**OMS POC – Order Notification**

**Scope**:

The main purpose of this task is to bring the same order status for MAI and Shopify. Maintain the up-to-date payment status in MAI and Shopify. If merchant can change the status of the Order in MAI this should be updated in Middleware and Shopify. So, all three (Shopify, Middleware and MAI) should be in same status.

**Current Scenario**:

In our current scenario if we change order status in MAI, the order status gets changed in our middleware but in Shopify changes are not updated. Shopify will show old status. Because gid is not created from Shopify side. gid is created when any of the option (Authorize, Capture, Cancel and Refund) is done in Shopify. Without gid no order status can be changed in Shopify through middleware. This is the flow of Shopify OMS update from third party apps.  
  
**Overcome this scenario:**

When researched for Shopify API feature to get rid of this issue, we found that through transaction Admin API and Webhook we can overcome this issue. But to achieve this we need **read\_orders** and **write\_orders** access scopes. To use these access scopes in our payment apps we need permission from Shopify Team.  
  
**Steps to Achieve:**

We can divide this task into two parts,

1. Format of MAI Order notification URL. Once the URL is triggered the changes that should be updated in Middleware
2. Based on the triggered notification URL the status should be updated in Shopify

**Step 1: MAI - Order Notification**

In MAI there is a space to add notification URL (See Image1). If we change order status in MAI, the notification URL of the MAI will be triggered. Once middleware receives the triggered URL it will change the order status in middleware. In case middleware failed to update the order status we will use CRON to sync the status.

Graphical user interface, application

Description automatically generated

Image 1:

We will face hurdle here because Shopify uses SaaS concept, and all the merchant will login to single middleware. All merchants will have same notification URL.

**Solution:**

The best way to overcome notification URL issue is to use Merchant entity reference or Merchant Name along with URL. Because every merchant will have unique Merchant entity reference and Merchant Name.  
  
Graphical user interface, application

Description automatically generated

We can use Merchant Name along with URL, like below

https://tryshopify.access-worldpay.com/notification/awpsapienttest

If we set this URL in MAI, this is how the response will be

{   "eventId": "58f3a259-6e57-47ae-970e-ee0a132d5a47",     "eventTimestamp": "2023-03-24T07:59:00.469",     "eventDetails": {         "classification": "payment",         "transactionReference": "rNcsmjsOxX0aV9jSECi5maNfJ",         "type": "sentForAuthorization",         "date": "2023-03-24",         "amount": {             "value": 45.99,             "currencyCode": "GBP"         },         "\_links": {             "payment": {                 "href": ""             }   } } }

**Step 2: Shopify API**

We will focus on Transaction Admin API and Webhook to update the status in Shopify when we change status in MAI.

**Shopify API:**

Extend Shopify’s core functionality by building apps that integrate into Shopify’s admin, online store, checkout and more  
  
Graphical user interface, text, application

Description automatically generated  
  
Ref: <https://shopify.dev/docs/api>

**Webhook:**

We can use webhook subscriptions to receive notifications about particular events in a shop. After we've subscribed to a webhook topic, our app can execute code immediately after specific events occur in shops that use our installed app, instead of having to make API calls periodically to check their status.

So webhooks only works when we update anything’s in Shopify Store.

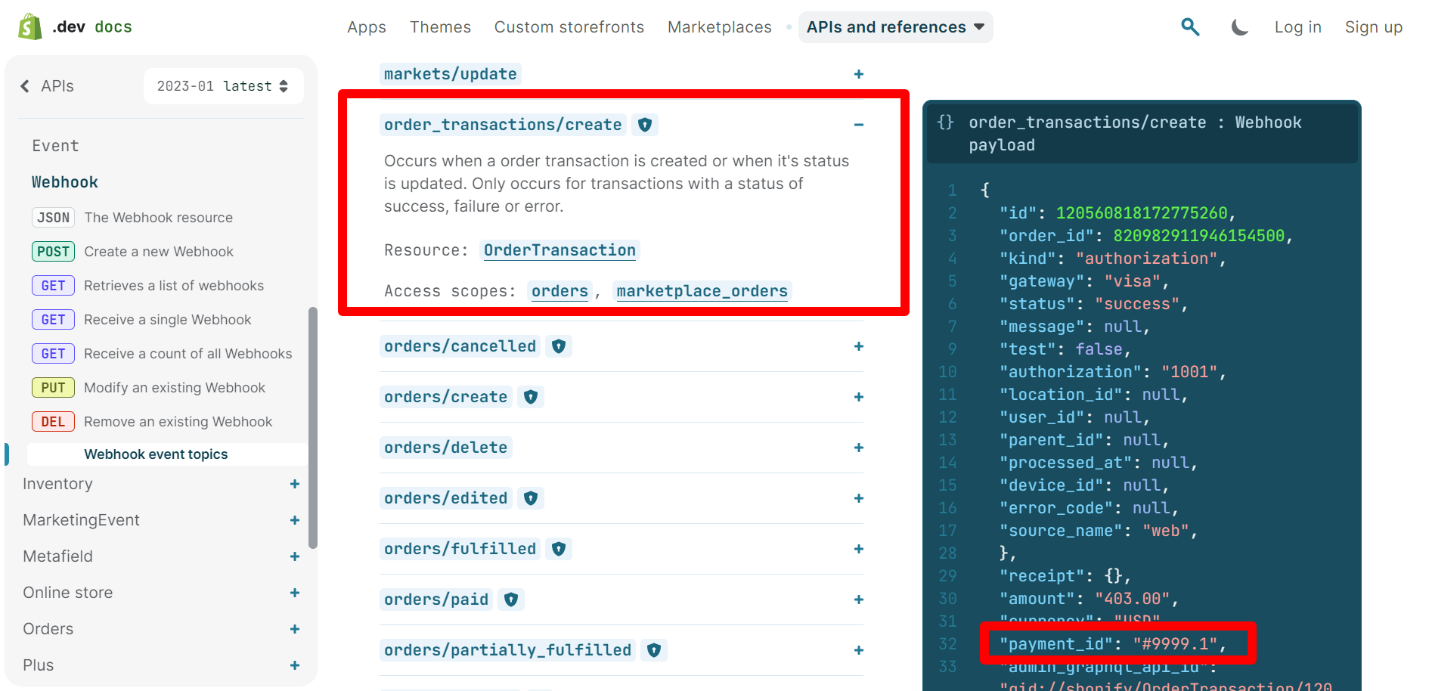
Graphical user interface, text

Description automatically generated with medium confidence

Ref: <https://shopify.dev/docs/api/admin-rest/2023-01/resources/webhook#event-topics>

**Workflow: Get Order ID**

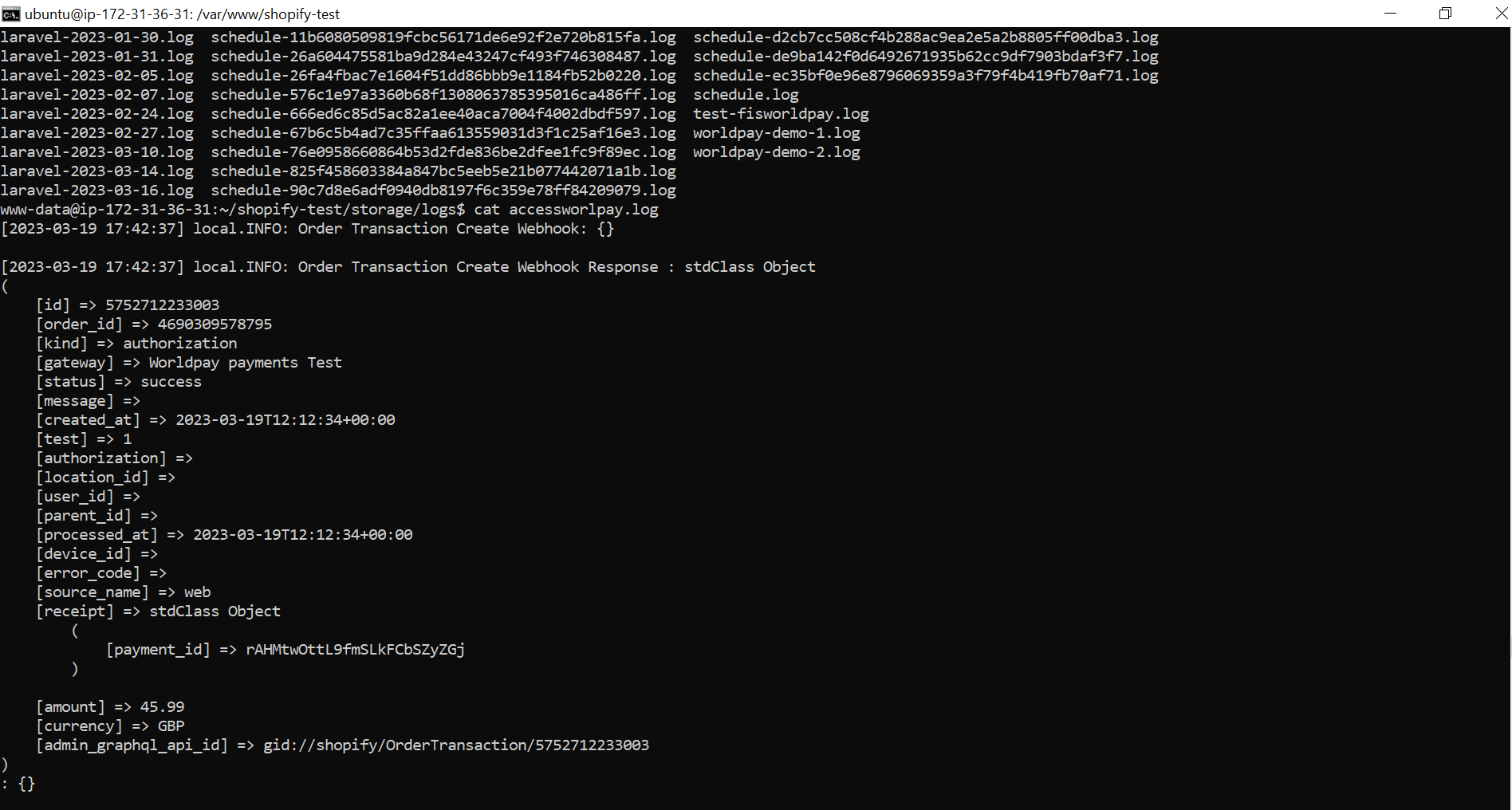
If we click on the Pay Now button in Shopify, the payment flow begins with an HTTP POST request sent from Shopify to the payment session URL of our offsite payments’ app extension provided during app extension configuration. This request contains information about the customer and the order. Order ID will not be sent from Shopify.  
  
Ref: <https://shopify.dev/docs/apps/payments/implementation/process-an-offsite-payment#initiate-the-flow> Graphical user interface, text, application

Description automatically generated  
Without order id we can’t update order status in Shopify through API. In order to get order id, we are using webhook (order\_transactions/create)  
  


This webhook gets triggered when an order transaction is created or when its status is updated.  
  
When order transaction is created this webhook gets triggered and provides as with Payment Id and Order Id.

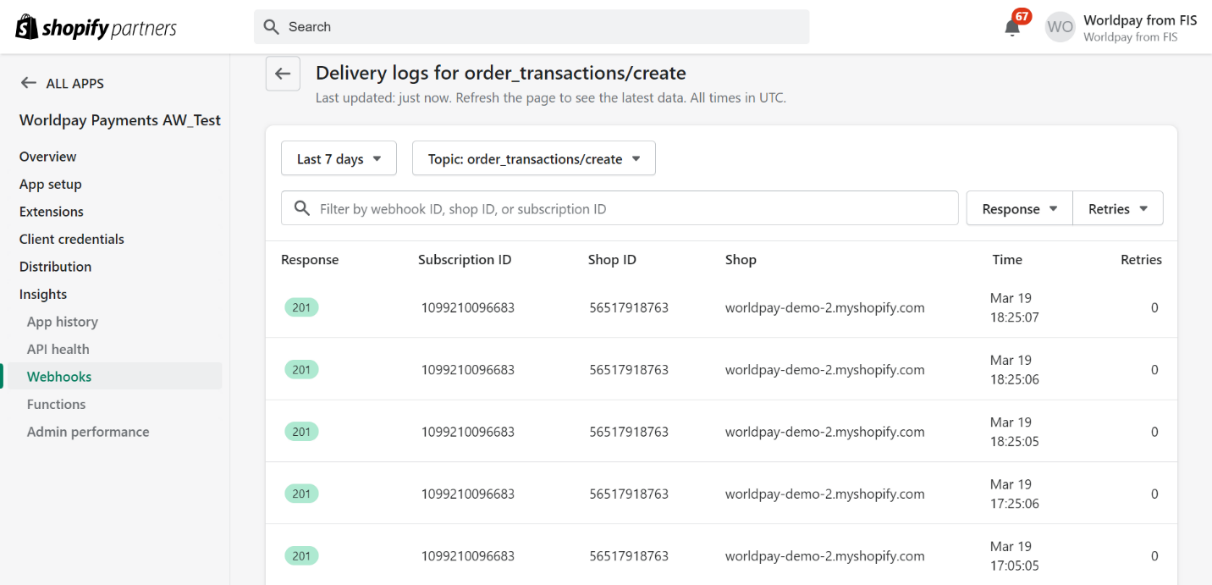
We map the Payment Id obtained from the Webhook with Payment Id in payment session request to get the Order Id that was not provided by Shopify in Payment session request.  
  
Ref: <https://shopify.dev/docs/api/admin-rest/2023-01/resources/webhook#event-topics-order-transactions-create>

**Webhook Response:**

  
  
In our middleware we should insert below webhook code in config/shopify\_app.php  
  
[  
 'topic' => 'ORDER\_TRANSACTIONS\_CREATE',

'address' => env('APP\_URL') . 'webhook/order-transactions-create'

]  
  
To see the webhook history in Partner Dashboard,



**To achieve this webhook functionality we need read\_orders access scope.**

**Workflow: Update Status to Shopify**

Now we have obtained the Order Id. Using this order id we can access order cancel, capture and refund functionality from middleware like as we perform it in Shopify.  
  
We can use Transaction Admin API and pass the transaction id and Order Id to Shopify and perform the capture, cancel and refund functionality. Once this is triggered, we can get the GID and later we can sync order status of MAI with Shopify.

Graphical user interface, text, application

Description automatically generated  
  
Ref : <https://shopify.dev/docs/api/admin-rest/2023-01/resources/transaction#post-orders-order-id-transactions>

Find the Postman script below,  
 **Capture postman script:**  
Graphical user interface, text, application, email

Description automatically generated  
  
**Void postman script**

Graphical user interface, text, application, email

Description automatically generated

**Refund Postman script**

**Graphical user interface, text, application, email

Description automatically generated**

**Partial Refund postman script**

**Graphical user interface, text, application, email

Description automatically generated**

We get the below Response from Shopify after capture event triggered,  
  
[2023-03-20T22:15:54.960448+05:30] Local Info.INFO: Shopify Capturing an authorized payment Request: : {"id":"ogsFgu95ZYg6m4DJRAI\_iol1","gid":"gid:\/\/shopify\/CaptureSession\/ogsFgu95ZYg6m4DJRAI\_iol1","payment\_id":"rI4YrBLZ1rxt5J6DyapeJDuUc","amount":"45.99","currency":"GBP","merchant\_locale":"en","proposed\_at":"2023-03-20T16:45:54Z","test":true}  
  
**Note** : If we proceed with above solution Partial capture is possible. At present we don’t have this functionality. We can work on it.  
  
**To achieve this admin API functionality we need write\_orders access scope.**  
**Conclusion:**

We can achieve this milestone only if read\_orders and write\_orders permission is provided by Shopify.  
  
If Permission is given then our app install screen will look like,Graphical user interface, text, application, email

Description automatically generated