Government e Marketplace (GeM)

GeM SOP for Integrating Payment System of CPSUs and other entities using ERP with GeM Platform

Version 4.0

Table of Contents

1. Metadata of the Standard

S. No.	Data elements	Description
1.	**Title**	GeM SOP - Integrating Payment System of CPSUs and other entities using ERP with GeM Platform, Version 1.0
2.	**Drafted by**	Dr. Rajesh Narang, CTO, GeM
3.	**Approval**	Competent authority of GeM
4.	**Publisher**	GeM
5.	**Brief Description**	This document covers the SOP for Integrating Payment System of CPSUs and other government entities using ERP, with GeM Platform for seamless transaction
6.	**Target Audience**	• Designated officer of CPSU and other government entities using ERP • GeM officials
7.	**Document Number**	GeM/SOP/ERP_Govt_entity_PSU/V4.0
8.	**Revision History**	• V1.0 SOP for GeM-ERP Integration Initial Draft was prepared. • V3.0 SOP for GeM-ERP Integration was enhanced with Description of each field used in the Web Services. • V4.0 SOP for GeM-ERP Integration was enhanced adding the SOAP based web services and the architectural diagram of the API's. (Revised on - 3rd December 2018)

2. Abbreviation

S.No.	Abbreviation	Description
1.	CPSU	Central Public Sector Undertaking
2.	CRAC	Consignee receipt and acceptance certificate
3.	DP	Delivery Period
4.	ERP	Enterprise Resource Planning
5.	FMS	Financial Management System
6.	GeM	Government eMarketplace
7.	GFR	General Financial Rules
8.	JSON	JavaScript Object Notation
9.	PRC	Provisional receipt certificate
10.	PSU	Public Sector Undertaking
11.	RA	Reverse Auction
12.	SGPA	State GeM Pool Account
13.	SLA	Service level agreement
14.	SOP	Standard operating procedure
15.	T&C	Terms and condition
16.	UAT	User acceptance testing
17.	VAN	Virtual account number

3. Introduction

Most of the CPSUs and other government entities, have implemented ERP solutions for supporting the functioning of the organization, including procurement. These ERP solutions, depending upon the scale and functionality of implementation, cater to the requirement ranging from and including:

- Human Resource.
- Inventory.
- Sales & Marketing.
- Purchase.
- Finance & Accounting.
- Customer Relationship Management(CRM)
- Engineering/ Production.

The various ERP packages being used by the CPSU include COTS ERP solution like SAP – being used by ONGC, NTPC and Oracle – being used by BPCL or Bespoke developed ERP solutions.

At present, each CPSU procures the products and services using their own ERP. However, the ERP does not provide the dynamic and real time prices of the products which are available on GeM Portal. Moreover, as per GFR rule 149, PSUs are mandated to procure the products through GeM marketplace.

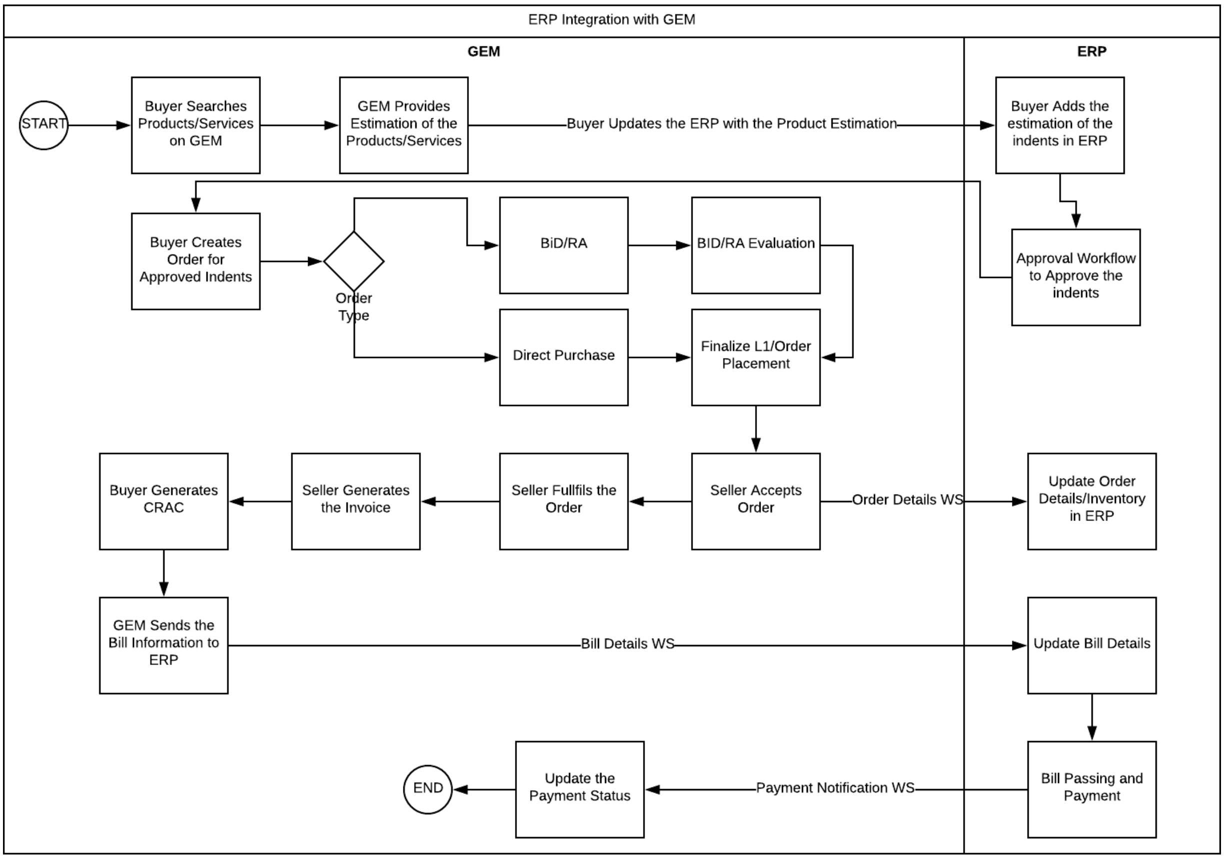
For Onboarding the CPSUs and such other government entities, GeM Portal has created a generic integration framework to integrate them. The buyers of such entities who use the ERP for procurement will require to register on GeM, with unique IDs. The PSU would be required to share the details of their existing and empaneled suppliers with GeM for onboarding. The entities are also requested to communicate the need to register on GeM, to their suppliers. Once registered on GeM portal the offered products and services of such suppliers will be available for procurement on GeM Portal.

A buyer will be able to procure product and services and specify the consignees where the product is to be delivered. The GeM marketplace will maintain the order details and generate the contract between the buyer and seller. The order summary and order details will be shared with the ERP via a web service (if integrated). The GeM marketplace will also maintain the bill summary and bill details, CRAC details and will share with the ERP via a Web Service.

4. Common flow for procurement through GeM and integration touch points between GeM & ERP system of CPSU

1. The buyer has access to both GeM Portal and ERP. The buyer logs into GeM portal, searches for the desired products and services, selects the most suitable among them, compares their prices for 3 OEMs and uses the prices for taking approval in his/her organization. The comparison sheet becomes a critical artifacts for future use, hence it is stored and saved on the GeM System.
2. On getting the approval, the buyer will place an order on GeM (direct purchase or BID/RA)
3. On acceptance of the order by seller, the GeM will send the order details to ERP via order summary and order details web services.
4. On fulfillment of the order by the seller, the buyer will generate the CRAC on GeM.
5. On generation of CRAC, the GeM portal will send the bill details to the ERP via web services bill summary and bill details..
6. The buyer will make the payment to the seller in ERP system and ERP system will update the GeM on payment status via web service.
7. The GeM will extend the functions of post contact management for products and services and will create the required services to share with the ERP.

Figure 1 ERP Integration with GeM


ERP Integration with GEM
GEM
ERP
START
Buyer Searches Products/Services on GEM
GEM Provides Estimation of the Products/Services
Buyer Updates the ERP with the Product Estimation
Buyer Adds the estimation of the indents in ERP
Approval Workflow to Approve the indents
Buyer Creates Order for Approved Indents
Order Type
BiD/RA
BID/RA Evaluation
Direct Purchase
Finalize L1/Order Placement
Seller Accepts Order
Order Details WS
Update Order Details/Inventory in ERP
Seller Fullfils the Order
Seller Generates the Invoice
Buyer Generates CRAC
GEM Sends the Bill Information to ERP
Bill Details WS
Update Bill Details
Bill Passing and Payment
Payment Notification WS
Update the Payment Status
END

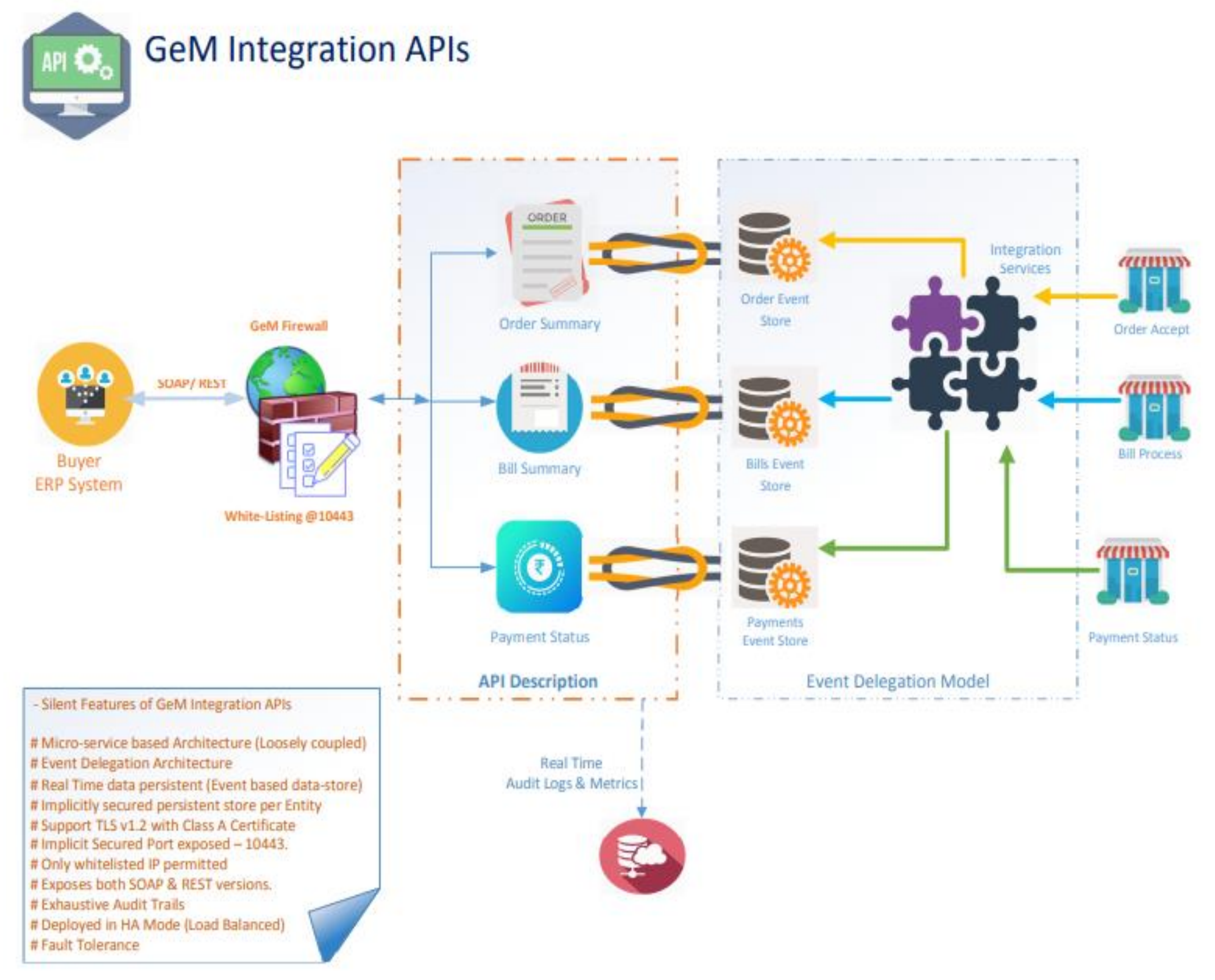
5. Architecture Snapshot

The architecture is micro services based which structures an application as collection of loosely coupled services, which implement business capabilities. The micro service architecture enables the continuous delivery/deployment of large, complex applications. To ensure security, only the secured port is used (10443).

The architecture is exposed to both SOAP as well as REST services for integration. TLS v1.2 with Class A certificate is supported. Additionally, only white listed IP’s are permitted.

The architecture is aligned with event delegation i.e. using event propagation (bubbling) to handle events at a higher level in the DOM than the element on which the event originated. It allows us to attach a single event listener for elements that exist now or in the future.

Figure 2 GeM Integration APIs


GeM Integration APIs
API
Buyer ERP System
SOAP/ REST
GeM Firewall
White-Listing @10443
ORDER
Order Summary
Bill Summary
Payment Status
API Description
Order Event Store
Bills Event Store
Payments Event Store
Integration Services
Order Accept
Bill Process
Payment Status
Event Delegation Model
Real Time Audit Logs & Metrics
- Silent Features of GeM Integration APIs
# Micro-service based Architecture (Loosely coupled)
# Event Delegation Architecture
# Real Time data persistent (Event based data-store)
# Implicitly secured persistent store per Entity
# Support TLS v1.2 with Class A Certificate
# Implicit Secured Port exposed – 10443.
# Only whitelisted IP permitted
# Exposes both SOAP & REST versions.
# Exhaustive Audit Trails
# Deployed in HA Mode (Load Balanced)
# Fault Tolerance

6. Web services for integration with ERP

All service are exposed and can be used as REST as well as SOAP. The data will be exchanged between GeM portal and the ERP system of the PSUs, using the following:

- Transport protocol – Https
- Data Format – JSON / XML
- Mode of communication - synchronous.

Below are the details of the web services which the ERP system of PSU need to consume after understanding of new json response structure.

6.1 Order Count

Entity will make a call to our web service (REST as well as SOAP services) which will internally make call to our database. REST/SOAP Services will immediately respond back to entity with relative response as per requested parameters.

Input Parameters

End Point	Request Method	Value(json)	Response format
To be shared separately	POST	{"clientCode":"CRPF","transactionId":"CRPF:12345:2172198713928","fromDate":"2017-05-01","toDate":"20117-05-11","authToken":"5AA8AFC3209A4381CE4C9B172992922E"}	Json(payload)

Request Parameters

Parameter Name	Type	Mandatory	**Description**
from_date	Date(YYYY-MM-DD)	TRUE (if as_on/to_date not set)	Start Date
to_date	Date(YYYY-MM-DD)	TRUE (if from_date is set)	End Date
transactionId	{clientcode}:{random 5 digit}:{date in yyyyMMddHHmmssSSS}	True	Transaction Id
authToken	Will be shared in email to client)	Yes	Auth Token
clientCode	String	Yes	Entity Name
encryptedResponseHash	String		
responseHash	String		

Parameters description

1. **from_date** : It will response all the record from requested date.
2. **to_date** : This parameter will not work without from_date.
3. **transactionId:** This will be a combination of client code, random 5 digit and date
4. **authToken:** This will be combination of multiple parameters and will be shared with client over email.
5. **clientCode: Entity Name**

Output fields

Field Name	Length	Type	Mandatory	Description
transactionId	{clientcode}:{random 5 digit}:{date in yyyyMMddHHmmssSSS}	String	Yes	This will be a combination of client code, random 5 digit and date
responseStatus	1 (“S/F”)	String	Yes	Response Status i.e. Success or Failure
Message	60	String	Yes	Success Message or Error Message in case of failure
responseData	Variable (Decryption key will be shared over email)	String	Yes	Example: fEkhNhUDEOaj1pBTabI5aAOgOpOaw4qLHbm4Ydu4P8LlLe14n9jaeFoQ1KMFSTiu
responseHash	32	String	Yes	2458404B0C6686031A96FE01563565F1
encryptedResponseHash	32	String	Yes	2458404B0C6686031A96FE01563565F1

ResponseData will have the following response fields:

Field Name	Length	Type	Mandatory	Description
Date	Date(YYYY-MM-DD)	String	Yes	Dates between the from date and to date having orderCount>0
Count	Length as of integer	Integer	Yes	Count of order on specific date

Entity will make a call to our web service exposed by RESTFUL layer which will internally make call to our database. RESTFUL layer will immediately respond back to entity with relative response as per requested parameters.

6.2 Get Order Details

Entity will make a call to our web service exposed by RESTFUL layer which will internally make call to our database. RESTFUL layer will immediately respond back to entity with relative response as per requested parameters. Get Order service will return the all details of order.

Input parameters

End Point	Request Method	Value(json)	Response format
To be shared separately	POST	{"user":"Entityname","method":"getorders",”as_on”:” YYYY-MM-DD”,"from_date":"YYYY-MM-DD","to_date":" YYYY-MM-DD ","offset":"0","limit":"1"}	Json(payload)

Request Parameters

Parameter Name	Type	Mandatory	Description
Offset	Int	TRUE	Starting Index. When this parameter is set. The response will come as per the value of offset. If the Offset value is set to 5, the Response will start from the 5th record till the limit record.
Limit	Int	TRUE	Number of Orders Required
from_date	Date(YYYY-MM-DD)	TRUE (if as_on/to_date not set)	Start Date
to_date	Date(YYYY-MM-DD)	TRUE (if from_date is set)	End Date
transactionId	{clientcode}:{random 5 digit}:{date in yyyyMMddHHmmssSSS}	True	Transaction Id
authToken	Will be shared in email to client)	Yes	Auth Token
clientCode	String	Yes	Entity Name

Parameters description

1. **from_date** :It will response all the record from requested date with offset & limit value. The As_On parameter should not exist in the Request.
2. **to_date** : This parameter will not work without from_date.
3. **offset** : When this parameter is set. The response will come as per the value of offset. If the Offset value is set to 5, the Response will start from the 5th record till the limit record.
4. **limit** : limit is basically number of records.
5. ClientCode
6. authToken
7. transactionId

Output fields

Field Name	Length	Type	Mandatory	Description
transactionId	{clientcode}:{random 5 digit}:{date in yyyyMMddHHmmssSSS}		Yes	This will be a combination of client code, random 5 digit and date
responseStatus	1 (“S/F”)	String	Yes	Response Status i.e. Success or Failure
Message	60	String	Yes	Success Message or Error Message in case of failure
responseData	Variable (Decryption key will be shared over email)	String	Yes	Example: fEkhNhUDEOaj1pBTabI5aAOgOpOaw4qLHbm4Ydu4P8LlLe14n9jaeFoQ1KMFSTiu
responseHash	32	String	Yes	
encryptedResponseHas	32	String	Yes	

Field Name	Length	Type	Mandatory	Description
OrderId	100	String	TRUE	Unique Order Id
buyerName	100	String	TRUE	Name of Buyer
buyerAddress	255	String	TRUE	Address of Buyer
vendorName	100	String	TRUE	Name of Vendor/Seller
vendorAddress	255	String	TRUE	Address of Vendors/ Seller
vendorCode	80	String	TRUE	Unique code generated by PFMS
vendorDistrict	80	String	TRUE	District of the vendor
vendorState	80	String	TRUE	State of the vendor
vendorPin	6	Numeric	TRUE	Pin code of the vendor
vendorBankAccountNo	25	Numeric	TRUE	Account number of the vendor
vendorBankIfsc	20	String	TRUE	IFSC code of the Bank account
vendorPan	12	String	TRUE	PAN number of the vendor
supplyOrderNo	100	String	TRUE	Order number on contract
vendorGstn	30	String	TRUE	Vendor GSTN number

supplyOrderDate	Date	Date	TRUE	Date of Order
faAdminDesignation	100	String	TRUE	Designation of Financial approver
ifaConcurrance	2	Numeric	TRUE	Financial approval number
ifdNumber	100	String	TRUE	Internal finance department number
ifdDate	Date	Date	TRUE	Approval Date
productCode	150	String	TRUE	Product code as per GeM
productName	150	String	TRUE	Product name as per GeM
productBrand	100	String	TRUE	Product brand as per GeM
totalValue	15,2	Float		Total value of product in Rs.
consignee_Fname	20	String	TRUE	Consignee first name
consignee_Lastname	20	String	TRUE	Consignee last name
consignee_state	20	String	TRUE	State of consignee
consignee_district	20	String	TRUE	District of consignee
consignee_pin	6	Numeric	TRUE	Pincode of the consignee

Json Structure Changes

As per new structure for bunch items we have used new hierarchy where we have placed the multiple product inside the consignee level. Below is the structure for bunch item.

```
{
"transactionId": "BSF:12345:20181217131211928",
  "responseStatus": "S",
            "message": "Request Successfully Executed",
    "responseHash": "2458404B0C6686031A96FE01563565F1",
            "encryptedResponseHash": "711747832A6B7EA8F7941A298157D1AC",
            "responseData":
  "5+jb3t496Vg0irRxc8iJU7+wvjYH0iy8ikxhpAZwCkrTmA/A7gFh+rbO/hgUwUnDMIYmdud7dk3Gr3+zJc
G+5ET
  QG2FENPNxtJf39b8JasDDpzDAYkwFYIJ5JbZU13xQZxeEOzyE/Q3eTKugLFMO+ua20OSVMlYih/bbv
Qbo9w
  WmH/hnmYTN7KJPQWGeB+an7dK1l0H0EOkRqVDPwvbjuYQnMur/oIagEo+YUJatrvfhmC9bXB62jhv
WODjE
  yKGKTgEAktoMAgdCK30yxLQCgE/gV8k3c/A/EOtSLx/34Kzd3teGkgtAIYyy7hGKq83g8Hb0jfJXUY+P
bEo+go
  md2FR7zOIIpGy919BdJXNgfumY5/0fIR+19/r61exvx2STWKUOAO50ZtGQWsALFaywwgSqa8yIjbRIaU
Sw7qv   7jZLdCZb9UuLt7sfJStHgjhnN"

}
```

6.3 Bill Summary

Entity will make a call to our web service exposed by RESTFUL layer which will internally make call to our database. RESTFUL layer will immediately respond back to entity with relative response as per requested parameters.

Input parameters

End Point	Request Method	Value(json)	Response format
Will be shared separately	POST	{"clientCode":"BSF","transactionId":"BSF:12345:20181217131211928","fromDate":"2017-12-11","toDate":"2020-05-18","authToken":"7F3FB3C32E1E22A7FCD5E2D353724B3D"}	Json(payload)

Request Parameters

Parameter Name	Type	Mandatory	**Description**
from_date	Date(YYYY-MM-DD)	TRUE (if as_on/to_date not set)	Start Date
to_date	Date(YYYY-MM-DD)	TRUE (if from_date is set)	End Date
Offset	Int	TRUE	Starting Index
Limit	Int	TRUE	Number of Bills Required
clientCode	String	Yes	Entity Name
transaction Id	{clientcode}:{random 5 digit}:{date in yyyyMMddHHmmssSSS}	True	Transaction Id
authToken	Will be shared in email to client)	Yes	Auth Token

Parameters description

1. **from_date** :It will response all the record from requested date with offset & limit value. The As_On parameter should not exist in the Request.
2. **offset** : When this parameter is set. The response will come as per the value of offset. If the Offset value is set to 5, the Response will start from the 5th record till the limit record.
3. **limit** : limit is basically number of records.

Output Parameters

Field Name	**Length**	**Type**	**Mandatory**	**Description**
transactionId	{clientcode}:{random 5 digit}:{date in yyyyMMddHHmmssSSS}	String	Yes	This will be a combination of client code, random 5 digit and date
responseStatus	1 (“S/F”)	String	Yes	Response Status i.e. Success or Failure
Message	60	String	Yes	Success Message or

				Error Message in case of failure
responseData	Variable (Decryption key will be shared over email)	String	Yes	Example: fEkhNhUDEOaj1pBTabI5aAOgOpOaw4qLHbm4Ydu4P8LlLe14n9jaeFoQ1KMFSTiu
responseHash	32	String	Yes	Ex. 2458404B0C6686031A96FE01563565F1imestamp
encryptedResponseHash	32	String	Yes	Ex. 2458404B0C6686031A96FE01563565F1

ResponseData would have following fields:

Field Name	Length	Type	Mandatory	Description
date	String Date(YYYY-MM-DD)	String	Yes	Date between fromDate & toDate where billCount>0
count	Integer length	Integer	Yes	Number of bills on the above date
amount	String Variable	String	Yes	Sum of all the the amounts on the above date
gemInvoiceNos	List Variable	List	Yes	Invoice Numbers of above date

Below are the sample json output:

```
{
   "transactionId": "BSF:12345:20181217131211928",
   "responseStatus": "S",
   "message": "Request Successfully Executed",
   "responseHash": "2458404B0C6686031A96FE01563565F1",
   "encryptedResponseHash": "711747832A6B7EA8F7941A298157D1AC",
   "responseData":
"5+jb3t496Vg0irRxc8iJU7+wvjYH0iy8ikxhpAZwCkrTmA/A7gFh+rbO/hgUwUnDMIYmdud7dk3
Gr3+zJcG+5ETQG2FENPNxtJf39b8JasDDpzDAYkwFYIJ5JbZU13xQZxeEOzyE/Q3eTKugLFM
O+ua20OSVMlYih/bbvQbo9wWmH/hnmYTN7KJPQWGeB+an7dK1l0H0EOkRqVDPwvbjuYQn
Mur/oIagEo+YUJatrvfhmC9bXB62jhvWODjEyKGKTgEAktoMAgdCK30yxLQCgE/gV8k3c/A/E
OtSLx/34Kzd3teGkgtAIYyy7hGKq83g8Hb0jfJXUY+PbEo+gomd2FR7zOIIpGy919BdJXNgfumY
5/0fIR+19/r61exvx2STWKUOAO50ZtGQWsALFaywwgSqa8yIjbRIaUSw7qv7jZLdCZb9UuLt7sf
JStHgjhnN"
}
```

Here's the response which you'll get after decrypting the responseData:

```
{"dataField":[{"date":"2018-04-
12","count":5,"amount":"4058437.5","gemInvoiceNos":[{"gemInvoiceNo":"67"},{"gemInvoiceNo"
:"GEM-1527955310796"},{"gemInvoiceNo":"GEM-1527473395991"},{"gemInvoiceNo":"GEM-
1527072948608"},{"gemInvoiceNo":"GEM-1527568254029"}]},{"date":"2018-04-
18","count":5,"amount":"4997.0","gemInvoiceNos":[{"gemInvoiceNo":"GEM-
1523612155645"},{"gemInvoiceNo":"GEM-1524031078141"},{"gemInvoiceNo":"GEM-
1524031451357"},{"gemInvoiceNo":"GEM-1524031451357"},{"gemInvoiceNo":"GEM-
1523612155645"}]},{"date":"2018-05-
01","count":6,"amount":"3958.9","gemInvoiceNos":[{"gemInvoiceNo":"GEM-
1524819437227"},{"gemInvoiceNo":"GEM-1525158428886"},{"gemInvoiceNo":"GEM-
1517202029692"},{"gemInvoiceNo":"GEM-1525160053938"},{"gemInvoiceNo":"GEM-
1525160017534"},{"gemInvoiceNo":"GEM-1525174232314"}]},{"date":"2018-05-
16","count":1,"amount":"164.16","gemInvoiceNos":[{"gemInvoiceNo":"GEM-
1526457347346"}]},{"date":"2018-06-
02","count":1,"amount":"35000.0","gemInvoiceNos":[{"gemInvoiceNo":"GEM-
1527575330536"}]},{"date":"2018-06-
05","count":1,"amount":"120.0","gemInvoiceNos":[{"gemInvoiceNo":"GEM-
1528193731558"}]},{"date":"2018-06-
07","count":2,"amount":"1447351.86","gemInvoiceNos":[{"gemInvoiceNo":"GEM-
1527568254029"},{"gemInvoiceNo":"GEM-1527955310796"}]},{"date":"2018-06-
22","count":1,"amount":"44.0","gemInvoiceNos":[{"gemInvoiceNo":"GEM-1529650497981"}]}]}
```

6.4 Get Bill Details

Entity will make a call to our web service exposed by RESTFUL layer which will internally make call to our database. RESTFUL layer will immediately respond back to entity with relative response as per requested parameters. Get Bill service will return the all details of Bill after confirmation of buyer bill process.

Input parameters

End Point	Request Method	Value(json)	Response format
To be shared separately	POST	{"clientCode":"BSF","transactionId":"BSF:12345:20181217131211928","fromDate":"2017-12-11","toDate":"2020-05-18","authToken":"7F3FB3C32E1E22A7FCD5E2D353724B3D","offset":0,"limit":100}	Json(payload)

Request Parameters

Parameter Name	Type	Mandatory	**Description**
from_date	Date(YYYY-MM-DD)	TRUE (if as_on/to_date not set)	Start Date
to_date	Date(YYYY-MM-DD)	TRUE (if from_date is set)	End Date
transaction Id	{clientcode}:{random 5 digit}:{date in yyyyMMddHHmmssSSS}	True	Transaction Id
authToken	Will be shared in email to client)	Yes	Auth Token
clientCode	String	Yes	Entity Name

Parameters description

1. **from_date** :It will response all the record from requested date with offset & limit value. The As_On parameter should not exist in the Request.
2. **to_date** : This parameter will not work without from_date.
3. **offset** : When this parameter is set. The response will come as per the value of offset. If the Offset value is set to 5, the Response will start from the 5th record till the limit record.
4. **limit** : limit is basically number of records.
5. AuthToken
6. TransactionId
7. ClientCode

Output fields

Field Name	Length	Type	Mandatory	Description
vendorDistrict	80	String	TRUE	District of the vendor
vendorState	80	String	TRUE	Status of the vendor
vendorPin	6	Numeric	TRUE	Pincode of the vendor
vendorBankAccountNo	25	Numeric	TRUE	Bank account number of the vendor
vendorBankIfsc	20	String	TRUE	IFSC code of the Bank account number
vendorPan	12	String	TRUE	PAN number of the vendor
ifdDate	Date	Date	TRUE	Vendor GSTN number

productCode	150	String	TRUE	Product Code as per GeM

Field Name	**Length**	**Type**	**Mandatory**	**Description**	**Parent Element**
productName	150	String	TRUE	Name of the product as per GeM	Body
productBrand	100	String	TRUE	Product brand as per GeM	Body
totalValue	15,2	Float	TRUE	Total value in Rs. of the product	Body
consignee_Fname	20	String	TRUE	Consignee first name	Body
consignee_Lastname	20	String	TRUE	Last name of the consignee	Body
consignee_state	20	String	TRUE	Consignee state	Body
consignee_district	20	String	TRUE	Consignee district	Body
consignee_pin	6	Numeric	TRUE	Pin code of consignee	Body
consignee_address	255	String	TRUE	Address of the consignee	Body
consignee_mobile	13	Numeric	TRUE	Consignee mobile number	Body
Gstn	20	String	TRUE	GSTN number	Body
expectedDeliveryDate	Date	Date	TRUE	Expected delivery date	Body
quantityOrdered	10	Numeric	TRUE	Quantity of the product ordered	Body
unitPrice	15,2	Float	TRUE	Price of each unit of the quantity	Body
quantityUnitType	10	String	TRUE	Type of Unit for the quantity .	Body
billAmount	15,2	Float	TRUE	Amount of the bill	Body

Field Name	Length	Type	Mandatory	Description	Parent Element
billDate	Date	date	TRUE	Date of bill	
billNo	100	String	TRUE	Bill number	Body
suppliedQuantity	10	Numeric	TRUE	No of Quantity supplied	Body
frieghtCharge • CGST • SGST_UTGST_IGST • Cess	10 10 10 10	Numeric Numeric Numeric Numeric	TRUE	Freight charges over & above the product value	Body Body Body Body
invoiceDate	Date	string	TRUE	Date of invoice generation	Body Body
invoiceNo	100	String	TRUE	Invoice number	Body
ldAmount	10	Numeric	OPTIONAL	Liquidation damage amount	Body
ldDays	2	Numeric	OPTIONAL	Delay in number of days for delivery	Body
invoiceFile	TEXT	TEXT	TRUE	PDF URL	Body
faFile	TEXT	TEXT	TRUE	PDF URL	Body
cracFile	TEXT	TEXT	TRUE	PDF URL	Body
contractFile	TEXT	TEXT	TRUE	PDF URL	Body
billFile	TEXT	TEXT	TRUE	PDF URL	Body
cracDate	Date	Date	TRUE	Date of CRAC	Body
cracNo	20	String	TRUE	CRAC number	Body
amountClaimed	15,2	Float	TRUE	Amount	Body
receiptNo	20	String	TRUE	CRAC number	Body
receiptDate	Date	Date	TRUE	CRAC date	Body

Field Name	Length	Type	Mandatory	Description
transactionId	{clientcode}:{random 5 digit}:{date in yyyyMMddHHmmssSSS}		Yes	This will be a combination of client code, random 5 digit and date
responseStatus	1 (“S/F”)	String	Yes	Response Status i.e. Success or Failure
Message	60	String	Yes	Success Message or Error Message in case of failure
responseData	Variable (Decryption key will be shared over email)	String	Yes	Example: fEkhNhUDEOaj1pBTabI5aAOgOpOaw4qLHbm4Ydu4P8LlLe14n9jaeFoQ1KMFSTiu
responseHash	32	String	Yes	Ex. - 2458404B0C6686031A96FE01563565F1
encryptedResponseHash	32	String	Yes	Ex. - 2458404B0C6686031A96FE01563565F1

Json Structure Changes

As per new structure for bunch items we have used new hierarchy where we have placed the multiple product inside the consignee level. Below is the structure for bunch item. In bill service few fields extra when buyers proceeds the bill.

{ As per new structure for bunch items we have used new hierarchy where we have placed the multiple product inside the consignee level. Below is the structure for bunch item. In bill service few fields extra when buyer proceed the bill.

```
                        {
{{
                 "transactionId": "BSF:12345:20181217131211928",
                  "responseStatus": "S",
                  "message": "Request Successfully Executed",
                  "responseHash": "2458404B0C6686031A96FE01563565F1",
                  "encryptedResponseHash": "711747832A6B7EA8F7941A298157D1AC",
                 "responseData":
        "5+jb3t496Vg0irRxc8iJU7+wvjYH0iy8ikxhpAZwCkrTmA/A7gFh+rbO/hgUwUnDMIYmdud7dk3Gr3+zJcG+5ET
        QG2FENPNxtJf39b8JasDDpzDAYkwFYIJ5JbZU13xQZxeEOzyE/Q3eTKugLFMO+ua20OSVMlYih/bbvQbo9w
        WmH/hnmYTN7KJPQWGeB+an7dK1l0H0EOkRqVDPwvbjuYQnMur/oIagEo+YUJatrvfhmC9bXB62jhvWODjE
        yKGKTgEAktoMAgdCK30yxLQCgE/gV8k3c/A/EOtSLx/34Kzd3teGkgtAIYyy7hGKq83g8Hb0jfJXUY+PbEo+go
        md2FR7zOIIpGy919BdJXNgfumY5/0fIR+19/r61exvx2STWKUOAO50ZtGQWsALFaywwgSqa8yIjbRIaUSw7qv
        7jZLdCZb9UuLt7sfJStHgjhnN"
                }
```

reponseData would have following fields after decryption

```
{
        "dataField": [
        {
                "buyerAddr": "df-324354",
                "buyerDistrict": "dist",
                "buyerEmail": "test@gem.gov.in",
                 "buyerGstn": "BUYERGSTN3435",
                "buyerMobile": "2354646768",
                "buyerName": "Buyer one",
                "buyerOrg": "Home Mistry",
                "buyerPincode": "2343543546",
                "buyerState": "Delhi",
                "demandId": "511687723878506",
                "designationFinancial": "DSFddf",
                 "ifdConcurrance": 45,
                "ifdDiaryDate": "2018-04-13",
                "ifdDiaryNo": "45465464",
                "orderAmount": null,
                "orderDate": null,
                "orderId": "GEMC-511687732882564",
               "sellerId": "3454656",
               "supplyOrderDate": "2018-04-26",
                "supplyOrderNo": "GEM-3243M23",
                "vendorAddress": "Gem Hall",
                "vendorBankAccountNo": "5556623324500034",
                "vendorBankIfscCode": "IFSC000334",
                "vendorCode": "vendor345",
                "vendorDistrict": "Testing",
                "vendorGstn": "VENDORGSTN3243535",
                "vendorName": "vendorName",
                "vendorPan": "BHN232443",
                "vendorPincode": "232434",
                "vendorState": "Delhi",
"billNo": "DG325345345",
"billDate": "2018-04-12",
"billAmount": "1014600.0",
"faFile": null,
"cracFile": null,
"contractFile": null,
"receiptNo": null,
 "receiptDate": "2018-04-12 10:25:25.0",
"cracDate": "2018-04-19 09:26:26.0",
"billFile": null,
"invoiceFile": null,
 "invoiceDate": "2018-04-12",
 "invoiceNo": "sdfsdfsf",
 "gemInvoiceNo": "GEM-1527568254029",
 "ldDays": "4",
"ldAmount": "34546.66",
 "consignmentDetails": [
         {
          "consigneeState": null,
        "consigneeLastname": null,
        "consigneeMobile": null,
         "consigneeFname": null,
       "consigneePin": null,
       "consigneeAddress": null,
        "consigneeDistrict": null,
        "products": [
         {
   "totalValue": null,
   "unitPrice": "5445",
   "productCode": "DG34",
   "expectedDeliveryDate": "2018-04-26 00:00:00",
   "productBrand": "B",
   "quantityOrdered": null,
   "quantityUnitType": "KG",
   "productName": "dsfgh"
           },
          {
         "totalValue": null,
      "unitPrice": "5445",
      "productCode": "DG34",
      "expectedDeliveryDate": "2018-04-26 00:00:00",
      "productBrand": "B",
       "quantityOrdered": null,
      "quantityUnitType": "KG",
       "productName": "dsfgh"
               }
                      ]
                      }
                       ]
                      }
                      ]
                     }
```

6.5 Payment status

The GeM will make a call to the ERP Endpoint to pull the details of the Payment.

- The Payment Request will have the following Information
  - Transaction Id – The Unique Transaction id
  - Status – This will capture the status of the Payment. The allowed values are Success/fail
  - Payment By – The allowed values can be Cheque, Demand Draft, Internet banking, RTGS, NEFT, Other
  - Contract No – GeM Contract number
  - GeM Invoice Number – The Invoice number Generated on GeM
  - Bill No – Bill Number generated by GeM
  - Bill Amount paid – Amount paid to the Seller
  - Transaction date – Date of Transaction
  - Deducted Amount – Amount Deducted
  - Deduction Type – Type of Deduction
  - Bank Name – Bank through which the payment is Made
  - Cheque Number – Cheque Number
  - Bank Transaction number – Bank Transaction Number applicable in case of internet Banking, RTGS,NEFT
  - Demand Draft Number – Demand Draft Number
  - Sanction number – sanction Number if applicable
  - Sanction date – date of sanction if applicable
  - Return Reason – Reason for the Return of Payment

- The Payment Response will have the following fields:
  - Transaction Id – Transaction id Received in request
  - Status – The status of Transaction – Success/fail
  - Payment Mode – Mode of Payment
  - Message – Success Message or Error Message in case of failure

7. Action items for CPSU and other Government entities using ERP

As a next step / action item, the CPSU and other government entities using ERP, would be required to carry out the following tasks:

- Identify a techno functional team from their end to carryout out the integration with the above API exposed by GeM Platform. The details of the techno functional team should be communicated to CTO GeM SPV at rajesh[dot]narang[at]gem[dot] gov[dot]in
- Carryout the integration requirements at their end and consume the information provided by the GeM exposed API and return the payment related required details to GeM portal.
- As part of the UAT testing of the integration, both GeM and the CPSU would be required to be part of the testing. GeM would push test transactions to the CPSU system and the CPSU would be required to consume the data in their ERP.

***************END OF DOCUMENT**************