# GPA – GeM Pool Account Integration with Banks

Technical Design Document

Government e Marketplace

## Table of Contents

# 1. Introduction

## 1.1. Background and Context

GeM Pool Account is a special purpose bank account opened, operated and controlled exclusively by each NPAE (Non-Public Financial Management System (PFMS) Agencies/Entities) for the purpose of crediting projected value of the contracts/supply orders in to the account and for subsequently making timely payments to the suppliers on successful supply and acceptance of goods & services ordered on GeM against supply orders placed by the NPAE on GeM.

The GeM Pool account has two models

1) Challan Model -
2) Non Challan Model

An NPAE can open either a Challan GPA or Non-Challan GPA with any of the GPA integrated banks

- **Challan model**
- Funds are transferred in The GeM Pool account after the demand/intent has been finalized
- A Challan is generated on GeM using which the Buyer funds the required amount in GeM Pool account.

- **Non Challan Model**
- A Floating amount based on Procurement forecast is maintained by Buyer in Pool account
- No challan is generated as pool account is already funded.

## 1.2. Document References

| # | Document Name | Description |
|---|---|---|
| 1. | GPA SOP | Outlines the GPA solution and operating procedure |
| 2. | GPA Office Memorandum issued by Department of expenditure | Outlines the directives of Department of expenditure for GPA |

This Technical Design document elaborates the *GeM Pool Account Integration solution.*

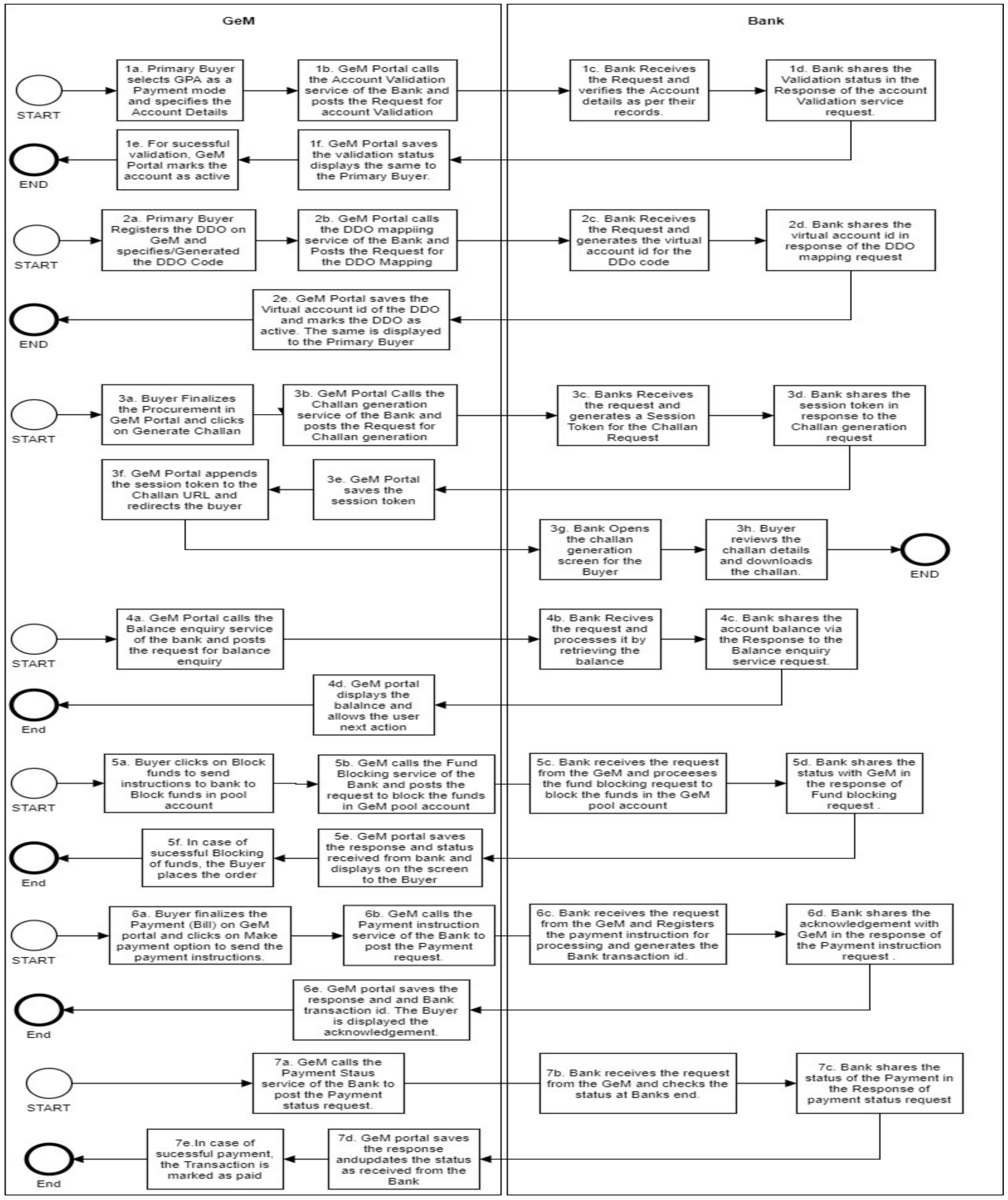
## 1.3. Abbreviation

| **Abbreviation** | **Description** |
|---|---|
| CRAC | Consignee receipt and acceptance certificate |
| Gem | Government eMarketplace |
| DP | Delivery Period |
| JSON | JavaScript Object Notation |
| GPA | GeM Pool Account |
| FMS | Financial Management System |
| PRC | Provisional receipt certificate |
| NPAE | Non-Public Financial Management System (PFMS) Agencies/Entities |
| RA | Reverse Auction |
| SBI | State Bank of India |
| SLA | Service level agreement |
| SOP | Standard operating procedure |
| T&C | Terms and condition |

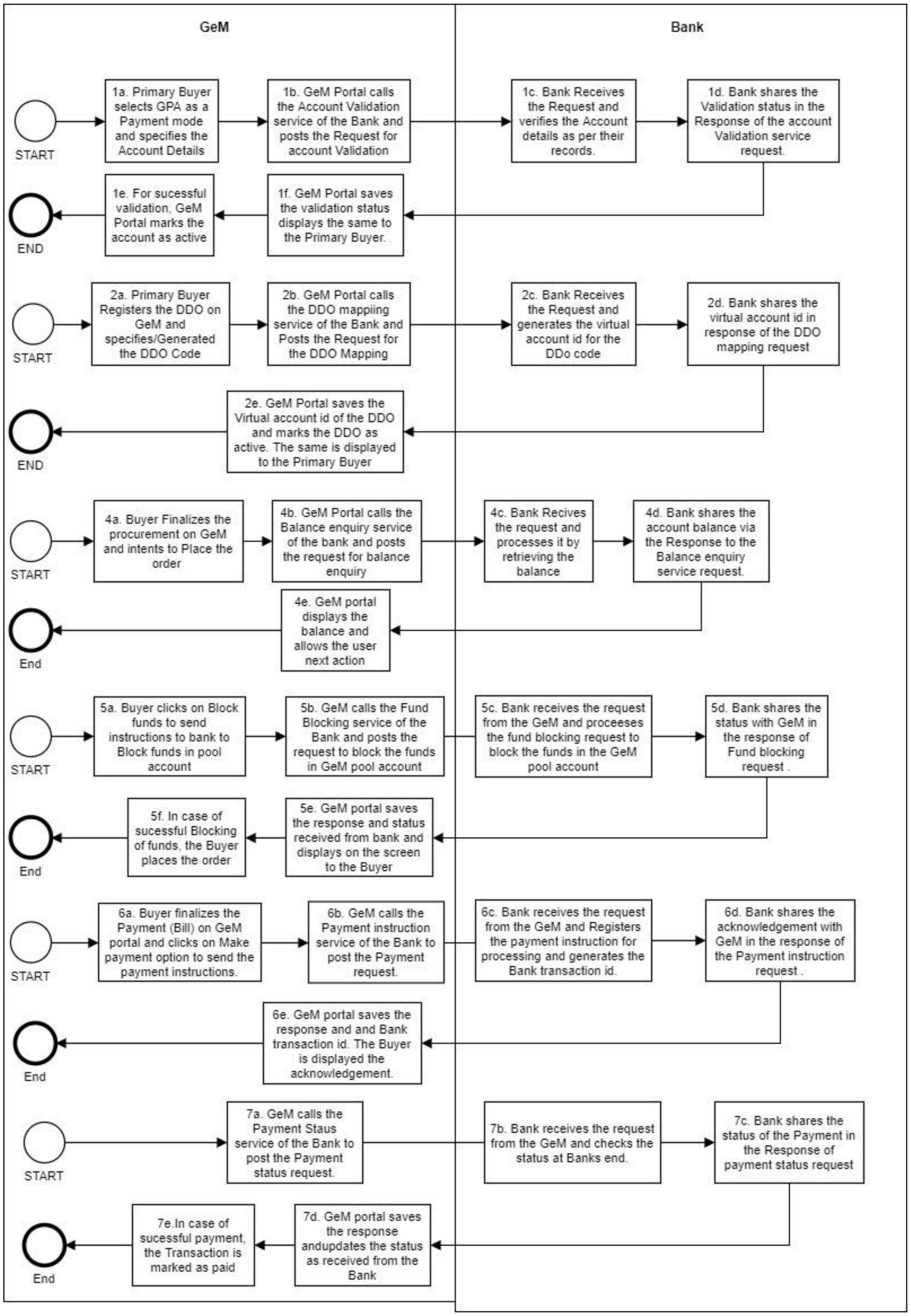
# 2. Solution Overview

## 2.1. Introduction to GeM Pool Account – Challan Model

The following figure provides a high-level context of the GeM Pool Account – Challan Model.


GeM
Bank
START
1a. Primary Buyer selects GPA as a Payment mode and specifies the Account Details
1b. GeM Portal calls the Account Validation service of the Bank and posts the Request for account Validation
1c. Bank Receives the Request and verifies the Account details as per their records.
1d. Bank shares the Validation status in the Response of the account Validation service request.
1f. GeM Portal saves the validation status displays the same to the Primary Buyer.
1e. For sucessful validation, GeM Portal marks the account as active
END
START
2a. Primary Buyer Registers the DDO on GeM and specifies/Generated the DDO Code
2b. GeM Portal calls the DDO mappiing service of the Bank and Posts the Request for the DDO Mapping
2c. Bank Receives the Request and generates the virtual account id for the DDo code
2d. Bank shares the virtual account id in response of the DDO mapping request
2e. GeM Portal saves the Virtual account id of the DDO and marks the DDO as active. The same is displayed to the Primary Buyer
END
START
3a. Buyer Finalizes the Procurement in GeM Portal and clicks on Generate Challan
3b. GeM Portal Calls the Challan generation service of the Bank and posts the Request for Challan generation
3c. Banks Receives the request and generates a Session Token for the Challan Request
3d. Bank shares the session token in response to the Challan generation request
3e. GeM Portal saves the session token
3f. GeM Portal appends the session token to the Challan URL and redirects the buyer
3g. Bank Opens the challan generation screen for the Buyer
3h. Buyer reviews the challan details and downloads the challan.
END
START
4a. GeM Portal calls the Balance enquiry service of the bank and posts the request for balance enquiry
4b. Bank Recives the request and processes it by retrieving the balance
4c. Bank shares the account balance via the Response to the Balance enquiry service request.
4d. GeM portal displays the balaInce and allows the user next action
End
START
5a. Buyer clicks on Block funds to send instructions to bank to Block funds in pool account
5b. GeM calls the Fund Blocking service of the Bank and posts the request to block the funds in GeM pool account
5c. Bank receives the request from the GeM and proceeses the fund blocking request to block the funds in the GeM pool account
5d. Bank shares the status with GeM in the response of Fund blocking request .
5e. GeM portal saves the response and status received from bank and displays on the screen to the Buyer
5f. In case of sucessful Blocking of funds, the Buyer places the order
End
START
6a. Buyer finalizes the Payment (Bill) on GeM portal and clicks on Make payment option to send the payment instructions.
6b. GeM calls the Payment instruction service of the Bank to post the Payment request.
6c. Bank receives the request from the GeM and Registers the payment instruction for processing and generates the Bank transaction id.
6d. Bank shares the acknowledgement with GeM in the response of the Payment instruction request .
6e. GeM portal saves the response and and Bank transaction id. The Buyer is displayed the acknowledgement.
End
START
7a. GeM calls the Payment Staus service of the Bank to post the Payment status request.
7b. Bank receives the request from the GeM and checks the status at Banks end.
7c. Bank shares the status of the Payment in the Response of payment status request
7d. GeM portal saves the response andupdates the status as received from the Bank
7e.In case of sucessful payment, the Transaction is marked as paid
End

## 2.2. Introduction to GeM Pool Account – Non Challan Model


GeM
Bank
START
1a. Primary Buyer selects GPA as a Payment mode and specifies the Account Details
1b. GeM Portal calls the Account Validation service of the Bank and posts the Request for account Validation
1c. Bank Receives the Request and verifies the Account details as per their records.
1d. Bank shares the Validation status in the Response of the account Validation service request.
1f. GeM Portal saves the validation status displays the same to the Primary Buyer.
1e. For sucessful validation, GeM Portal marks the account as active
END
START
2a. Primary Buyer Registers the DDO on GeM and specifies/Generated the DDO Code
2b. GeM Portal calls the DDO mappiing service of the Bank and Posts the Request for the DDO Mapping
2c. Bank Receives the Request and generates the virtual account id for the DDo code
2d. Bank shares the virtual account id in response of the DDO mapping request
2e. GeM Portal saves the Virtual account id of the DDO and marks the DDO as active. The same is displayed to the Primary Buyer
END
START
4a. Buyer Finalizes the procurement on GeM and intents to Place the order
4b. GeM Portal calls the Balance enquiry service of the bank and posts the request for balance enquiry
4c. Bank Recives the request and processes it by retrieving the balance
4d. Bank shares the account balance via the Response to the Balance enquiry service request.
4e. GeM portal displays the balance and allows the user next action
End
START
5a. Buyer clicks on Block funds to send instructions to bank to Block funds in pool account
5b. GeM calls the Fund Blocking service of the Bank and posts the request to block the funds in GeM pool account
5c. Bank receives the request from the GeM and proceeses the fund blocking request to block the funds in the GeM pool account
5d. Bank shares the status with GeM in the response of Fund blocking request .
5e. GeM portal saves the response and status received from bank and displays on the screen to the Buyer
5f. In case of sucessful Blocking of funds, the Buyer places the order
End
START
6a. Buyer finalizes the Payment (Bill) on GeM portal and clicks on Make payment option to send the payment instructions.
6b. GeM calls the Payment instruction service of the Bank to post the Payment request.
6c. Bank receives the request from the GeM and Registers the payment instruction for processing and generates the Bank transaction id.
6d. Bank shares the acknowledgement with GeM in the response of the Payment instruction request .
6e. GeM portal saves the response and and Bank transaction id. The Buyer is displayed the acknowledgement.
End
START
7a. GeM calls the Payment Staus service of the Bank to post the Payment status request.
7b. Bank receives the request from the GeM and checks the status at Banks end.
7c. Bank shares the status of the Payment in the Response of payment status request
7d. GeM portal saves the response andupdates the status as received from the Bank
7e.In case of sucessful payment, the Transaction is marked as paid
End

## 2.3. Shared Services Infrastructure

The core of the GPA payment solution is the Shared Services Infrastructure that provides several enabling technical functionalities to deliver payment services. This includes Security, Authentication and Authorization, account Validation, DDO Mapping, Challan generation, Fund Blocking/unblocking, Payment instructions, etc.

## 2.4. Pre Requisites for GPA Integration

| S.No | Details Required | Remarks |
|---|---|---|
| 1 | Whitelisting of GeM IP by the Bank | Bank would whitelist the UAT and Production Public IP of GeM |
| 2 | Whitelisting of Bank IP by GeM | Bank would share the UAT and Production Public IP with GeM for whitelisting. The Port number should be 443 |

## 2.5. Service Summary

The GPA integration consists of the following shared services that would be integrated with the Banks to deliver the GPA functionality.

| Web-service Name | Hosted by | Consumed by | Applicable in GPA Model | Description |
|---|---|---|---|---|
| Account validation | Bank | GeM | 1) Challan<br>2)Non Challan | This Service would be used to validate the Account details registered by the Buyer on GeM portal |
| DDO Mapping | Bank | GeM | 1) Challan<br>2)Non Challan | This Service would be used to map the DDO code with the GPA account and to generate the virtual account for each DDO |

| | | | | |
|---|---|---|---|---|
| Challan Generation | Bank | GeM | 1) Challan | This Service would be used to Generate the challan by the Buyer to fund the GPA account. |
| Balance Enquiry | Bank | GeM | 1) Challan<br>2)Non Challan | 1) Challan Model - This Service would be used to enquire on the sucessful credit of funds in GeM Pool account as per the challan generated by Buyer.<br>2) Non Challan Model:This Service would be used to enquire on the available funds in GeM Pool account. |
| Fund Blocking/Unblocking | Bank | GeM | 1) Challan<br>2)Non Challan | This Service would be used to Block or Unblock the funds in the GeM Pool account of the Buyer |
| Payment Instructions | Bank | GeM | 1) Challan<br>2)Non Challan | This Service would be used to initiate the Payment to the supplier from the Blocked funds in the GeM Pool account |
| Payment Status | Bank | GeM | 1) Challan<br>2)Non Challan | This Service would be used to enquire on the status of the payment initiated by the Buyer. |

# 3. Service Description

## 3.1. Pool Account Validation

### 3.1.1. Service Details

| | |
|---|---|
| Service Name | Pool Account validation |
| Hosted By | Bank |
| Consumed By | Gem |
| Purpose of the Service | Purpose of this interface is to validate the pool account number entered Buyers in GeM on-boarding form provided to them. On receipt the details from Buyer & after successful validation in GeM portal through this service from the Bank, GeM will allow Buyer to move ahead with registration process. |
| Usage in GPA | 1) Challan Model<br>2) Non Challan Model |
| Method Of Integration | RESTFul services would be integrated as Json Structure |
| Service Availability Window Processing | The service should be available throughout year 24*7 |
| Where is it Invoked | The Pool account validation |

| | |
|---|---|
| Process Summary | GPA account Validation on primary Buyer Registration<br>• Primary Buyer selects GPA as the Payment Mode and specifies the following details of the GeM Pool account – Bank Name, Account Number, IFSC code, Account Holder name, Account holder email id<br>• The GeM Portal calls the Account validation service of the Bank and posts the Request<br>• Banks validate the account details received in Request parameters and Post the response<br>• GeM Portal on receipt of the successful response, makes the account active in the Buyer profile on GeM portal. |
| Process Output | • Validation status – Success or Fail<br>• Account Holder email status – Success, Fail, Not Available<br>• Mode of operation – Challan, Non challan<br>• Account Holder name as in Bank Records |
| Participating Roles | • Primary Buyer<br>• GeM Portal<br>• Bank |

|  |  |
|---|---|
| Other Notes | 1) GeM Pool account validation is Mandatory<br>2) This will be a synchronous call; So GeM Portal will not allow the user to continue with next activity till GeM gets the response back from Bank.<br>3) Once GeM Portal receives the response from bank Service, immediately same will be indicated on the screen.<br>4) The Bank would validate the following information at their end – Account number, IFSC code and Account holder email.<br>5) The Bank would return the account validation status, account holder email matching status, Account holder email, Account Holder Name and Mode of operation of the Account |

### 3.1.2. Input parameters

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| Body | | | | | |
| gemReqID | 80 | String | TRUE | Unique Request ID generated by GeM | body |
| OrgCode | 20 | String | TRUE | Org code shared by GeM | body |
| buyerPoolAcctNo | 30 | String | TRUE | GeM Pool account number | body |
| IfsCode | 11 | String | TRUE | IFSC Code of Bank | body |
| accountHolderName | 120 | String | TRUE | without any special characters | Body |
| accountHolderEmail | 100 | String | True | Email id specified by Primary user | body |

### 3.1.3. Output fields

| Field Name | Length | Type | Mandatory | Values | Parent Element |
|---|---|---|---|---|---|
| Body | | | | | |
| gemReqID | 80 | String | TRUE | Request ID generated by GeM | Body |
| bankTransID | 40 | String | TRUE | Bank Transaction ID | Body |
| OrgCode | 20 | String | TRUE | Org code shared by GeM | Body |
| buyerPoolAcctNo | 30 | String | TRUE | GeM Pool account number | Body |
| IfsCode | 11 | String | TRUE | IFSC Code of Bank | body |
| accountHolderName | 120 | String | TRUE | without any special characters | Body |
| modeOfOperation | 1 | String | TRUE | C – Challan, N – Non Challan | Body |
| accountStatus | 1 | String | TRUE | V – Valid, I – Invalid<br>The Status of the account to be returned by Bank after matching IFSC code and account number | Body |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| accountHolderEmailStatus | 1 | String | TRUE | **V – Valid, I – Invalid, N – Not available,** **B - When email is not validated in bank database but correct email is shared in response.** | Body |
| Status | 1 | String | TRUE | S – Success, F-Failed | Body |
| remarks | 200 | String | TRUE | Banks to share the reason of failure | Body |
| accountHolderEmail | 100 | String | True | For accountHolderEmailStatus as 'V','I', 'N" - Email id received in request. For accountHolderEmailStatus as 'B'- Email id as in bank records | body |

## 3.2. DDO Mapping

### 3.2.1. Service details

|  |  |
|---|---|
| Service Name | DDO Mapping |
| Hosted By | Bank |

| | |
|---|---|
| Consumed By | GeM |
| Purpose of the Service | Purpose of this interface is to share the DDO code specified by the primary Buyer during Registration. The Bank would generate the virtual account id for each DDo code and share with the GeM Portal via the response of this service. |
| Usage in GPA | 1) Challan Model<br>2) Non Challan Model |
| Method Of Integration | RESTFul services would be integrated as Json Structure |
| Service Availability Window Processing | The service should be available throughout year 24*7 |
| Process Summary |  Primary Buyer would create the DDO user on GeM Portal<br> Primary Buyer Maps the verified GeM Pool Account to the DDO<br> Primary buyer specifies the Unique DDO code. In cases, DDO code is not available; the Primary Buyer can generate a unique code on GeM Portal.<br> GeM Portal calls the DDO mapping service of the Bank and posts the Request. The Request parameters also include the DDO code.<br> Bank generates a virtual account id for the DDO and maps it with the GeM Pool account.<br> Bank shares the DDO virtual account id back in the Response to the GeM Portal<br> GeM Saves the Virtual account id for the DDO. |
| Process Output |  Status – Success or Fail<br> DDO Virtual account id (DDO Registration id) |

| | |
|---|---|
| Participating Roles | Primary Buyer<br>GeM Portal<br>Bank |
| Other Notes | 1) DDO Mapping is mandatory for Buyer to complete the Registration of GeM pool account .<br>2) This will be a synchronous call; So GeM Portal will not allow the user to continue with next activity till GeM gets the response back from Bank.<br>3) Once GeM Portal receives the response from bank Service, immediately same will be indicated on the screen. |

### 3.2.2. Input parameters

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| Body | | | | | |
| gemReqID | 80 | String | TRUE | Request ID generated by GeM | Body |
| orgCode | 20 | String | TRUE | Org code shared by GeM | Body |
| buyerPoolAcctNo | 30 | String | TRUE | GeM Pool account number | Body |
| ifsCode | 11 | String | TRUE | IFSC Code of Bank | Body |
| accountHolderName | 120 | String | OPTIONAL | without any special characters | Body |
| buyerID | 120 | String | TRUE | Session id of the user | Body |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| ddoCode | 10 | String | TRUE | Dynamic code generated by GeM | Body |

### 3.2.3. Output fields

| Field Name | Length | Type | Mandatory | Values | Parent Element |
|---|---|---|---|---|---|
| Body |  |  |  |  |  |
| gemReqID | 80 | String | TRUE | Request ID generated by GeM | Body |
| bankTransID | 40 | String | TRUE | Bank Transaction ID | Body |
| orgCode | 20 | String | TRUE | Org code shared by GeM | Body |
| buyerPoolAcctNo | 30 | string | TRUE | GeM pool account number | Body |
| ifsCode | 11 | String | TRUE | IFSC Code of Bank | Body |
| accountHolderName | 120 | String | OPTIONAL | without any special characters | Body |
| buyerID | 120 | String | TRUE | Session id of the user | Body |
| ddoCode | 10 | String | TRUE | Dynamic code generated by GeM | Body |
| ddoRegistrationNo | 20 | String | TRUE | Virtual account number generated by Bank | Body |
| Status | 1 | String | TRUE | S – Success, F – Failed | Body |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Remarks | 200 | String | TRUE | Banks to share the reason of failure | Body |

## 3.3. Challan Generation

### 3.3.1. Service Details

|  |  |
|---|---|
| Service Name | Challan Generation |
| Hosted By | Bank |
| Consumed By | GeM |
| Purpose of the Service | Purpose of this interface is to Generate the Challan for the Purchase to be done on GeM by the Buyer. The Buyer would use the generated Challan to fund the pool account. |
| Usage in GPA | Challan Model |
| Method Of Integration | RESTFul services would be integrated as Json Structure |
| Service Availability Window Processing | The service should be available throughout year 24*7 |

|  |  |
|---|---|
| Process Summary | Buyer would finalize the Purchase on GeM portal via any of the available procurement modes – Direct Purchase, Bid, RA, Softbid, etc.<br><br>Before generating the final order/contract, Buyer would click on the Generate challan Option available on the GeM Portal<br><br>GeM portal would call the Challan generation api of the Bank and post the Request.<br><br>Bank would generate the secure token and send the same to GeM portal in Response.<br><br>GeM portal would append the secure token received from the Bank to the URL of Challan generation api and redirect the Buyer to the Bank screen.<br><br>Bank would display the Challan details to the Buyer. Bank would use the details received in the Request to populate the challan form. Bank can also display any additional details or take additional inputs from the Buyer as per their agreement with the Buyer organization.<br><br>Buyer would review the challan details and download or print the challan. |
| Process Output | Status – Success or Fail<br><br>Secure Token |
| Participatin g Roles | Buyer<br><br>GeM Portal<br><br>Bank |

| | |
|---|---|
| Other Notes | 1) Challan generation is mandatory for Challan based GeM Pool account.<br>2) This will be a synchronous call; So GeM Portal will not allow the user to continue with next activity till GeM gets the response back from Bank.<br>3) Once GeM Portal receives the response from bank Service, immediately same will be indicated on the screen.<br>4) Token validity is for maximum 5 minutes. After 5 minutes current token will expire. It is the bank responsibility to give proper information to user about expired token.<br>5) Post expiry of challan, Buyer can initiate the challan generation again. The Bank would generate a new token and display the challan details to the Buyer.<br>6) The Unique reference number (URN) generated by GeM is a unique identifier for the particular procurement on GeM.The Bank should generate the Virtual account number for each Unique reference number(URN) and accept the payment in the Virtual account number<br>7) The bank should ensure that the Payment received in virtual account is equal to the challan amount.<br>8) The BlockReqid generated by GeM is a unique identifier for a challan/Block payload |

### 3.3.2. Input parameters

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| Body | N/A | N/A | TRUE | | N/A |
| gemReqID | 80 | String | TRUE | GeM Requested ID | Body |
| gemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM. This id would be unique to a procurement on GeM | Body |
| blockReqId | 50 | Numeric | TRUE | Unique id generated by GeM for a Challan/Fund Blocking. This id would be unique for a challan | Body |
| Date | 25 | Datetime | TRUE | Date at which challan request initiated DD-MM-YYYY HH:MM:SS | Body |
| orgCode | 20 | String | TRUE | Org code shared by GeM | Body |
| Amount | 15.2 | Numeric | TRUE | Amount for which chalan need to be generated | Body |
| ddoRegistrationNo | 20 | String | TRUE | | Body |
| ddoName | 100 | String | TRUE | Name of the ddo | Body |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Topup | 1 | String | TRUE | Y-Yes, N-No, for confirming whether chalan is generated first time or not. For first time, value would be 'N' and for all subsequent times, the value would be 'Y' | Body |
| ddoCode | 10 | String | TRUE |  | body |

### 3.3.3. Output parameters

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| Body | N/A | N/A | TRUE |  | N/A |
| gemReqID | 80 | String | TRUE | GeM Requested ID | body |
| gemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | body |
| blockReqId | 50 | Numeric | TRUE | Unique id generated by GeM for a Challan/Fund Blocking. This id would be unique for a challan | Body |
| Amount | 15.2 | Numeric | TRUE | Amount for which chalan need to be generated | body |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Date | 25 | Datetime | TRUE | Date at which chalan request initiated DD-MM-YYYY HH:MM:SS | body |
| orgCode | 20 | String | TRUE | Org code shared by GeM | body |
| ddoRegistrationNo | 20 | String | TRUE |  | Body |
| ddoName | 100 | String | TRUE | Name of the ddo | Body |
| topup | 1 | String | TRUE | Y-Yes, N-No, for confirming whether chalan is generated first time or not | Body |
| status | 1 | String | TRUE | S – Success, F-Failure | Body |
| remarks | 200 | String | TRUE |  | Body |
| token | 100 | String | TRUE | Token generated by Bank | Body |
| ddoCode | 10 | String | TRUE |  | Body |

## 3.4. Balance Inquiry

### 3.4.1. Service Details

|  |  |
|---|---|
| Service Name | Balance Enquiry |
| Hosted By | Bank |
| Consumed By | GeM |

| | |
|---|---|
| Purpose of the Service | Purpose of this interface is to enquire on the Pool account balance.<br>In challan mode, the service would return the balance for a particular challan id of a Unique reference number(URN) received in the Request.<br>In Non Challan mode, the service would return the balance in the GeM Pool account. |
| Usage in GPA | 1) Challan Model<br>2) Non Challan Model |
| Method Of Integration | RESTFul services would be integrated as Json Structure |
| Service Availability Window Processing | The service should be available throughout year 24*7 |
| Process Summary |  Buyer would enquire on the balance to check if the funding has been made in challan model or if the Balance is available in non challan model to fund the Purchase<br> The GeM Portal would Call the service and post the Request. Bank would Check the balance and return the same via the response to the service.<br> On receiving the success, the Gem portal would allow the Buyer to proceed ahead. |
| Process Output |  Status – Success or Fail<br> Available Balance |
| Participating Roles |  Buyer<br> GeM Portal<br> Bank |

| | |
|---|---|
| Other Notes | 1) Balance enquiry check is mandatory before Funds can be blocked.<br>9) This will be a synchronous call; So GeM Portal will not allow the user to continue with next activity till GeM gets the response back from Bank.<br>10) Once GeM Portal receives the response from bank Service, immediately same will be indicated on the screen. |

### 3.4.2. Input parameters

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| Body | | | | | |
| gemReqID | 80 | String | TRUE | Transaction ID generated by GeM | body |
| gemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | body |
| blockReqId | 50 | Numeric | TRUE | Unique id generated by GeM for a Challan/Fund Blocking. This id would be unique for a challan | Body |
| orgCode | 20 | String | TRUE | Org code shared by GeM | body |
| buyerPoolAcctNo | 30 | String | TRUE | GeM pool account number | body |
| ifsCode | 11 | String | TRUE | IFSC Code of Bank | body |

| | | | | | |
|---|---|---|---|---|---|
| accountHolderName | 120 | String | TRUE | without any special characters | body |
| ddoRegistrationNo | 20 | String | TRUE | Virtual account number generated by Bank | body |

### 3.4.3. Output fields

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| Body | | | | | |
| gemReqID | 80 | String | TRUE | Transaction ID generated by GeM | Body |
| gemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | Body |
| blockReqId | 50 | Numeric | TRUE | Unique id generated by GeM for a Challan/fund blocking. This id would be unique for a challan/block payload | Body |
| bankTransID | 40 | String | TRUE | Bank Transaction ID | Body |
| orgCode | 20 | String | TRUE | Org code shared by GeM | Body |
| BuyerTreasuryCode | 20 | String | OPTIONAL | State Treasury Code | body |
| buyerPoolAcctNo | 30 | String | TRUE | GeM pool account number | Body |

| | | | | | |
|---|---|---|---|---|---|
| ddoRegistrationNo | 20 | String | TRUE | Virtual account number generated by Bank | Body |
| ifsCode | 11 | String | TRUE | Ifsc code of Bank | Body |
| accountHolderName | 120 | String | TRUE | | Body |
| availableBalance | 15.2 | String | TRUE | Balance against a URN | Body |
| Status | 1 | String | TRUE | S – Success, F – Failed | Body |
| Remarks | 200 | String | TRUE | Banks to share the reason of failure | Body |

## 3.5. Block-Unblock Amount

### 3.5.1. Service Details

| | |
|---|---|
| Service Name | Block-Unblock Request |
| Hosted By | Bank |
| Consumed By | GeM |
| Purpose of the Service | Purpose of this interface is to send the Blocking or Unblocking request from GeM portal to Bank. Banks on receipt of the Request would Block the funds in the pool account or Unblock the funds. |
| Usage in GPA | 1) Challan Model<br>2) Non Challan Model |
| Method Of Integration | RESTFul services would be integrated as Json Structure |

|  |  |
|---|---|
| Service Availability Window | The service should be available throughout year 24*7 |
| Processing Process Summary | Fund Blocking By Buyer<br>- Buyer would choose to block the Funds while finalizing the Order on GeM<br>- GeM would call the Fund Blocking-Unblocking service and Post the Request to Block the funds<br>- Bank would process the request and Block the funds in the pool account as per the instructions received in the Request.<br>- Bank would generate the fund block id for blocking of the funds and share the same in the Response<br>- GeM on receipt of successful blocking from Bank would allow the Buyer to place the order.<br>Fund Un-Blocking By Buyer<br>- The Unblocking request would be applicable in case of Order cancellation by Buyer, order decline by seller or release of excess or unutilized funds.Additionally the Fund Unblocking would also be used to unblock the Top Up Blocking or excess remaining from the Top Up blocking.<br>- GeM would call the Fund Block-unblock service of the Bank and post the Request for unblocking. The Request would include the fund block id received from the bank.<br>- Bank would process the request and Unblock the funds from GeM Pool account and share the status in the Response.<br>- GeM would mark the Transaction as completed on receipt of the successful response from the Bank. |

|  |  |
|---|---|
| Process Output | - Status – Success or Fail<br>- Fund Block id |
| Participating Roles | - Buyer<br>- GeM Portal<br>- Bank |
| Other Notes | 2) Fund Blocking is mandatory before Order can be placed.<br>3) This will be a synchronous call; So GeM Portal will not allow the user to continue with next activity till GeM gets the response back from Bank.<br>4) Once GeM Portal receives the response from bank Service, immediately same will be indicated on the screen.<br>5) There can be more than one blocking request for a Unique reference number. For each block request, a BlockReqId would be generated<br>6) There can be more than one Unblock Request (Partial Unblocking) for one Unique Reference Number or for a BlockReq id. |

### 3.5.2. Input parameters

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| Body |  |  |  |  |  |
| gemReqID | 80 | String | TRUE | GeM Request ID | body |
| gemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | Body |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| blockReqId | 50 | Numeric | Conditional | Unique id generated by GeM for a Challan/Fund Blocking. Bank to maintain duplicate blocking request check on this parameter.<br>For fund unblocking request, this parameter would be not be send | Body |
| ddoRegistrationNo | 20 | String | TRUE |  | body |
| fundBlockTransID | 40 | String | Optional | Fund Blocked Transaction ID generated by Bank. Will be send only in case of Unblocking and if received from bank in Fund blocking response | body |
| orgCode | 20 | String | TRUE | Org code shared by GeM | body |
| supplyOrderNo | 80 | String | OPTIONAL |  | body |
| buyerID | 120 | String | TRUE | Session ID of User | body |
| budgetHead |  | JSON | OPTIONAL | This information will be sent as a part of JSON and bank need to store this information as it is. Please refer parent element for list of fields under this JSON | body |

| | | | | | |
|---|---|---|---|---|---|
| functionHead | 10 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| objectCode | 10 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| grantNo | 10 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| Category | 50 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| Amount | 15.2 | Numeric | TRUE | | Body |
| Type | 1 | String | TRUE | B – Block, U – Unblock | Body |

### 3.5.3. Output fields

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| Body | N/A | N/A | TRUE | | N/A |
| gemReqID | 80 | String | TRUE | GeM Request ID | Body |
| GemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | Body |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| blockReqId | 50 | Numeric | TRUE | Unique id generated by GeM for a Challan/Fund Blocking. Bank to maintain duplicate blocking request check on this parameter. | Body |
| FundBlockTransID | 40 | String | Optional | Fund Blocked Transaction ID generated by Bank. For multiple blocking requests for a URN using different Block request id, the Bankk would return the same FundBlockTransactionID | body |
| Amount | 15 | Numeric | TRUE |  | body |
| Type | 1 | String | TRUE | B – Block, U – Unblock | body |
| Status | 1 | String | TRUE | S – Success, F – Failed | Body |
| Remarks | 200 | String | TRUE | Banks to share the reason of failure | Body |

**Unblocking Api**

**Input Parameters**

| Field Name | Length | Type | Mandatory | Values | Parent Element |
|---|---|---|---|---|---|
| Body |  |  |  |  |  |
| gemReqID | 80 | String | TRUE | GeM Request ID | body |

| | | | | | |
|---|---|---|---|---|---|
| gemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | Body |
| ddoRegistrationNo | 20 | String | TRUE | | body |
| fundBlockTransID | 40 | String | Optional | Fund Blocked Transaction ID generated by Bank. Will be send only in case of Unblocking and if received from bank in Fund blocking response | body |
| orgCode | 20 | String | TRUE | Org code shared by GeM | body |
| supplyOrderNo | 80 | String | OPTIONAL | | body |
| buyerID | 120 | String | TRUE | Session ID of User | body |
| budgetHead | | JSON | OPTIONAL | This information will be sent as a part of JSON and bank need to store this information as it is. Please refer parent element for list of fields under this JSON | body |

| | | | | | |
|---|---|---|---|---|---|
| functionHead | 10 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| objectCode | 10 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| grantNo | 10 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| Category | 50 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| Amount | 15.2 | Numeric | TRUE | | Body |
| Type | 1 | String | TRUE | B – Block, U – Unblock | Body |
| unblockReqId | 50 | Numeric | TRUE | Unique id generated by GeM for a Fund UnBlocking. Bank to maintain duplicate unblocking request check on this parameter. | body |

**Output Parameters:**

| Field Name | Length | Type | Mandatory | Values | Parent Element |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| Body | N/A | N/A | TRUE | | N/A |
| Amount | 15.2 | Numeric | TRUE | Amount for which chalan need to be generated | body |
| GemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | Body |
| gemReqID | 80 | String | TRUE | GeM Request ID | Body |
| Type | 1 | String | TRUE | B – Block, U – Unblock | Body |
| remarks | 200 | String | TRUE | | Body |
| Status | 1 | String | TRUE | S – Success, F – Failed | Body |
| unblockReqId | 50 | Numeric | TRUE | Unique id generated by GeM for a Fund UnBlocking. Bank to maintain duplicate unblocking request check on this parameter. | body |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| fundBlockTransID | 40 | String | Optional | Fund Blocked Transaction ID generated by Bank. Will be send only in case of Unblocking and if received from bank in Fund blocking response | body |
| errorCode |  |  |  |  |  |
| blockReqId | 50 | Numeric |  |  | body |

## 3.6. Payment Instruction

### 3.6.1. Service Details

|  |  |
|---|---|
| Service Name | Payment Instruction Request |
| Hosted By | Bank |
| Consumed By | GeM |
| Purpose of the Service | Purpose of this interface is to enable Payment processing from the Blocked Funds. The GeM Portal would use this interface to send the payment instruction to Bank to process the payment to the Beneficiary account. |
| Usage in GPA | 1) Challan Model<br>2) Non Challan Model |
| Method Of Integration | RESTFul services would be integrated as Json Structure |

| | |
|---|---|
| Service Availability Window Processing | The service should be available throughout year 24*7 |
| Process Summary | Payment processing by Buyer<br>- Buyer would generate a Bill on GeM Portal and push the Payment instruction.<br>- GeM Portal would call the Payment instruction service of the Bank and post the Payment instruction in the Request.<br>- Bank would register the Payment request and send the status to Gem in the Response.<br>Payment processing by System on violation of Payment SLA<br>- On CRAC+11 Day, if the Payment is not processed by Buyer, the system would auto trigger the payment for 80% of the amount.<br>- On CRAC+45 Days, if the Payment is not processed by Buyer, the system would auto trigger the payment for the remaining 20% of the amount.<br>- GeM Portal would call the Payment instruction service of the Bank and post the Payment instruction in the Request.<br>- Bank would register the Payment request and send the status to Gem in the Response |
| Process Output | - Status – Success or Fail<br>- Bank Transaction id |
| Participating Roles | - Buyer<br>- GeM Portal<br>- Bank |

| | |
|---|---|
| Other Notes | 1) Once GeM Portal receives the response from bank Service, immediately same will be indicated on the screen.<br>2) The Payment request also contains a unique identifier – paymentId to identify the Payment. The Banks should use this unique identifier to check the duplicate payments. |

### 3.6.2. Input parameters

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| Body | | | | | |
| gemReqID | 80 | String | TRUE | GeM Request ID | Body |
| gemUniqueRefe renceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | Body |
| ddoRegistration No | 20 | String | TRUE | | Body |
| fundBlockTrans ID | 40 | String | Optional | Fund Blocked Transaction ID generated by Bank | Body |
| orgCode | 20 | String | TRUE | Org code shared by GeM | Body |
| supplyOrderNo | 80 | String | OPTIONAL | | Body |
| invoiceNo | 20 | String | TRUE | Generated by GeM | Body |
| invoiceDate | 10 | Date | TRUE | DD-MM-YYYY | Body |
| buyerID | 120 | String | TRUE | Session ID of the user | Body |
| ddoCode | 10 | String | TRUE | | Body |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| budgetHead |  | JSON | OPTIONAL | This information will be sent as a part of JSON and bank need to store this information as it is. Please refer parent element for list of fields under this JSON | Body |
| functionHead | 10 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| objectCode | 10 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| grantNo | 10 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| Category | 50 | String | OPTIONAL | This fields will be consumed based on the different DDOs | Budget Head |
| Amount | 15.2 | Numeric | TRUE |  | Body |
| Type | 1 | String | TRUE | B – Block, U – Unblock | Body |
| lastPayment | 1 | String | TRUE | Y- Yes, N- No | Body |
| beneficiaryIFSCode | 11 | String | TRUE | IFSC Code of beneficiary | Body |
| beneficiaryAccountNo | 20 | String | TRUE | Account no of beneficiary | Body |
| beneficiaryAccountHolderName | 120 | String | TRUE | Name of beneficiary | Body |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| paymentId | 50 | Numeric | TRUE | Unique Payment ID generated by GeM for each payment instruction.Bank should place the duplicate payment check on this parameter | Body |

### 3.6.3. Output fields

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| body | N/A | N/A | TRUE |  | N/A |
| gemReqID | 80 | String | TRUE | GeM Request ID | Body |
| gemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | Body |
| transID | 40 | String | TRUE | Transaction ID generated by Bank | Body |
| amountBlocked | 15.2 | Numeric | TRUE |  | body |
| invoiceNo | 20 | String | TRUE | Generated by GeM | Body |
| invoiceDate | 10 | Date | TRUE | DD-MM-YYYY | Body |
| status | 1 | String | TRUE | S – Success, F – Failed | Body |
| remarks | 200 | String | TRUE | Banks to share the UTR Number | Body |
| paymentId | 50 | Numeric | TRUE | Unique Payment ID generated by GeM for each payment instruction. | Body |

## 3.7. Payment Status

### 3.7.1. Service Details

| | |
|---|---|
| Service Name | Payment Status Request |
| Hosted By | Bank |
| Consumed By | GeM |
| Purpose of the Service | Purpose of this interface is to get the status of the Payment request to check if the Payment is credited to the beneficiary. |
| Usage in GPA | 1) Challan Model<br>2) Non Challan Model |
| Method Of Integration | RESTFul services would be integrated as Json Structure |
| Service Availability Window Processing | The service should be available throughout year 24*7 |
| Process Summary | GeM would call the service of the bank to check the payment status using the Bank Transaction id or Payment id.<br>The Bank would process the request and share the Transaction status<br>GeM would save the transaction details received from bank. |
| Process Output | Status – Success or Fail<br>Transaction Reference |
| Participating Roles | Buyer<br>GeM Portal<br>Bank |

|  |  |
|---|---|
| Other Notes | 1) GeM would call the service using Bank transactionid or Paymentid in case banktransaction id is not available. |

### 3.7.2. Input parameters

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| Body | N/A | N/A | TRUE |  | N/A |
| gemReqID | 80 | String | TRUE | GeM Requested ID | Body |
| gemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | Body |
| bankTransID | 40 | String | CONDITIONAL | Transaction ID generated by Bank. If it is available return status based on bankTransID | body |
| transactionDate | 10 | Date | TRUE | DD-MM-YYYY | Body |
| paymentId | 50 | Numeric | CONDITIONAL | Unique Payment ID generated by GeM for each payment instruction.If it is availabale return status based on paymentId | Body |

### 3.7.3. Output fields

| Field Name | Length | Type | Mandatory | Values | Parent Element |
|---|---|---|---|---|---|
| Body | N/A | N/A | TRUE | | N/A |
| gemReqID | 80 | String | TRUE | Transaction ID generated by GeM | Body |
| gemUniqueReferenceNumber | 20 | Numeric | TRUE | Unique id generated by GeM | Body |
| bankTransID | 40 | String | TRUE | Transaction ID generated by Bank | Body |
| transactionDate | 10 | Date | TRUE | DD-MM-YYYY | Body |
| drCrDate | 10 | Date | TRUE | DD-MM-YYYY | body |
| paymentTransID | 40 | String | TRUE | | Body |
| amountofTransaction | 15.2 | Numeric | TRUE | | Body |
| transactionStatus | 1 | String | TRUE | | Body |
| transactionRemarks | 200 | String | TRUE | | Body |
| Status | 1 | String | TRUE | S – Success, F – Fail | Body |
| Remarks | 200 | String | TRUE | Banks to share the reason of failure | Body |
| paymentId | 50 | Numeric | TRUE | Unique Payment ID generated by GeM for each payment instruction. | Body |

# 4. Security

## 4.1. Authentication and Authorization

Clientid and secret key will be passed as header information. Bank will validate that information & then the request will be accepted. Each bank will have different client id and secret key.

## 4.2. AES Encryption and Decryption

For Handling request and response GeM is using AES -128 Encryption/Decryption Algorithm.
Bank have to generate the Bank Encryption/Decryption key and share with GeM. Bank will also use it to encrypt the response and decrypt the request.

AES encryption key will be different for each Banks. Key length should be 24 bit & cbc mode .

### 4.2.1. ENCYRYPTION:

GeM is using mcrypt library and AES-128 algorithm. Mcrypt mode is cipher block chaining i.e. CBC.
The second step is to generate initialization vector (IV). In our case, CBC mode requires IV.
Then using mcrypt encrypt function encrypt data and create the cipher text using
BANK_ENCRYPTION_DECRYPTION_KEY key.
Then base64 encode the data to be sent in response.

### 4.2.2. DECRYPTION :

Using Base 64 decode the request.
Create generate initialization vector (IV) and generated decrypted cipher.

```
$ciphertext_dec = base64_decode($value);
$iv_size = mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128, MCRYPT_MODE_CBC);
$iv_dec = substr($ciphertext_dec, 0, $iv_size);
$ciphertext_decrypted = substr($ciphertext_dec, $iv_size);
```

Using mcrypt_decrypt function decrypt the encoded cipher using
BANK_ENCRYPTION_DECRYPTION_KEY key into plain text.

```
$plaintext_decrypted = mcrypt_decrypt(MCRYPT_RIJNDAEL_128,
BANK_UNIQUE_KEY,$ciphertext_decrypted, MCRYPT_MODE_CBC, $iv_dec);
```

**JAVA Sample Code for Encryption and decryption :**

**Decryption Code :-**

```
                    public String decryptEncData (String encData1, String ivstring, String encryptionKey)
{

                              String decryptedText = "";
                              String finalText="";

                              try {
                                                  byte[] secretKeyInByte = encryptionKey.getBytes();

                                                  SecretKeySpec secretkeyspec = new
SecretKeySpec(secretKeyInByte, "AES");
                                                  IvParameterSpec ivparameterspec = new
IvParameterSpec(ivstring.getBytes());
                                                  Cipher cipher = Cipher.getInstance("AES/CBC/NoPadding");
                                                  cipher.init(Cipher.DECRYPT_MODE, secretkeyspec,
ivparameterspec);
                                                  byte[] encByteArray = (new
org.apache.commons.codec.binary.Base64()).decode(encData1.getBytes());
                                                  byte[] cipherText = cipher.doFinal(encByteArray);
                                                  decryptedText = new String(cipherText, "UTF-8");
                                                  int startIndex=decryptedText.indexOf("{");
                                                  int lastIndex=decryptedText.lastIndexOf("}");

          //logger.info("startIndex.........."+startIndex+"...lastIndex..."+lastIndex);

                                                  finalText=decryptedText.substring(startIndex, lastIndex+1);

                              } catch (Exception e) {
```

```
                                        e.printStackTrace();
                                        //return "Error";
                        }

                        return finalText;

                }
}
```

**Encryption Code :-**

```
                public String encrypt(String plainText, String ivstring, String encryptionKey) throws
Exception {
                                String encryptedText = "";
                                String characterEncoding= "UTF-8";
                                String aesEncryptionAlgorithem = "AES";
                                try
                                {
                                        while(plainText.getBytes().length % 16!=0)
                                        {
                                                plainText+='\u0020';
                                        }

                                        Cipher cipher = Cipher.getInstance("AES/CBC/NoPadding");
                                        byte[] key = encryptionKey.getBytes(characterEncoding);
                                        SecretKeySpec secretKey = new SecretKeySpec(key,
aesEncryptionAlgorithem);
                                        IvParameterSpec ivparameterspec = new
IvParameterSpec(ivstring.getBytes());
                                        cipher.init(Cipher.ENCRYPT_MODE, secretKey,
ivparameterspec);
                                        //byte[] decByteArray = (new
org.apache.commons.codec.binary.Base64()).encode(plainText.getBytes("UTF-8"));
```

```
                                        //byte[] cipherText = cipher.doFinal(decByteArray);
                                        byte[] cipherText = cipher.doFinal(plainText.getBytes("UTF8"));
                                        //Base64.Encoder encoder = (new
org.apache.commons.codec.binary.Base64()).encodeAsString(pArray)
                                        encryptedText = (new
org.apache.commons.codec.binary.Base64()).encodeAsString(cipherText);


                                } catch (Exception E) {

                                        System.err.println("Encrypt Exception : " + E);
                                }
                                return encryptedText;
                        }


}
```

# 5. Error Codes

## 5.1. Description

Error codes are implemented to handle specific error occurring at the bank side when GeM make request to different APIs endpoint. Below is api wise specific error codes.

## 5.2. Timeout for third party system

If we don't get the HTTP status of request as 200 we will treat the request as timeout from third party/bank for all the apis.

## 5.3. Error code list

### 5.3.1. Pool Account Validation:

| Cases | Validation Message | Error Code |
|---|---|---|
| **If org code does not match with Pool Account No. & IFSC** | Invalid Orgcode found | 902 |
| **If Org code is already registered with different account** | Duplicate Orgcode | 903 |
| **If Account no. and IFSC does not match** | Invalid Pool Account No. OR IFSC Code found | 904 |

| | | |
|---|---|---|
| **If IP address is wrong** | Invalid IP address | 921 |

### 5.3.2. Van DDO Mapping

| **Cases** | **Validation Message** | **Error Code** |
|---|---|---|
| **If IP address is wrong** | Invalid IP address | **921** |
| **If DDO code is already registered** | DDO is already registered | 908 |

### 5.3.3. Challan generation

| **Cases** | **Validation Message** | **Error Code** |
|---|---|---|

| | | |
|---|---|---|
| **If DDO Registration No. is not found for a given OrgCode** | Invalid DDO Registration No. | 909 |
| **If IP address is wrong** | Invalid IP address | 921 |

### 5.3.4. Balance Enquiry

| **Cases** | **Validation Message** | **Error Code** |
|---|---|---|
| **If invalid Pool Account No. found for a given valid Organization Code** | Invalid Pool Account No. | 918 |
| **If DDO Registration No. & Buyerpool AC No. does not match** | Verify Parameters : ddoRegistrationNo or BuyerPoolAccount | 919 |

| | | |
|---|---|---|
| **If IP address is wrong** | Invalid IP address | 921 |
| **DDO Registration No. & Org Code does not match** | Invalid DDO Registration No. | 909 |

### 5.3.5. Block-Unblock

| **Cases** | **Validation Message** | **Error Code** |
|---|---|---|
| **If invalid DDO Registration No. is provided for a given Organization Code** | Invalid DDO Registration No. | 909 |
| **If IP address is wrong** | Invalid IP address | 921 |
| **Unblocking of amount before blocking** | Operation failed! Insufficient balance in blocked account | 922 |

| | | |
|---|---|---|
| **If user is blocking the amount more than the available balance** | Operation failed! Sufficient Balance is not available in DDO's Virtual Account | 923 |
| **If user is unblocking the amount after passing Y flag in payment instruction API** | Last Payment flag is already received. You can't unblock fund now. | 924 |
| **After unblocking the amount if user block the amount with same GEM URN(Previous one)** | No further Operation is allowed after Unblocking Fund Request | 926 |

### 5.3.6. Payment Instructions

| **Cases** | **Validation Message** | **Error Code** |
|---|---|---|

| | | |
|---|---|---|
| **If all fields are blank** | orgCode:Value not found;buyerPoolAcctNo:Value not found;ifsCode:Value not found | 905 |
| **Payment failed** | Payment failed   buyer account is dormant or unaccessible | 913 |
| **If IP address is wrong** | Invalid IP address | 921 |
| **If user has  done payment by passing Y flag** | Further Payment is not allowed | 927 |
| **Without blocking the amount if user is trying to do the payment** | Operation failed! Sufficient Balance is not available in DDO's Virtual Account | 928 |
| **when payment amount is more than the blocked amount** | Operation failed! Requested payment amount is greater than the Blocked Amount" | 929 |

|  |  |  |
|---|---|---|
| **If invalid DDO Registration No. is provided for a given Organization Code** | Invalid DDO Registration No. | 909 |

### 5.3.7. Payment Status

| **Cases** | **Validation Message** | **Error Code** |
|---|---|---|
| **If any of the field is blank** | Value not found | 905 |
| **If bank transaction id is wrong** | Verify Parameters : bankTransId | 915 |
| **Payment failed** | Payment failed  seller  account is dormant or unaccessible or not credited | 913 |
| **If IP address is wrong** | Invalid IP address | 921 |

## 5.4. Response Structure for Error Codes

Bank need to share the for API response as same as above response parameters for each API as well with below error parameters in case of failure. In case of multiple error codes , All error codes need to append with **errors** key.

Error Response = APIs Failure Response with Response architecture of API + Error Response

| **Field Name** | **Length** | **Type** | **Mandatory** | **Values** | **Parent Element** |
|---|---|---|---|---|---|
| Body | | | | | |
| errors | | JSON | TRUE | | body |
| errorCode | 3 | String | | Error code as per document shared by Gem | error |
| Message | 200 | String | | Error Message as shared by Gem | error |

## 5.5. Error Code Sample Response

```
{
        "body" :
                {
                        "errors": [
                                {
                        "errorCode": "902",
                         "message": "Invalid Orgcode found."
```

```
                                        },
                        {
                         "errorCode": "904",
                         "message": "Invalid Pool Account No. OR IFSC Code found."
                                }
                            ]
              }
    }
```

**Error Response Example for Pool Account validation API :**

**In case of 902 & 903 Error codes :**

```
 {
   "accountStatus": "I",
   "ifsCode": "BANKIFSC123456",
   "bankTransID": "f1b3e26f03cb4a8b9f2287d7c159c8a5",
   "buyerPoolAcctNo": "0000000000",
   "orgCode": "ORG-000",
   "gemReqID": "Gem-01234567890",
   "accountHolderName": "Account Holder name",
   "remarks": "Test reson failure from banks",
   "accountHolderEmail": "Account Holder email id",
   "accountHolderEmailStatus": "I",
   "modeOfOperation": "C",
   "status": "F",
   "errors": [
      {
         "errorCode": "902",
         "message": "Invalid Orgcode found."
      },
      {
            "errorCode": "904",
            "message": "Invalid Pool Account No. OR IFSC Code found."
        }
    ]
}
```

# 6. Sample JSON Responses for the APIs

## 6.1. Pool Account Validation API :

**Endpoint** : [BANK INTERFACE URL]/AccountValidation

**Sample Request** :

```
{
  "body" :
           {
            "gemReqID":"Gem-15427850392948451",
             "orgCode":"ORG-123",
             "buyerPoolAcctNo":"0000000000000",
             "ifsCode":"BANKIFSC123456",
             "accountHolderName":"Account holder name without special characters",
            “accountHolderEmail”:”Email id of the account Holder”,
          }
  }
```

**Sample Success Response:**

```
{
           "body" :
            {
                "accountStatus": "V",
                "ifsCode": "BANKIFSC123456",
                "bankTransID": "aada94502bb14d07a0c30909a737f22c",
                "buyerPoolAcctNo": "19876543210",
                "orgCode": "ORG-000",
                "gemReqID": "Gem-01234567890"
                "accountHolderName": "Account Holder name",
                “accountHolderEmail”:”Account Holder email id”,
               “accountHolderEmailStatus”:”S”,
              “mode of Operation”:”C”,
                    "remarks": "Account Validated successfully",
                        "status": "S"
                }
        }
```

**Sample Error Response :**

```
  {
      "body" :
          {
             "accountStatus": "I",
             "ifsCode": "BANKIFSC123456",
             "bankTransID": "f1b3e26f03cb4a8b9f2287d7c159c8a5",
             "buyerPoolAcctNo": "0000000000",
             "orgCode": "ORG-000",
             "gemReqID": "Gem-01234567890",
             "accountHolderName": "Account Holder name",
             "remarks": "Duplicate gem Id",
                   "accountHolderEmail":"Account Holder email id",
                       "accountHolderEmailStatus":"I",
                      "mode of Operation":"C",
             "status": "F"
          }
  }
```

## 6.2. Van DDO Mapping:

**Endpoint** : [BANK INTERFACE URL]/VanDdoRegistration

**Sample Request :**

```
"{
  "body":
        {
        "gemReqID":"Gem-1234512390103",
        "orgCode":"ORG-000",
                "buyerPoolAcctNo":"010200100203456",
                "ifsCode":"BANKIFSC0012313",
                "accountHolderName":"ACCOUNT HOLDER NAME WITHOUT SPECIAL
CHARACTERS",
                "buyerID":"12343"
                "ddoCode":"DDO-12-34"
                }
         }"
```

**Sample  Success Response :**

```
 {
        "body":
                {
                 "ifsCode": "BANKIFSC0012313",
                  "ddoRegistrationNo": "GEM123400001",
                  "bankTransID": "ef0e7a8235346dsf223ss8552b66f",
                  "buyerPoolAcctNo": "012345670987",
                  "orgCode": "ORG-001",
                  "gemReqID": "Gem-987654321010",
                 "buyerID": "012",
                  "ddoCode": "DDO-01-123",
                  "accountHolderName": "Account Holder Name",
                 "remarks": "DDO code is successfully mapped with the given ORG Code",
                 "status": "S"
          }
}
```

**Response When DDO code is already       mapped with org code.**

```
  {
        "body":
                {
                  "ifsCode": "BANKIFSC0012313",
                    "ddoRegistrationNo": "GEM123400001",
                     "bankTransID": "ef0e7a8235346dsf223ss8552b66f",
                    "buyerPoolAcctNo": "012345670987",
                     "orgCode": "ORG-001",
                     "gemReqID": "Gem-987654321010",
                     "buyerID": "012",
                     "ddoCode": "DDO-01-123",
                     "accountHolderName": "Account Holder Name",
                     "remarks": "DDO code is already mapped with the given ORG Code",
                     "status": "D"
            }
}
```

**Sample Error Response :**

```
{
            "body":
                  {
                     "ifsCode": "BANKIFSC0012313",
                     "ddoRegistrationNo": "",
                     "bankTransID": "ef0e7a8235346dsf223ss8552b66f",
                     "buyerPoolAcctNo": "012345670987",
                       "orgCode": "ORG-001",
                      "gemReqID": "Gem-987654321010",
                     "buyerID": "012",
                      "ddoCode": "DDO-01-123",
                    "accountHolderName": "Account Holder Name",
                    "remarks": "Unable to map the DDO code with Org Code",
                    "status": "F"
            }
}
```

## 6.3. Challan generation :

**Endpoint** : [BANK INTERFACE URL]/ChallanToken

**Sample request :**

```
{
            "body":
                        {
                          "gemReqID":"Gem-987654321010",
                         "gemUniqueReferenceNumber":"026725345",
                       "blockReqId":"2323",
                          "amount":"5000",
                          "date":"01-07-2019 11:58:00",
                         "orgCode":"ORG-123",
                        "ddoRegistrationNo":"GEM1234567",
                        "ddoCode":"DDO-12-123",
                       "ddoName":"test",
           "topup":"N"
             }
}
```

**Sample Success response :**

```
{
   "body" :
     {
      "gemReqID": "Gem-987654321010",
      "gemUniqueReferenceNumber": "026725345",
      "blockReqId":"2323",
      "amount": 5000,
      "date": "2019-07-09T11:02:45.3332728+05:30",
      "orgCode": "ORG-123",
      "ddoRegistrationNo": "GEM98765243",
      "ddoName": "test",
        "topup": "N",
        "status": "S",
        "remarks": "Challan Generated Successfully",
        "token": "NJKHASDKJHKJHADNnsdjdIGISDjkjNjk4MDAwMDgwMjY3MjUzNDU=",
        "ddoCode": "DDO-12-123"
      }
   }
```

**Sample Error response :**

```
{
        "body":
                {
                  "gemReqID": "Gem-987654321010",
                  "gemUniqueReferenceNumber": "026725345",
                "blockReqId":"2323",
                  "amount": 5000,
                  "date": "2019-07-09T14:45:00.4402861+05:30",
                 "orgCode": "ORG-123",
                 "ddoRegistrationNo": "GEM1234567",
                 "ddoName": "test",
                 "topup": "N",
                "status": "F",
                 "remarks": "Duplicate Gem Request Id",
                 "token": "",
                 "ddoCode": "DDO-12-123",
                "url_chalan": ""
                }
        }

        {
          "body":
                 {
                    "gemReqID": "Gem-987654321010",
                    "gemUniqueReferenceNumber": "026725345",
                    “bockReqId”: “ 45678987”
                     "amount": 5000,
                    "date": "2019-07-09T14:45:00.4402861+05:30",
                    "orgCode": "ORG-123",
                    "ddoRegistrationNo": "GEM1234567",
                      "ddoName": "test",
                    "topup": "N",
                    "status": "F",
                    "remarks": "Unable to generate challan.",
                    "token": "",
                     "ddoCode": "DDO-12-123",
                    "url_chalan": ""
                    }
}
```

## 6.4. Balance enquiry API :

**Endpoint** : [BANK INTERFACE URL]/BalanceEnquiry

**Sample Request :**

```
    {
      "body":
           {
             "gemReqID":"GEM-987654321010",
             "gemUniqueReferenceNumber":"026725345",
          "blockReqId":"2323",
              "orgCode":"ORG-123",
             "buyerPoolAcctNo":"0300023123456789",
            "ifsCode":"BANKIFSC01234567",
            "accountHolderName":"Account holder name",
           "ddoRegistrationNo":"GEM1234567"
```

```
                    }
            }
```

**Sample  success response :**

```
        {
        "body":
                {
                   "gemReqID": "GEM-987654321010",
                  "gemUniqueReferenceNumber": "026725345",
                "blockReqId":"2323",
                  "bankTransID": "34345164a4sd234asd23e097414b",
                  "orgCode": "ORG-123",
                   "buyerTreasuryCode": "",
                   "buyerPoolAcctNo": "01999992312121",
                   "ddoRegistrationNo": "GEM1234567",
                   "ifsCode": "BANKIFSC01234567",
                   "accountHolderName": "Account holder name",
                   "availableBalance": "10000.00",
                   "status": "S",
                   "remarks": "Request Processed Successfully"
                }
        }
```

**Sample Error response :**

```
        {
        "body":
                {
                   "gemReqID": "GEM-987654321010",
                   "gemUniqueReferenceNumber": "026725345",
                "blockReqId":"2323",              "bankTransID": "7fb852d61a7dfgfdw323fd24ts7c2b3",
                   "orgCode": "",
                   "buyerTreasuryCode": "",
                   "buyerPoolAcctNo": "",
                        "ddoRegistrationNo": "",
                        "ifsCode": "",
                        "accountHolderName": "",
                        "availableBalance": "0.00",
                        "status": "F",
                        "remarks": "One or more errors occurred."
                    }
            }
  {
            "body" :
                    {
                        "gemReqID": "GEM-987654321010",
                        "gemUniqueReferenceNumber": "026725345",
                        "bankTransID": "dbsdfhk9284y29jaske7ebe3716b",
                        "orgCode": "ORG-123",
                        "buyerTreasuryCode": "",
                        "buyerPoolAcctNo": "0100002312891",
                        "ddoRegistrationNo": "GEM12345678",
                        "ifsCode": "BANKIFSC01234567",
                        "accountHolderName": "Account Holder Name",
                        "availableBalance": "0.00",
                        "blockReqId":"2323",
                        "status": "F",
                        "remarks": "Duplicate Gem Request Id"
                    }
}
```

## 6.5. Blocking Unblocking :

**Endpoint** : [BANK INTERFACE URL]/BlockUnblock

**Sample Request :**

```
{
"body":
```

```
                    {
                    "gemReqID":"Gem-1542785990900999",
                    "gemUniqueReferenceNumber":"026725345",
                   "blockReqId":"2323",
                    "ddoRegistrationNo":"GEM123800008",
                    "fundBlockTransID":"152472592",
                    "orgCode":"ORG-123",
                    "supplyOrderNo":"333333",
                    "buyerID":"24234",
                    "budgetHead":
                            {
                                    "functionHead":"test",
                                    "objectCode":"test",
                                    "grantNo":"test",
                                    "category":"test"
                            },
                   "amount":"1000",
                    "type":"B"
          }
}
```

**Sample Success Response**

```
        {
                "body":
                    {
                    "gemReqID":"Gem-1542785990900999",
                    "gemUniqueReferenceNumber":"026725345",
                   "blockReqId":"2323",
                    "fundBlockTransID":"152472592",
                    "amount":"1000",
                    "type":"B"
                    "status":"S"
                    "remarks":"Banks to share failure/success remarks"
            }
    }
```

Sample Unblocking Request:

```
    {
        "body": {
            "gemReqID": "1634391629",
            "gemUniqueReferenceNumber": "511687714473293",
            "ddoRegistrationNo": "APAP00130026881",
            "fundBlockTransID": "",
            "orgCode": "ICIC0-231234",
            "supplyOrderNo": "511687714473293",
            "buyerID": "69846668139340",
            "budgetHead": {
                "functionHead": "",
                "objectCode": "",
                "grantNo": "",
                "category": ""
            },
            "amount": "3.0",
            "type": "U",
            "unblockReqId": "91634391629163439l629"
        }
    }
```

**Sample Unblocking response:**

```
    {
        "body": {
            "amount": "3.0",
            "errorCode": "",
            "gemReqID": "1634391629",
            "gemUniqueReferenceNumber": "511687714473293",
            "type": "U",
            "blockReqId": null,
            "remarks": "SUCCESS",
            "fundBlockTransID": "36556",
            "status": "S",
            "unblockReqId": "91634391629163439l629"
        }
    }
```

## 6.6. Payment Instruction :

**Endpoint** : [BANK INTERFACE URL]/PaymentInstructions

Sample Request :

```
{
        "body":
                {
        "gemReqID":"P2343541",
        "gemUniqueReferenceNumber":"022412",
        "ddoRegistrationNo":"92823882300",
        "fundBlockTransID":"1234",
        "supplyOrderNo":"012091391293",
        "invoiceNo":"333332",
        "invoiceDate":"14-06-2019",
        "buyerID":"123455",
        "ddoCode":"SAMPLEDDO_CODE",
         "budgetHead":
                        {
                                "functionHead":"test",
                                "objectCode":"test",
                                "grantNo":"test",
                                "category":"test"
                        },
        "amount":"5000",
        "type":"U",
        "lastPayment":"N",
        "beneficiaryIFSCode":"IFSCBANKN0000437",
        "beneficiaryAccountNo":"91239023901230",
        "beneficiaryAccountHolderName":"Account holder name",
        "orgCode":"Organization_Code",
        “paymentId”:”2423525”
}
```

}

**Sample success response :**

```
{
"body":
        {
   "gemReqID": "P2343541",
   "gemUniqueReferenceNumber": "022412",
   "transID": "34345164a4sd234asd23e097414b",
   "amountBlocked": "5000",
   "invoiceNo":"333332",
   "invoiceDate":"14-06-2019",
   "status": "S",
   "remarks": "BANKIFSC01234567",
   “paymentId”:”2423525”
   }
}
```

**Sample error response :**

```
{
        "body":
                {
                  "gemReqID": "P2343541",
                   "gemUniqueReferenceNumber": "022412",
                   "transID": "34345164a4sd234asd23e097414b",
                  "amountBlocked": "",
                  "invoiceNo":"333332",
                  "invoiceDate":"14-06-2019",
                   "status": "F",
                    “paymentId”:”2423525”
                   "remarks": "Failure to initiate payment instruction"
                    }
    }
```

## 6.7. Payment Status :

**Endpoint** : [BANK INTERFACE URL]/PaymentStatus

**Sample Request :**

```
    {
            "body":
                    {
                            "gemReqID":"P2343541",
                            "gemUniqueReferenceNumber":"022412",
                            "bankTransID":"92823882300",
                            "transactionDate":"17-07-2019",
                             “paymentId”:”2423525”
                    }
            }
```

**Sample success response :**

```
    {
    "body":
            {
              "gemReqID": "P2343541",
              "gemUniqueReferenceNumber": "022412",
              "bankTransID": "34345164a4sd234asd23e097414b",
              "transactionDate": "09-07-2019",
              "drCrDate":"17-07-2019",
              "paymentTransID":"14-06-2019",
                    "amountofTransaction": "5000.00",
                    "transactionStatus": "S",
                    "transactionRemarks": "Transaction successfull",
                     "status": "S",
                    “paymentId”:”2423525”,
                    "remarks": "Banks to share the reason of success"
                }
        }
```

**Sample  error response :**

```
        {
        "body":
                {
           "gemReqID": "P2343541",
           "gemUniqueReferenceNumber": "022412",
           "bankTransID": "34345164a4sd234asd23e097414b",
           "transactionDate": "09-07-2019",
           "drCrDate":"17-07-2019",
            "paymentTransID":"14-06-2019",
           "amountofTransaction": "5000.00",
           "transactionStatus": "F",
           "transactionRemarks": "Transaction failure",
           "status": "F",
          “paymentId”:”2423525”,
           "remarks": "Banks to share the reason of failure"
           }
}
```

## 7. Requirements from the Bank

For communicating with GeM Portal , Following details are to be shared by Bank to GEM to utilize GeM GPA services :

-  Bank Encryption Key : GeM is using AES 128 encryption for request and response encryption and decryption. Bank encryption key is used to encrypt/decrypt the request and response. Only Bank Encryption/Decryption keys of sizes 24 bit provided by bank as per GPA integration document. It should not contain special characters.

Example : 86A49F0JSDK459D3BCB9E1

-  Client Id Key & Client Id Value : For security purposes the bank needs to provide the Client id. It is sent with header for validating and accepting the request. It should not contain special characters.

Example : **KEY**: ClientID **Value** : b23jje-0f64-41234-9a86-e8fawf895 or BANKGEMINB

-  Client token Key & Client token Value : For security purposes the bank needs to provide the Client token. It is sent with header for validating and accepting the request. It should not contain special characters.

Example : **KEY**: SecretKey **Value** : 2345235023895 or Mesdfl2q1239a4

-  Challan url : This is bank challan generation url, bank needs to provide this for generating challan.

Example : https://gem.testbank.co.in/Challan_Generation/ChallanRequest?token=

-  Bank IP to be whitelisted : Bank need to provide the IP address for whitelisting to GeM to allow request and response. Gem handles traffic at Port 443 so bank need to utilize network traffic on the same port.

-  Bank Interface URL : This is the bank url at which the GeM send the request using different endpoints as per the API document.

Example : *https://test.bank.in/BankApi/api/GEMWebService/*

-  Bank Endpoints URL Sample :

https://test.bank.in/BankApi/api/GEMWebService/AccountValidation

https://test.bank.in/BankApi/api/GEMWebService/VanDdoRegistration

https://test.bank.in/BankApi/api/GEMWebService/BlockUnblock

https://test.bank.in/BankApi/api/GEMWebService/PaymentInstructions

https://test.bank.in/BankApi/api/GEMWebService/PaymentStatus

https://test.bank.in/BankApi/api/GEMWebService/BalanceEnquiry

https://test.bank.in/BankApi/api/GEMWebService/ChallanToken

## 8. Acceptance of Integration Design

We have understood the design details presented for API integration and accept the design and process flow.

____________________________

Authorized Signatory / IT Nodal officer

<<Bank Name >>

**********************End of document******************************