HCL Commerce V9.1

PunchOut2Go Integration with HCL Commerce

Dec 29, 2020

Document Source

Version #	Date	Version Description and Summary of changes	Editor
0.1	Dec 29, 2020	First version for review	Rahul Yewale

Introduction

This document is about the integration of HCL Commerce with PunchOut2Go where HCL Commerce provides commerce functionality and PunchOut2Go provides two-way integration between an HCL Commerce store and hundreds of e-procurement and ERP platforms. With PunchOut2Go, B2B sellers can provide WebSphere punchout to their customers without the expense of manual integration.

Currently HCL Commerce has Generic Procurement System Integration but it uses the Messaging Queue Technique for communication between Commerce and Procurement System.

In new approach will implement the integration using REST API approach so that it will be easy to integrate the same with Sapphire (B2B React Store).

Prerequisites:

- Buyer and Supplier Organization should be created and approved
- Under Buyer Organization Procurement Buyer Administrator User should be created and same should be approved
- Contract between Buyer and Supplier Organization should be placed.
- All the Database queries should be created with above data and same should be executed.

Scope of the POC

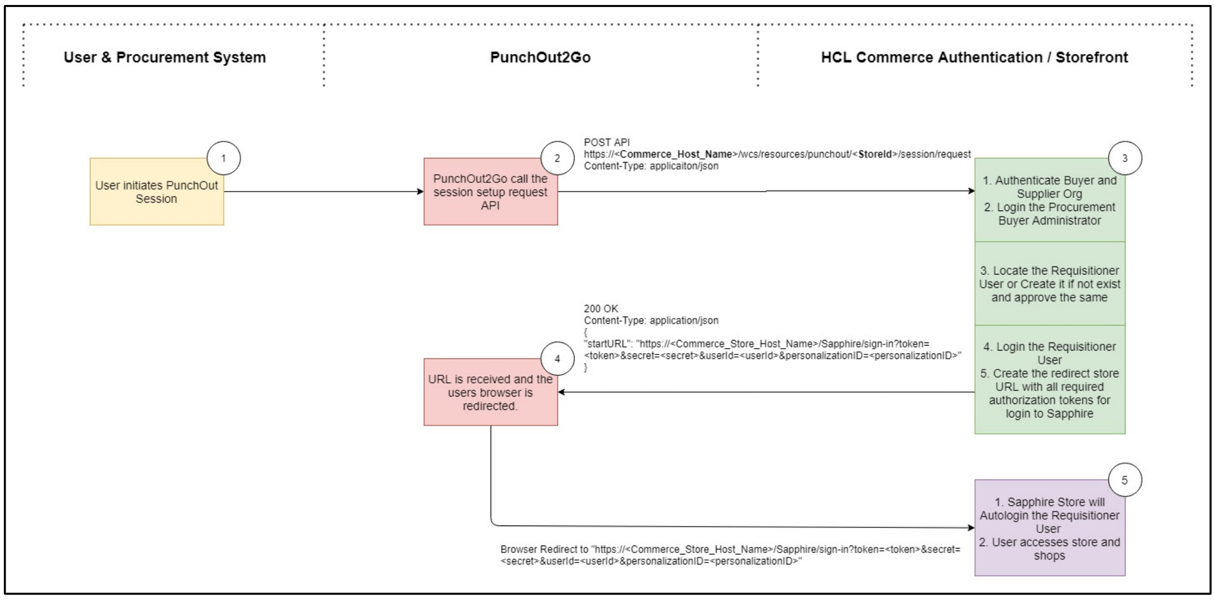
- Session setup:
  1. Buyer and Supplier Org Authentication
  2. Create the Requisitioner User with default Password.
  3. Login the Requisitioner User and create the Auto Login URL with all required tokens.
- Transfer Cart:
  1. Using autologin URL, login the user to Sapphire.
  2. User can search and add the products to the cart.
  3. Transfer the cart data to Punchout2Go and redirect the User to redirect URL provided by Punchout2Go.
- Order Submission:
  1. Create the Order for items which are approved from procurement system.
  2. "COD" as Default payment method used.

Out of Scope for this POC

- Commerce will not maintain the Order History it will be taken care by Punchout/Procurement.

Design Overview

1. Session Setup



User & Procurement System
PunchOut2Go
HCL Commerce Authentication / Storefront
1
User initiates PunchOut Session
2
PunchOut2Go call the session setup request API
POST API
https://<Commerce_Host_Name>/wcs/resources/punchout/<StoreId>/session/request
Content-Type: applicaiton/json
3
1. Authenticate Buyer and Supplier Org
2. Login the Procurement Buyer Administrator
3. Locate the Requisitioner User or Create it if not exist and approve the same
4. Login the Requisitioner User
5. Create the redirect store URL with all required authorization tokens for login to Sapphire
200 OK
Content-Type: application/json
{
"startURL": "https://<Commerce_Store_Host_Name>/Sapphire/sign-in?token=<token>&secret=<secret>&userId=<userId>&personalizationID=<personalizationID>"
}
4
URL is received and the users browser is redirected.
Browser Redirect to "https://<Commerce_Store_Host_Name>/Sapphire/sign-in?token=<token>&secret=<secret>&userId=<userId>&personalizationID=<personalizationID>"
5
1. Sapphire Store will Autologin the Requisitioner User
2. User accesses store and shops

Steps:

1) User initiates Punchout Session

- Procurement Buyer initiate the Punchout request by selecting HCL Commerce Store as Supplier.
- Procurement system will call PunchOut2Go API with Requisitioner information

2) PunchOut2Go call the session setup request API

- PunchOut2Go receives the request from any Procurement system for HCL Commerce Store Session creation
- PunchOut2Go call the HCL Commerce “PunchOut2Go Session Creation” REST API.

3) HCL Commerce Punchout Session Creation REST API

Need to **Create new REST API** in HCL Commerce which will be responsible to perform following tasks.

1. Authenticate Buyer and Supplier Org
   - Authenticate Buyer and Supplier Organizations using Org Code and Code type provided by PunchOut2Go in request payload
   - If authentication is successful, then only proceed to next step otherwise send the appropriate Error message to Punchout2Go.
2. Login the Procurement Buyer Administrator
   - Login Procurement Buyer Administrator using Credentials provided by PunchOut2Go in request payload

- Procurement Buyer administrator login will be required because using admin login we can register the Requisitioner user for Buyer Organization
- If login is successful, then only proceed to next step otherwise sent the appropriate Error message to Punchout2Go.

3. Locate the Requisitioner User or Create it if not exist and approve the same
   - Check the Requisitioner user is already register or not.
   - If already register, then the fetch the Requisitioner user
   - If not registered already then Create new Requisitioner User
   - If fetching / creating requisitioner user is successful, then proceed to next step otherwise send the appropriate Error message to Punchout2Go
4. Login the Requisitioner User
   - Login the Requisitioner User fetched/created in previous step.
   - Create/fetch the following information required for Login to Sapphire
     - WCToken
     - WCTrustedToken
     - personalizationID
     - userId
5. Create the redirect store URL with all required authorization tokens for login to Sapphire
   - Create the store autologin URL using information created/fetched in previous step
   - Example Response:

```
{
    "startURL": "https://<Commerce_Store_Host_Name>/Sapphire/sign-in?token=<WCToken>&secret=<WCTrustedToken>&userId=<userId>&personalizationID=<personalizationID>&source=punchOut2Go"
}
```

4) URL is received and the user's browser is redirected

- PunchOut2Go receives the success response (200 OK) with startURL in response body
- PunchOut2Go will redirect the User to startURL on browser.

5) Sapphire Store Autologin and Shopping

- Sapphire store will autologin the Requisitioner User using required information provided in startURL query string
- Once successful login, Requisitioner user can search the products and add them into the cart.

Below is the REST API details for Commerce PunchOut2Go Session creation:

New REST API is created for PunchOut2Go Session creation

- Level1 => logonMode: create

Request	**REST API Endpoint:**
	https://<Commerce_Host_Name>/wcs/resources/punchout/<StoreId>/session/request
	Method: POST
	Header:
	Content-Type: application/json
	
	Request Payload:
	Sample request payload
	`{`
	`  "protocolName": "JSON-WSP",`
	`  "protocolVersion": "1.0",`
	`  "supplierCode": "<Seller Organization>",`
	`  "supplierCodeType": "DUNS",`
	`  "buyerCode": "<Buyer Organization>",`
	`  "buyerCodeType": "DUNS",`
	`  "buyerUserID": "<buyerAdminUserID>",`
	`  "buyerPassword": "<buyerAdminPassword>",`
	`  "logonMode": "create",`
	`  "reqId": "<requisitionerId>",`
	`  "reqName": "<requisitionerName>",`
	`  "reqEmailId": "<requisitionerEmail>"`
	`}`

Response	Sample Success Response
	200 OK
	Content-Type: application/json
	`{`
	` "startURL": "https://<Commerce_Store_Host_Name>/Sapphire/sign-in?token=13002%2CduAk5s4RMBJlvaD7nlDX4BanK1ASmyK%2BK0lJts6ZxYjgDCyEM4wb0dzxrlT4XY7b7Z3Mp4rzcTD302vmXemlwTSrg6YnOYGhAaj%2BNTyx0jNZo85U%2B%2BIKpBkIaMN8L4WCnl2D6ZU3tmJ3DixknnaHohRGcqOfDk3KAOINuiioqLY8YJtIa8MAs5%2Fsc8x%2B%2BbXlBeacrSR72fiKi42F7dKvrhW%2F8uUb9nCC13UUQq25CASrKFbYSU7A0TDn8GnQkZIpaezqpnPZnCWcUb9zXYMTJQ%3D%3D&secret=13002%2CVhzuzo0wg7%2FjknUty6UX974s2jzvp6p9Y%2BRDQLDTdTs%3D&userId=13002&personalizationID=1608547106222-1&source=punchOut2Go"`
	`}`

- Level2 => logonMode: edit

Request

REST API Endpoint:
https://<Commerce_Host_Name>/wcs/resources/punchout/<StoreId>/session/request
Method: POST
Header:
Content-Type: application/json

Request Payload:
Sample request payload

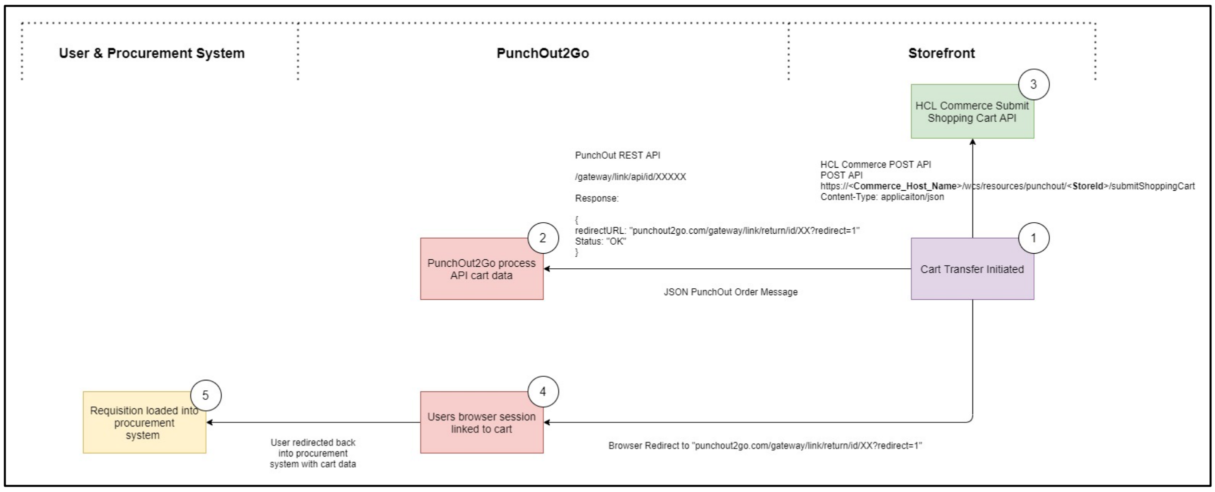
```
{
  "protocolName": "JSON-WSP",
  "protocolVersion": "1.0",
  "supplierCode": "<Seller Organization>",
  "supplierCodeType": "DUNS",
  "buyerCode": "<Buyer Organization>",
  "buyerCodeType": "DUNS",
  "buyerUserID": "<buyerAdminUserID>",
  "buyerPassword": "<buyerAdminPassword>",
  "logonMode": "edit",
  "quoteNumber": "<hcl_commerce_order_id>",
  "reqId": "<requisitionerId>",
  "reqName": "<requisitionerName>",
  "reqEmailId": "<requisitionerEmail>"
}
```

Response

Sample Success Response
200 OK
Content-Type: application/json

```
{
  "startURL": "https://<Commerce_Store_Host_Name>/Sapphire/sign-
in?token=13002%2CduAk5s4RMBJlvaD7nlDX4BanK1ASmyK%2BK0lJts6ZxYjgDC
yEM4wb0dzxrlT4XY7b7Z3Mp4rzcTD302vmXemlwTSrg6YnOYGhAaj%2BNTyx0jN
Zo85U%2B%2BIKpBkIaMN8L4WCnl2D6ZU3tmJ3DixknnaHohRGcqOfDk3KAOIN
uiioqLY8YJtIa8MAs5%2Fsc8x%2B%2BbXlBeacrSR72fiKi42F7dKvrhW%2F8uUb9n
CC13UUQq25CASrKFbYSU7A0TDn8GnQkZIpaezqpnPZnCWcUb9zXYMTJQ%3D%
3D&secret=13002%2CVhzuzo0wg7%2FjknUty6UX974s2jzvp6p9Y%2BRDQLDTd
Ts%3D&userId=13002&personalizationID=1608547106222-
1&source=punchOut2Go"
}
```

2. Transfer Cart


User & Procurement System
PunchOut2Go
Storefront
3
HCL Commerce Submit Shopping Cart API
PunchOut REST API
/gateway/link/api/id/XXXXX
Response:
{
redirectURL: "punchout2go.com/gateway/link/return/id/XX?redirect=1"
Status: "OK"
}
HCL Commerce POST API
POST API
https://<Commerce_Host_Name>/wcs/resources/punchout/<StoreId>/submitShoppingCart
Content-Type: applicaiton/json
2
PunchOut2Go process API cart data
1
Cart Transfer Initiated
JSON PunchOut Order Message
5
Requisition loaded into procurement system
4
Users browser session linked to cart
User redirected back into procurement system with cart data
Browser Redirect to "punchout2go.com/gateway/link/return/id/XX?redirect=1"

Steps:

1) Cart Transfer Initiated

- Once Requisitioner user successfully logged in to Sapphire store, user can search for the product he/she want to order.
- Add the required products into the cart
- Sapphire will disable the checkout flow for Punchout Requisitioner user if user is auto logged in with token provided on query string
- In cart screen, “Transfer Cart” button is displayed instead of “Checkout” button to transfer the order back to procurement system through PunchOu2Go.
- User clicks on “Transfer Cart” button.
- Sapphire will call PunchOut2Go REST API to transfer the cart

2) PunchOut2Go process API cart data

- PunchOut2Go will process the “Transfer Cart” request initiated by HCL store.
- After successful processing, PunchOut2Go will success response with “redirectURL”.

3) HCl Commerce Submit Shopping Cart API:

New REST API is created in Transaction server to submit the shopping cart

- Sapphire will receive the response from PunchOut2Go with “redirectURL”
- On successful response from “PunchOut2Go Transfer Cart” API, Sapphire store will call the commerce “Submit Shopping Cart” REST API to change the order and order item status from “P” (Pending Order) to “W” (Approval Pending”)

4) Users browser session linked to cart

- On successful response from Commerce “Submit Shopping Cart” REST API, Sapphire will redirect the user to “redirectURL”
- PunchOut2Go will redirect the user to procurement system

5) Requisition loaded into procurement system

- User will be redirected to Procurement system by PunchOut2Go with requisition loaded.
- Requisition is sent for approval’s
- Waiting for approval

Below is the REST API details for PunchOut2Go Transfer Cart:

Request

REST API Endpoint:
https://connect.punchout2go.com/gateway/link/api/id/xxxxxxx
Method: POST
Header:
Content-Type: application/json

Request Payload:
Sample request payload

```
{
  "total":"<total_price>",
  "items":[
     {
        "supplierid":"<hcl_commerce_part_number>",
        "supplierauxid":"<hcl_commerce_order_id>/1",
        "description":"<hcl_commerce_item_name>",
        "classification":"",
        "uom":"<hcl_commerce_uom>",
        "unitprice":"<itemPrice>",
        "quantity":"<quantity>"
     }
  ]
}
```

Response

Sample Success Response
200 OK
Content-Type: application/json

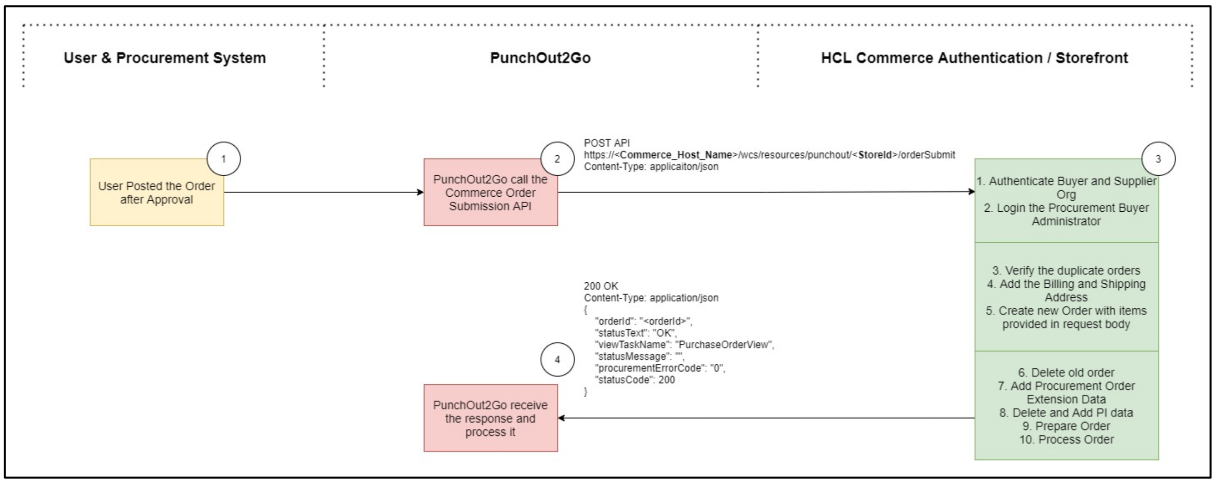
```
{
  "redirect_url":"https:\/\/connect.punchout2go.com\/gateway\
/link\/return\/id\/xxxxxxx?redirect=1"
}
```

Below are the REST API details for Commerce Submit Shopping Cart

Request	**REST API Endpoint:**
	https://<Commerce_Host_Name>/wcs/resources/punchout/<StoreId>/submitShoppingCart
	Method: POST
	Header:
	`headers: {`
	`  "Content-Type": "application/json",`
	`  "WCToken": <WCToken>,`
	`  "WCTrustedToken": <WCTrustedToken>`
	`}`
	
	Request Payload:
	Sample request payload
	`{`
	`  "orderId": "<hcl_commerce_order_id>"`
	`}`

Response	Sample Success Response
	200 OK
	Content-Type: application/json
	`{`
	`  "orderId": <hcl_commerce_order_id>,`
	`  "viewTaskName": "SubmitShoppingCartView"`
	`}`

3. Order Submit


User & Procurement System
PunchOut2Go
HCL Commerce Authentication / Storefront
1
User Posted the Order after Approval
2
PunchOut2Go call the Commerce Order Submission API
3
1. Authenticate Buyer and Supplier Org
2. Login the Procurement Buyer Administrator
3. Verify the duplicate orders
4. Add the Billing and Shipping Address
5. Create new Order with items provided in request body
6. Delete old order
7. Add Procurement Order Extension Data
8. Delete and Add PI data
9. Prepare Order
10. Process Order
4
PunchOut2Go receive the response and process it

```
POST API
https://<Commerce_Host_Name>/wcs/resources/punchout/<StoreId>/orderSubmit
Content-Type: applicaiton/json
```

```
200 OK
Content-Type: application/json
{
   "orderId": "<orderId>",
   "statusText": "OK",
   "viewTaskName": "PurchaseOrderView",
   "statusMessage": "",
   "procurementErrorCode": "0",
   "statusCode": 200
}
```

Steps:

1) Users initiate the order submission after approval

- Procurement Buyer initiate the order submission flow once he/she receives the approval for quote.

- Procurement system will call PunchOut2Go API with Requisitioner information and order information.

2) PunchOut2Go call the commerce order submission API

- PunchOut2Go receives the request from any Procurement system for HCL Commerce order submission
- PunchOut2Go call the HCL Commerce “PunchOut2Go Order Submission” REST API.

3) HCL Commerce Punchout Order Submission REST API

Need to **Create new REST API** in HCL Commerce which will be responsible to perform following tasks.

1. Authenticate Buyer and Supplier Org
   - Authenticate Buyer and Supplier Organizations using Org Code and Code type provided by PunchOut2Go in request payload
   - If authentication is successful, then only proceed to next step otherwise send the appropriate Error message to Punchout2Go.
2. Login the Procurement Buyer Administrator
   - Login Procurement Buyer Administrator using Credentials provided by PunchOut2Go in request payload
   - Procurement Buyer administrator login will be required because using admin login we can submit the Requisitioner user order for Buyer Organization
   - If login is successful, then only proceed to next step otherwise sent the appropriate Error message to Punchout2Go
3. Verify the duplicate order submission
   - If duplicate order false, then only proceed to next step otherwise sent the appropriate Error message to Punchout2Go
4. Create Billing and Shipping address provided in request payload
5. Create the new order with items array provided in request payload
6. Delete the old order
   - Change the order and order item status from “W” (Approval Pending) to “X” (Canceled).
7. Add procurement order and order item extension data.
8. Delete and add PI data
   - Delete the old Payment Instruction
   - Add new payment instruction with default payment method as “COD”
9. Prepare the order using existing Commerce commands
10. Process the order using existing Commerce commands

4) PunchOut2Go receive the response and process it

- PunchOut2Go will receive the response from HCL commerce Order Submission REST API.
- PunchOut2Go will do further processing with procurement system.

Below is the REST API details for PunchOut2Go Order Submission:

New REST API is created for PunchOut2Go Order Submission

Request	**REST API Endpoint:**
	https://<Commerce_Host_Name>/wcs/resources/punchout/<StoreId>/orderSubmit
	Method: POST
	Header:
	Content-Type: application/json
	Request Payload:
	Sample request payload

```
{
  "protocolName":"JSON-WSP",
  "protocolVersion":"1.0",
  "supplierCode": "<Seller Organization>",
  "supplierCodeType": "DUNS",
  "buyerCode": "<Buyer Organization>",
  "buyerCodeType": "DUNS",
  "buyerUserID": "<buyerAdminUserID>",
  "buyerPassword": "<buyerAdminPassword>",
  "reqId":"<requisitionerId>",
  "quoteNumber":"<hcl_commerce_order_id>",
  "order_billTo_vector":[
     {
        "name":"<Address_Name>",
        "deliverTo":"",
        "streetAddress1":"<streetAddress1>",
        "streetAddress2":"",
        "streetAddress3":"",
        "city":"<city>",
        "state":"<state>",
        "country":"<country>",
        "postalCode":"<postalCode>",
        "email":"<email>",
        "emailType":"",
        "telefaxNumber":"",
        "faxCountryCode":"",
        "faxAreaCode":"",
        "faxNumber":"",
        "faxType":"",
        "telephoneNumber":"",
            "phoneCountryCode":"",
            "phoneAreaCode":"",
            "phoneNumber":"",
            "telephoneType":""
        }
    ],
    "order_shipTo_vector":[
        {
            "name":"<Address_Name>",
            "deliverTo":"",
            "streetAddress1":"<streetAddress1>",
            "streetAddress2":"",
            "streetAddress3":"",
            "city":"<city>",
            "state":"<state>",
            "country":"<country>",
            "postalCode":"<postalCode>",
            "email":"<email>",
            "emailType":"",
            "telefaxNumber":"",
            "faxCountryCode":"",
            "faxAreaCode":"",
            "faxNumber":"",
            "faxType":"",
            "telephoneNumber":"",
            "phoneCountryCode":"",
            "phoneAreaCode":"",
            "phoneNumber":"",
            "telephoneType":""
        }
    ],
    "order_items_vector":[
        {
            "quantity":"<quantity>",
            "requestDeliveryDate":"",
            "quoteNumber":"<hcl_commerce_order_id>",
            "itemID":"<hcl_commerce_part_number>",
            "itemPrice":"<itemPrice>",
            "itemDescription":"",
            "unitOfMeasure":"<hcl_commerce_uom>",
            "classificationCode":"",
            "classificationDomain":"",
            "shipmodeId":"15501"
        }
    ],
    "messageId": "<hcl_commerce_order_id>",
    "buyerOrderDate":"",
    "buyerOrderID":"<hcl_commerce_order_id>",
```

	```
	  "orderMode":"",
	  "totalAmount":"<totalAmount>",
	  "orderStatusUrl":"",
	  "shipmodeId":"15501",
	  "comment":""
	}
	```

Response	Sample Success Response
	200 OK
	Content-Type: application/json
	```
	{
	  "orderId": "<hcl_commerce_order_id>",
	  "statusText": "OK",
	  "viewTaskName": "PurchaseOrderView",
	  "statusMessage": "",
	  "procurementErrorCode": "0",
	  "statusCode": 200
	}
	```

Note: Placeholder parameters from request payload are mandatory.