager"** option.
5. Generate keys by clicking on to the **"Generate Keys"** button.

6. Download your package and keys.
7. Configure application according to the downloaded file.

Note: In case, you do not have the login details, contact to our support department.

4.3.1. Bridge Config

Bridge Config is the configuration parameters file required for usage in Bridge PG as well as to connect to the gateway. The configuration file can be generated from the Merchant Center portal; however a live Configuration would require an admin approval for changes to take place.

```
MERCHANT_ID = 15679
MERCHANT_NAME = Demo Merchant
PUBLIC_KEY = LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tLS0NCk1JR
PRIVATE_KEY = LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tDQpN
SUCCESS_URL = https://demo.merchant.in/cart/success/
FAILURE_URL = https://demo.merchant.in/cart/failure
PAY_URL = https://pay.csccloud.in/v1/payment
API_KEY = dfdsfn3223fjndsfn2398scdsf
API_URL = https://bridgeapi.csccloud.in/v1/
VERSION= V1
```

The table below lists the Configuration parameters of Bridge PG.[1]

PARAMETER	DESCRIPTION
MERCHANT_ID	The unique ID provided by CSC to every Merchant.
MERCHANT _NAME	The unique name provided by CSC to every Merchant.
PUBLIC_KEY	1024 RSA Public key CSC provided to send encrypted data.

[1] URL*s* used are for example reference only. Actual URLs are part of the Config *file.*

PRIVATE_KEY	1024 RSA Private key of Merchant to decrypt data from Bridge PG.
SUCCESS_URL	URL called upon successful transaction.
FAILURE_URL	URL called upon failure transaction.
PAY_URL	URL of the payment gateway where message needs to be send.
API_KEY	The token used for identification of programs using the API services.
API_URL	URL of the API where message needs to be send.
VERSION	Version of Bridge platform

Table 17

4.3.2. PG Keys

Public and Private Key

Any data exchange happening between CSC and Merchant will be encrypted end to end so that the information is protected, secured and can't be misused. The Public and Private Key pair comprise of two uniquely related cryptographic keys.

★
Note: Private Key must not be shared with anyone and must be kept in safe custody. If private key gets shared or compromised, it may lead to serious security issue. You are solely responsible for safety of your private key and hence utmost care must be taken by you. In future whenever you feel that your private key has been compromised, you should immediately generate new key pair and should share your public key with our support team so that your key can be replaced immediately.

4.4. Pay Parameters (Transaction Payload)

4.4.1. URL and Parameters Require to Connect the Gateway

Interaction between the Merchant and CSC server is done over a secure HTTP connection.

Note: Bridge PG URL is a part of the config file generated and should be referenced from configuration file itself. Bridge Library should return constant value `PAY_URL`.

The parameters to be posted from merchant site are as follows:

Parameter Name	Description	Type
merchant_id	Merchant ID- 5 Digit Merchant ID E.g. 54321	Integer Fixed Length- 5
csc_id	CSC ID- 12 Digit CSC ID E.g. 725917548839	Integer Fixed Length -12
merchant_txn	Unique MTM ID- Unique Merchant Transaction ID E.g. 234_76_17022017	String of characters Max. Length- 50
merchant_txn_datetime	Date time recorded by merchant- In ISO 8601 Format (yyyy-mm-dd HH:mm:ss) E.g. 2016-08-22 14:00:00	Date - Time
product_id	Product ID- 10 Digit Product ID as assigned by CSC E.g. 1234567890	Integer Fixed Length -10
product_name	Product Name- Product Name Displayed on Pay page E.g. abcdefgh	Variable Max. Length- 32
txn_amount	Transaction Amount- Value E.g. 599.6	Decimal (10.2)
amount_parameter	Special Instruction for PG- Pipe separated multiple values as agreed with CSC during integration E.g. qwertyuiop	String Max. Length- 100
txn_mode	Mode- D = Debit; C = Credit E.g. D	Character Fixed Length -1
txn_type	Type- P = Production, D = Demo E.g. P	Character Fixed Length -1

csc_share_amount	Amount for CSC- Commission amount for CSC E.g. 100.2	Decimal (10.2)
return_url	Url on success E.g. https://www.abcd.com	Variable Max. Length-100
cancel_url	Url on cancel E.g. https://www.efghi.com	Variable Max. Length-100
currency	INR or 3 Character ISO currency code E.g. INR	Character Fixed Length -3
discount	Discount by merchant E.g. 7.2	Decimal (8.2)
txn_status_message	Success- "Success" will be display. In case of failure, description of the failure will be display.	Variable Max. Length- 256
merchant_receipt_no	Invoice / Receipt No E.g. 1256346ab	String Max. 32
pay_to_email	Email ID of payee to inform to E.g. abc@deg.com	Variable Max. Length -64
param_1	Additional Parameter that is returned in response E.g. abcdefgh1234	Variable Max. Length-100
param_2	Additional Parameter that is returned in response E.g. abcdefgh1234	Variable Max. Length-100
param_3	Additional Parameter that is returned in response E.g. abcdefgh1234	Variable Max. Length-100
param_4	Additional Parameter that is returned in response E.g. abcdefgh1234	Variable Max. Length-100

Table 18

Response parameters are as follows:

Parameter Name	Description	Type
txn_status	Transaction Status	Numeric Fixed Length- 3
status_message	Detailed message from Bridge Pay provides the description of status code	String Max Length- 64
merchant_id	Merchant ID- 5 Digit Merchant ID E.g. 54321	Integer Fixed Length- 5
csc_id	CSC ID- 12 Digit CSC ID E.g. 725917548839	Integer Fixed Length -12
merchant_txn	Unique MTM ID- Unique Merchant	String of

	Transaction ID E.g. 234_76_17022017	character Max. Length- 50
merchant_txn_datetime	Date time recorded by merchant- In ISO 8601 Format (yyyy-mm-dd HH:mm:ss) E.g. 2016-08-22 14:00:00	Date - Time
product_id	Product ID- 10 Digit Product ID as assigned by CSC E.g. 1234567890	Integer Fixed Length -10
csc_txn	Payment gateway transaction id- numeric unique value	Numeric Fixed Length- 24
txn_amount	Transaction Amount- Value E.g. 599.6	Decimal (10.2)
amount_parameter	Special Instruction for PG- Pipe separated multiple values as agreed with CSC E.g. qwertyuiop	Variable Max. Length- 100
txn_mode	Mode- D = Debit; C = Credit E.g. D	Character Fixed Length -1
txn_type	Type- P = Production, D = Demo E.g. P	Character Fixed Length -1
csc_share_amount	Amount for CSC- Commission amount for CSC E.g. 100.2	Decimal (10.2)
currency	INR or 3 Character ISO currency code E.g. INR	Character Fixed Length -3
discount	Discount by merchant E.g. 7.2	Decimal (8.2)
txn_status_message	Success- "Success" will be display. In case of failure, description of the failure will be display.	Variable Max. Length- 256
merchant_receipt_no	Invoice / Receipt No E.g. 1256346ab	String Max. 32
pay_to_email	Email ID of payee to inform to E.g. abc@deg.com	Variable Max. Length -64
param_1	Additional Parameter that is returned in response E.g. abcdefgh1234	Variable Max. Length-100
param_2	Additional Parameter that is returned in response E.g. abcdefgh1234	Variable Max. Length-100
param_3	Additional Parameter that is returned in response E.g. abcdefgh1234	Variable Max. Length-100
param_4	Additional Parameter that is returned in response E.g. abcdefgh1234	Variable Max. Length-100

Table 19

4.4.2. Characters Restricted on the Gateway

We strongly urge you to restrict the usage of the below mentioned characters as these characters would be blocked on the firewall leading to transaction failures.

Parameter	Symbol
Single Quote	‘
Double Quote	”
Less Than	<
Greater Than	>

Table 20

4.4.3. Return Pages

Return Page is the page to which the Client is redirected after a transaction. Currently there are two parameters that are supported on the bridge PG for return pages:

- Success Page- Success Page is the page where user is redirected after a successful transaction.
- Failure Page- Failure Page is posted after a failed, returned or a cancelled transaction. The user is redirected to the custom error page of the merchant.

4.5. Security

Bridge PG uses the industry standard methodologies for data and transaction security by deploying symmetric and asymmetric encryption methods which ensures that, transaction and data, both are safe.

Security is essential for online payments. CSC ensures that every transaction performed is done in a safe and secured manner.

★

Security of Transaction

CSC ensures the security of every transaction by the following:

- Payment payload is end to end encrypted utilizing industry standard RSA utilities.
- Merchant keys are dynamically generated from Merchant center and are known to concerned merchants only and not stored in our system.
- CSC Digital wallet utilizes two factor authorizations for payment completion as an added layer for additional security of the payments being made through the wallet.

4.6. Integration Language Support

The Bridge PG is built on open standards and ensures seamless integration in any language or platform used. While integration kits and sample merchant website applications have been built and are ready in **dot net, Java & PHP**, it is being worked out to put up similar examples in other languages.

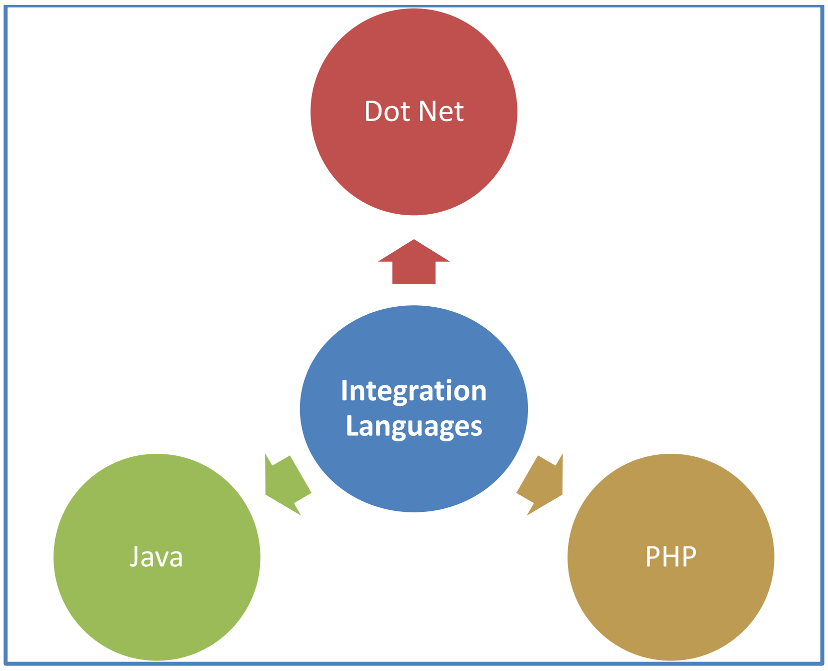


Figure 7

The Integration Kit comprises of below components-

- Bridge PG - Bridge PG is the core library of the integration kit.
- BridgePGUtil – BridgePGUtil is a wrapper to access the functions of Bridge library as well as the bridge API calls
- Payment - Sample payment forum page, from where user is redirected from merchant to bridge.
- Success - Sample page to illustrate success response
- Failure - Sample page to illustrate failure response
- Readme
- License

[1] URL*s* used are for example reference only. Actual URLs are part of the Config *file.*

★

Download Link:

You can download the Bridge PG integration kit from the following link.

https://github.com/csc-egov/BridgePG

4.6.1. dot net

The illustrated code sample below provides the understanding of using the **dot net** integration kit.

Step 1 Create an encrypted payment request as in example file. **(shop.aspx.cs)**

```
        protected void Button1_Click(object sender, EventArgs e)
        {
            try
            {
                if ((string)Session["logedin"] != "true")
Response.Redirect("login.aspx", true);
                else
                {

                    string merchant_id =
ConfigurationManager.AppSettings["merchant_id"];
                    string csc_id = Session["username"].ToString();
                    string merchant_txn = merchant_id +
DateTime.Now.Year.ToString().PadLeft(4, '0') +
DateTime.Now.Month.ToString().PadLeft(2, '0') +
DateTime.Now.Day.ToString().PadLeft(2, '0') +
DateTime.Now.Hour.ToString().PadLeft(2, '0') +
DateTime.Now.Minute.ToString().PadLeft(2, '0') +
DateTime.Now.Second.ToString().PadLeft(2, '0') +
DateTime.Now.Millisecond.ToString().PadLeft(4, '0');
                    string merchant_txn_datetime =
DateTime.Now.Year.ToString().PadLeft(4, '0') + "-" +
DateTime.Now.Month.ToString().PadLeft(2, '0') + "-" +
DateTime.Now.Day.ToString().PadLeft(2, '0') + " " +
DateTime.Now.Hour.ToString().PadLeft(2, '0') + ":" +
DateTime.Now.Minute.ToString().PadLeft(2, '0') + ":" +
DateTime.Now.Second.ToString().PadLeft(2, '0');
                    string product_id =
ConfigurationManager.AppSettings["product_id1"];
                    string product_name =
ConfigurationManager.AppSettings["product_name1"];
                    string txn_amount = "50";
                    string amount_parameter = "NA";
                    string txn_mode = "D";
                    string txn_type = "D";
                    string merchant_receipt_no = merchant_id +
DateTime.Now.Year.ToString().PadLeft(4, '0') +
DateTime.Now.Month.ToString().PadLeft(2, '0') +
DateTime.Now.Day.ToString().PadLeft(2, '0') +
DateTime.Now.Hour.ToString().PadLeft(2, '0') +
DateTime.Now.Minute.ToString().PadLeft(2, '0') +
DateTime.Now.Second.ToString().PadLeft(2, '0') +
DateTime.Now.Millisecond.ToString().PadLeft(4, '0');
                    string csc_share_amount = "0";
                    string pay_to_email = "a@abc.com";
                    string return_url =
ConfigurationManager.AppSettings["SUCCESS_URL"];
                    string cancel_url =
ConfigurationManager.AppSettings["pay_cancel_uri"];
```

```
                    string Currency = "INR";
                    string Discount = "0";
                    string param_1 = "NA";
                    string param_2 = "NA";
                    string param_3 = "NA";
                    string param_4 = "NA";



                    string message =
BridgePGUtil.CreateMessage(merchant_id, csc_id, merchant_txn,
merchant_txn_datetime, product_id,
                                         product_name, txn_amount,
amount_parameter, txn_mode, txn_type, merchant_receipt_no,
                                         csc_share_amount, pay_to_email,
return_url, cancel_url, Currency, Discount, param_1,
                                         param_2, param_3, param_4);

                    message =
ConfigurationManager.AppSettings["merchant_id"] + "|" + message;

                    Response.Clear();

                    StringBuilder sb = new StringBuilder();
                    sb.Append("<html>");
                    sb.AppendFormat(@"<body
onload='document.forms[""form""].submit()'>");
                    sb.AppendFormat("<form name='form' action='{0}'
method='post'>", BridgePGUtil.CreateURLappendString());
                    sb.AppendFormat("<input type='hidden' name='message'
value='{0}'>", message);
                    sb.Append("</form>");
                    sb.Append("</body>");
                    sb.Append("</html>");
                    string strpost = sb.ToString();

                    Response.Write(strpost);

                    Response.End();
               }
            }
            catch (Exception)
            {


            }

        }
```

Request parameters are adequately set in provided library to their suitable default value. These values can be over written as shown. **(BridgePGUtil.cs)**

```
    public static class BridgePGUtil
    {
        public static string CreateMessage(string merchant_id,
                                            string csc_id,
                                            string merchant_txn,
                                            string merchant_txn_datetime,
                                            string product_id,
                                            string product_name,
                                            string txn_amount,
                                            string amount_parameter,
                                            string txn_mode,
                                            string txn_type,
                                            string merchant_receipt_no,
                                            string csc_share_amount,
                                            string pay_to_email,
                                            string return_url,
                                            string cancel_url,
                                            string Currency,
                                            string Discount,
                                            string param_1,
                                            string param_2,
                                            string param_3,
                                            string param_4)
        {


            string postMessage = "merchant_id=" + merchant_id + "|"
                                + "csc_id=" + csc_id + "|"
                                + "merchant_txn=" + merchant_txn + "|"
                                + "merchant_txn_datetime=" +
merchant_txn_datetime + "|"
                                + "product_id=" + product_id + "|"
                                + "product_name=" + product_name + "|"
                                + "txn_amount=" + txn_amount + "|"
                                + "amount_parameter=" + amount_parameter +
"|"
                                + "txn_mode=" + txn_mode + "|"
                                + "txn_type=" + txn_type + "|"
                                + "merchant_receipt_no=" +
merchant_receipt_no + "|"
                                + "csc_share_amount=" + csc_share_amount +
"|"
                                + "pay_to_email=" + pay_to_email + "|"
                                + "return_url=" + return_url + "|"
                                + "cancel_url=" + cancel_url + "|"
                                + "Currency=" + Currency + "|"
                                + "Discount=" + Discount + "|"
                                + "param_1=" + param_1 + "|"
                                + "param_2=" + param_2 + "|"
                                + "param_3=" + param_3 + "|"
                                + "param_4=" + param_4 + "|";

            Crypto.privateKey =
ConfigurationManager.AppSettings["PRIVATE_KEY"];
            Crypto.publicKey =
ConfigurationManager.AppSettings["PUBLIC_KEY"];


            return Crypto.encrypt(postMessage, Crypto.publicKey,
Crypto.privateKey);
        }

}
```

Step 2 Create a response as in example file. **(pay_success.aspx.cs)**

```
public string bridgeResponseMessage = "Error ", drcResponse = "Error",
walletResponseMessage = "", merchant_txn = "", merchant_txn_datetime = "",
csc_txn = "";
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["logedin"] != "true")
Response.Redirect("login.aspx", true);

            NameValueCollection nvc = Request.Form;
            if (!string.IsNullOrEmpty(nvc["walletResponseMessage"]))
            {
                walletResponseMessage = nvc["walletResponseMessage"];



            }
            if (!string.IsNullOrEmpty(nvc["bridgeResponseMessage"]))
            {
                bridgeResponseMessage = nvc["bridgeResponseMessage"];
                Crypto.privateKey =
ConfigurationManager.AppSettings["PRIVATE_KEY"];
                Crypto.publicKey =
ConfigurationManager.AppSettings["PUBLIC_KEY"];

                drcResponse = Crypto.decrypt(bridgeResponseMessage,
Crypto.privateKey, Crypto.publicKey, true);
                string[] arr = drcResponse.Split("|".ToCharArray());

                for (int i = 0; i < arr.Length; i++)
                {
                    string[] arr2 = arr[i].Split("=".ToCharArray());
                    if (arr2[0] == "merchant_txn") merchant_txn = arr2[1];
                    TextBoxmerchant_txn.Text = merchant_txn;
                    if (arr2[0] == "merchant_txn_datetime")
merchant_txn_datetime = arr2[1];
                    TextBoxmerchant_txn_datetime.Text =
merchant_txn_datetime;
                    if (arr2[0] == "csc_txn") csc_txn = arr2[1];
                    TextBoxcsc_txn.Text = csc_txn;
                    if (arr2[0] == "merchant_id") TextBoxmerchant_id.Text =
arr2[1];
                    if (arr2[0] == "csc_id") TextBoxcsc_id.Text = arr2[1];
                    if (arr2[0] == "product_id") TextBoxproduct_id.Text =
arr2[1];
                    if (arr2[0] == "txn_amount") TextBoxtxn_amount.Text =
arr2[1];
```

```
                    if (arr2[0] == "amount_parameter")
TextBoxamount_parameter.Text = arr2[1];
                    if (arr2[0] == "txn_mode") TextBoxtxn_mode.Text =
arr2[1];
                    if (arr2[0] == "txn_type") TextBoxtxn_type.Text =
arr2[1];
                    if (arr2[0] == "merchant_receipt_no")
TextBoxmerchant_receipt_no.Text = arr2[1];
                    if (arr2[0] == "csc_share_amount")
TextBoxcsc_share_amount.Text = arr2[1];

                }
            }
        }
}
```

Functions within BridgePGUtil for BridgeAPI methods

APIs	Method
Transaction Reversal	transaction/reverse
Transaction Status	transaction/status
Refund Log	refund/log
Refund Status	refund/status
Recon Log	recon/log

For using above methods there is class already present named BridgePGUtil:

This class can be used in your code for using the above methods by simply passing the required parameters.

Sample Call For Recon Log API:

```
BridgePGUtil obj=new BridgePGUtil();

string   str   =   obj.recon_log(merchant_id,   merchant_txn,    csc_txn,
cscuser_id,       product_id,      txn_amount,       merchant_txn_datetime,
merchant_txn_status,  merchant_reciept_no);
```

4.6.2. Java

The illustrated code sample below provides the understanding of using the **java** integration kit.

Step 1 Create an encrypted payment request as in example file. **(payment.jsp)**

```
<%
            int random_num =  (int)(Math.random() * (100000 - 10000) +
10000 );
            String transactionNo     = "MTXN" + random_num;
            String receiptNo         = "RCPT" + random_num;
            String transactionAmount = request.getParameter("amt");
            String merchantId        = "11121";

            String url = request.getRequestURL().toString();
            String baseURL = url.substring(0, url.length() -
request.getRequestURI().length()) + request.getContextPath() + "/";

            BridgePgUtil bg = new BridgePgUtil();
            bg.createBridgeDefaultParameters();
            bg.addParameter("merchant_id", merchantId);
            bg.addParameter("csc_id", username);
            bg.addParameter("txn_amount", transactionAmount);
            bg.addParameter("merchant_txn", transactionNo);
            bg.addParameter("merchant_receipt_no", receiptNo);
            bg.addParameter("return_url", baseURL + "pay_success.jsp");
            bg.addParameter("cancel_url", baseURL + "pay_success.jsp");

            String encText = bg.getEncryptedParameters(bridgePrivateKey,
bridgePublicKey);
            String urlFraction = bg.getUrlFraction();

...

<form method="post" action="<%= bridgePayAddress + urlFraction %>" >
     <input type="hidden" name="message" value="<%= merchantId + "|" +
encText %>" />
     <input type="submit" value="Pay" />
</form>
```

Following is the wrapper for accessing bridge. (**BridgePGUtil.jsp**)

```
    class BridgePGUtil{
        private HashMap<String, String> parameters;

        public BridgePGUtil(){
            this.parameters = new HashMap<String, String>();
        }

        public Map<String, String> getBridgeDefaultParameters(){
            return this.parameters;
        }

        public boolean addParameter(String key, String value){
            parameters.put(key, value);
            return true;
        }

        public String getParameterString(){
            String ret = "";
            for (Map.Entry<String, String> entry : parameters.entrySet()) {
                ret +=  entry.getKey().trim() + "=" + entry.getValue().trim()
+ "|";
            }
            return ret;
        }

        public String getEncryptedParameters(String bridgePrivateKey,String
bridgePublicKey){
            BridgeCryptor bc  = BridgeFactory.getBridgeCryptor();
            bc.setKeys(bridgePrivateKey, bridgePublicKey);
            return bc.encrypt(getParameterString());
        }

        public String getUrlFraction(){
            SimpleDateFormat sdf = new SimpleDateFormat("yyMMddHHmmss");
            String date_str = sdf.format(new Date()).toString();
            long date_str_long = Long.parseLong(date_str);

            return date_str_long * 883 + (1000 - 883) + "";
        }

        public Map<String, String> createBridgeDefaultParameters(){
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
            String date_str = sdf.format(new Date()).toString();
            int rand =  (int)(Math.random() * (100000 - 10000) + 10000 );
            parameters.put("merchant_id"          ,  "11121"            );
            parameters.put("merchant_txn"         ,  "M" + rand         );
            parameters.put("merchant_txn_datetime", date_str            );
            parameters.put("product_id"           ,  "1112101"          );
            parameters.put("product_name"         ,  "my product"       );
            parameters.put("txn_amount"           ,  "100"              );
            parameters.put("amount_parameter"     ,  "NA"               );
            parameters.put("txn_mode"             ,  "D"                );
            parameters.put("txn_type"             ,  "D"                );
            parameters.put("merchant_receipt_no"  ,  "RCPT" + rand      );
            parameters.put("csc_share_amount"     ,  "0"                );
            parameters.put("pay_to_email"         ,  "na"               );
            parameters.put("return_url"           ,  "na"               );
            parameters.put("cancel_url"                 ,  "na"                   );
            parameters.put("Currency"                   ,  "INR"                  );
            parameters.put("Discount"                   ,  "0"                    );
            parameters.put("param_1"                    ,  "NA"                   );
            parameters.put("param_2"                    ,  "NA"                   );
            parameters.put("param_3"                    ,  "NA"                   );
            parameters.put("param_4"                    ,  "NA"                   );

            return this.parameters;
        }
    }
```

Step 2 Process pay response to get status of payment as in sample file. **(payment_response.jsp)**

```
<%
 String enc_vals = request.getParameter("bridgeResponseMessage");
 bridgeutil.BridgeCryptor bg = bridgeutil.BridgeFactory.getBridgeCryptor();

 bg.setKeys(bridgePrivateKey, bridgePublicKey);

 String dec_vals = bg.decrypt(enc_vals);
 %>
```

Functions within BridgePGUtil for BridgeAPI methods

APIs	Method
Transaction Reversal	transaction/reverse
Transaction Status	transaction/status
Refund Log	refund/log
Refund Status	refund/status
Recon Log	recon/log

For using above methods there is class already present named BridgePGUtil:

This class can be used in your code for using the above methods by simply passing the required parameters.

4.6.3. PHP

The illustrated code sample below provides the understanding of using the **php** integration kit.

Step 1 Create an encrypted payment request as in example file. (**payment.php**)

```
<?php
require_once 'includes/BridgePGUtil.php';

$bconn = new BridgePGUtil();
$p = array(
//   'csc_id' => $_SESSION['user']['username'],
     'csc_id' => $_SESSION['username'],
     'merchant_id' => '11121',
     'merchant_receipt_no' => 'Recpt#' . rand(100,999),
     'txn_amount' => $_GET['amt'],
     'return_url' => 'https://merchant.csccloud.in/pay_success.php',
     'cancel_url' => 'https://merchant.csccloud.in/pay_success.php',
     'product_id' => '1112101',
     'merchant_txn' => 'MW' . rand(1000, 9999)
);

$bconn->set_params($p);
$enc_text = $bconn->get_parameter_string();
$frac = $bconn->get_fraction();
?>


  <form method="post" action="https://pay.csccloud.in/v1/payment/<?php echo
$frac;?>">
        <input type="hidden" name="message" value="<?=$enc_text;?>" />
        <input type="submit" value="Pay" />
  </form>
```

Following is the wrapper for accessing bridge. **(BridgePGUtil.php)**

```
include_once __DIR__."/BridgePG.php";
class BridgePGUtil {

    private $bridgePG;
    private $bridge_parameters;

    public function __construct(){
        $this->bridgePG = new BridgePG();
        $this->bridge_parameters = self::get_default_parameters();
    }

    private static function get_default_parameters(){
        $brj_params = array(
            'merchant_id'             => '11121',
            'merchant_txn'            => 'MDM101'.time().
rand(10,99),//'2016-06-21 18:11:58',
            'merchant_txn_datetime'  => date('Y-m-d H:i:s'),//'2016-06-21
18:11:58',
            'product_id'              => '1112101',
            'product_name'            => 'my product',
            'txn_amount'              => '100',
            'amount_parameter'        => 'NA',
            'txn_mode'                => 'D',
            'txn_type'                => 'D',
            'merchant_receipt_no'     => '2016-06-21 18:11:58',
            'csc_share_amount'        => '0',
            'pay_to_email'            => 'a@abc.com',
            'return_url'              => '',
            'cancel_url'              => '',
            'Currency'                => 'INR',
            'Discount'                => '0',
            'param_1'                 => 'NA',
            'param_2'                 => 'NA',
            'param_3'                 => 'NA',
            'param_4'                 => 'NA'
        );
        return $brj_params;
    }

    public function set_params($params){
        foreach($params as $p => $v){
            $this->bridge_parameters[$p] = $v;
        }
    }

    public function get_parameter_string(){
        $message_text = '';
        foreach($this->bridge_parameters as $p => $v ){
            $message_text .= $p . '=' . $v . '|';
        }
        $message_cipher = $this->bridgePG-
>encrypt_message_for_wallet($message_text, FALSE);
        return $this->bridge_parameters['merchant_id'] . '|' .
$message_cipher;
    }

    public function get_bridge_message(){
        $d = "Invalid Bridge message";
        if($_POST['bridgeResponseMessage']){
            $c = $this->bridgePG-
>decrypt_wallet_message($_POST['bridgeResponseMessage'], $d, FALSE);
        }
        return $d;
    }

    public function get_fraction($ddhhmm = ""){
        $time_format = "ymdHis";
        $algo_num    = "883";
        if(!$ddhhmm)
            $ddhhmm = date($time_format, time());
        $frac = $this->large_op1($ddhhmm, $algo_num );
        $frac = $this->large_op2($frac, "" . (1000 - $algo_num) );
        return $frac;
    }

    public function large_op1($n0, $x0){
        $n = '' . $n0;
        $x = '' . $x0;
        $sz = strlen('' . $n);
        $vals = array();
        $tens = 0;
        for($i = 0; $i < $sz; $i++ ){
            $d = $n[$sz - $i - 1];
            $res = $d * $x + $tens;
            $ones = $res % 10;
            $tens = (int)($res / 10);
            array_unshift($vals, $ones);
        }
        if($tens > 0)
            array_unshift($vals, $tens);
        return implode("", $vals);
    }

    public function large_op2($n0, $x0){
        $n = '' . $n0;
        $x = '' . $x0;
        $sz = strlen('' . $n);
        $vals = array();
        $tens = 0;
        for($i = 0; $i < $sz; $i++ ){
            $d = $n[$sz - $i - 1];
            if($i == 0)
                $res = $d + $x;
            else
                $res = $d + $tens;
            $ones = $res % 10;
            $tens = (int)($res / 10);
            array_unshift($vals, $ones);
        }
        if($tens > 0)
            array_unshift($vals, $tens);
        return implode("", $vals);
    }
}
```

Step 2 Process pay response to get status of payment as in sample file. **(payment_response.php)**

```
<?php
require_once 'includes/ BridgePGUtil.php';

$bconn = new BridgePGUtil ();
$bridge_message = $bconn->get_bridge_message();
?>


Encrypted Values:  <?php echo $_POST['bridgeResponseMessage']; ?>

Decrypted Values: <?php $bridge_message; ?>
```

Functions within BridgePGUtil for BridgeAPI methods

APIs	Method
Transaction Reversal	transaction/reverse
Transaction Status	transaction/status
Refund Log	refund/log
Refund Status	refund/status
Recon Log	recon/log

For using above methods there is class already present named BridgePGUtil:

This class can be used in your code for using the above methods by simply passing the required parameters.

5. Transaction Reconciliation

During transactions there might be instances when the control is lost while the response is being awaited from the Bridge system due to reasons like connectivity issue/ customer browser being closed, thus causing a failure of the transactions.

In order to overcome such issues, the reconciliation services are used. Bridge PG supports the below methods for transaction reconciliation and merchant settlement:

a. **Instant Recon**

   This service can be used by merchants to submit transaction success logs as well as refund request on the gateway online, which enables to either complete or drop the transaction on the same day.

b. **Data Exchange through SFTP**

   Merchants who are not ready to submit real time response for a transaction executed on the Bridge PG may utilize the file based transfers on the SFTP account created to perform next day reconciliation.

Merchants interested to render services in real time are encouraged to use the Instant Recon service.