Pay with Shop Your Way (PYW) – Implementation Guide

Contents

Introduction **2**

Installation **2**

1. Tag Javascript 2
2. ReactJS 5
3. Android Native SDK 6
4. iOS Native SDK 8
5. Flutter+iOS SDK 11
6. Vanilla Java-Script 12
7. Contactless 15
8. Web 16

Fields & Attributes Definition **16**

Fields & Attributes Definition (Tag Javascript) **19**

API Specifications **22**

1. Contactless Payment API 22
2. Generate Payment Id API 24
3. Payment Confirmation API 26
4. Return Payment API 29
5. Subscription renewal payment API 30
6. Update Order API (Applicable for TYPE-A integration) 32

Appendix **34**

Introduction

Pay with Shop Your Way (PYW) provides a payment facility for any online purchase transaction. It provides an instant credit card apply & use facility along with using various other credit cards. PYW also provides payment facility using the *Shop Your Way (SYW)* points.

Pay with Shop Your Way supports below implementation methods.

1. Direct integration with merchants' online checkout using Pay with SYW SDK (TYPE-A Integration)
2. Contactless method using API integration (TYPE-B Integration)
3. URL based redirection (TYPE-C Integration)

1. TYPE-A Integration: This is the default Pay with Shop Your Way Implementation. In this case, merchants will add the Pay with SYW sdk to their checkout. The sdk will sit on the merchant's payment page and will show 'Pay with Shop Your Way' as one of the payment options to customers as part of their online checkout.

2. TYPE-B Integration: This type of integration is used for in-store purchase or associate assisted payment scenarios where the merchant systems do not give direct access to customers to use Pay with Shop Your Way. In this case, the merchant's system will trigger the Pay with SYW contactless API to request payments from customers. Pay with SYW will then create a unique payment link and send it to the customer via text and/or email. Customers will click this link to access Pay with SYW and complete the payment.

3. TYPE-C Integration: This is URL based Pay with Shop Your Way Implementation. In this approach, merchants will generate a unique payment ID which will be then passed as a query parameter in Pay with Shop Your Way (PYW) URL to open the PYW module.

Installation

Pay with Shop Your Way (PYW) can be implemented in your web applications in following ways.

1. Tag Javascript

Use the following method to integrate the **<pyw-app>** tag by importing CSS and JS files directly into your HTML page. Pass props within the **<pyw-app>** tag.

Advantages of using this tag -

- No external dependencies are required.
- Compatible with any web framework e.g. Angular, ReactJS, vanilla JS, etc.
- Minimal setup is necessary.
- Usable as a standalone web component.
- Custom elements can be used directly in HTML with their properties passed as attributes or properties.

Linking to CC and JS

Copy the following script and css link and paste it in the HTML file.

Non production

```
JavaScript

<link rel="stylesheet"
href="https://pywweb.uat.telluride.shopyourway.com/public/pyw_library/styles/pyw_sdk_lib.min.css">

<script
src="https://pywweb.uat.telluride.shopyourway.com/public/pyw_library/scripts/pyw_sdk_lib.min.js" defer></script>
```

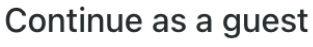
Production

```
JavaScript

<link rel="stylesheet"
href="https://pywweb.telluride.shopyourway.com/public/pyw_library/styles/pyw_sdk_lib.min.css">
<script
src="https://pywweb.telluride.shopyourway.com/public/pyw_library/scripts/pyw_sdk_lib.min.js" defer></script>
```

This is how button will be shown which will invoke PYW:

This is how link will be shown which will invoke PYW:



Sample Code

Below is sample Html code snippet **(for your reference)** showing a Button/Link option, once selected it renders the PYW payment component

```
JavaScript
<pyw-app
```

```
 version="1"
 refid={refId}
 merchant={merchantType}
 total-due={totalDue}
 return-url={returnUrl}
 pay-type={"subscription-<id>"}
 profile="qa"
 open-with={opentype}
 button='{
           "name": "Continue to payment"
         }'
 link='{
          "name": "Continue as a guest user"
       }'
 disable="false"
 items='[{
            "itemId": "1001",
            "itemName": "Amazon Voucher",
            "itemDesc": "Amazon Voucher value of 1000",
            "imageUrl": "",
            "price": "4"
       }]'
addresses= '[{
               "type": "S",
               "line1": "1250 Waters Pl",
               "line2": "",
               "city": "Bronx",
               "state": "NY",
               "zip": "10461"
             },
             {
               "type": "B",
               "line1": "1250 Waters Pl",
               "line2": "",
               "city": "Bronx",
               "state": "NY",
               "zip": "10461"
           }]',
 params='{
             "usertype": "R",
             "useremail": "test@test.com"
           }'
>
</pyw-app>
```

Note: Please refer to the *Fields & Attributes Definition (Tag Javascript)* section for more details.

2. ReactJS

The **telluride-pyw** React component can be installed in your application using the below `npm` command,

```
npm install --save <file_path>/telluride-pyw-2.1.1.tgz
```

Note:
`<file_path>` (optional), if **telluride-pyw-2.1.1.tgz** file is present in the same location where your application's `package.json` file is kept then you can directly mention the file name and for different location you need to provide a fully qualified path.

Import **telluride-pyw** component in the appropriate component of your React application.

```
import { Pyw, Refid } from 'telluride-pyw';
import { Items, Item, Addresses, Address } from 'telluride-pyw';
import { updateTotalDue, updateReturnURL, updateDisablePay } from 'telluride-pyw';
```

For rendering the PYW UI in your application, add the below code snippet at the appropriate level in your code. Please refer to the *Fields & Attributes Definition* section for more details.

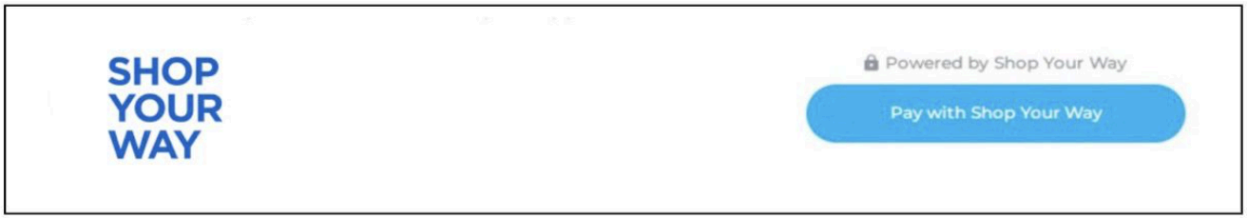
```
<div id="pywdiv">
    <Pyw merchant="<client_id attribute received during onboarding process e.g.
EXT_CLIENT_NAME>"
            totalDue="<total due amount to be paid>"
            returnUrl="<callback url to redirect the control after payment is complete>"
            openWith="<_popup | _self>"
            profile="<select the value as uat|prod depending upon the environment>"
            payType="<text value to distinguish payments for various transaction types>"
            disable="<true|false>" >

        <Refid key="k101" value="<encrypted value of
client_id~~access_token~~requestorId~~timestamp~~transactionId~~userId>"/>

        <Items key="k102">
          <Item id="10448" name="Test1" imageUrl="/a/b/c.jpg" qty="2" price="2" unit="usd"/>
          <Item id="20448" name="Test2" imageUrl="/a/b/d.jpg" qty="1" price="5" unit="usd"/>
        </Items>

        <Addresses key="k103">
          <Address type="S" line1="1 Waters Pl" line2="" city="Bronx" state="NY"
zip="10461"/>
          <Address type="B" line1="1 Waters Pl" line2="" city="Bronx" state="NY"
zip="10461"/>
        </Addresses>
    </Pyw>
</div>
```

Once the PYW component is added into your application it looks like below,



Note: This is a sample banner content and layout. Both, content and layout can be customized based on your design needs.

3. Android Native SDK

Pay with Shop Your Way (PYW) can be implemented in your native applications in following ways.

a. Copy PywSdk-Release.aar in libs folder of your Android Project
b. Add the below lines to gradle dependencies to access the aar library

```
dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation files('libs/PywSDK-release.aar')
}
```

c. In the xml where you want to show the PywWidget add the following code in the xml

```
<com.pyw.widget.PywWidget
    android:id="@+id/pywWidget"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

d. Link the xml widget in the activity

```
PywWidget pywWidget = findViewById(R.id.pywWidget);
```

e. Create the PywConfig object and set config to PywWidget Instance like shown below

```
PywConfig config = new PywConfigBuilder()
        .setContext(this)
        .setTotalDue("<totalDue - mandatory field*>")
        .setRefId("<refId - mandatory field*>")
        .setMerchantId("<merchantId - mandatory field*>")
        .setAddresses("<addresses - optional field*>")
        .setItems("<items - optional field*>")
        .setPayType("<payType - optional field*>")
        .setOpenWith("<openWith - optional field*>")
        .setTransactionId("<transactionId - mandatory field*>")
        .setMode(PywMode.UAT / PywMode.PROD) - optional field default value is PROD
        .setPrepareButtonEnabled("boolean default is true - optional field")
        .setResponseListener("<PywResponseListener - mandatory field*>"))
        .setPrepareListener("<PywPreparePaymentListener - optional field*>")
        .setStatusListener("<PywStatusListener - optional field*>")
        .build();

pywWidget.setConfig(config);
```

f. Create the PywResponseListener object and set it to config Instance like shown below

```
.setResponseListener(new PywResponseListener() {
  @Override
  public void onPywSuccess(String pywId, PywConfirmation pywConfirmation) {
    //ToDo handle success scenario
  }
  @Override
  public void onPywHold(String pywId) {
    //ToDo handle hold scenario
  }
  @Override
  public void onPywFailure(String pywId) {
    //ToDo handle failure scenario
  }
  @Override
  public void onPywCanceled(String pywId) {
    //ToDo handle cancelled scenario
  }
  @Override
  public void onPywInvalidSession() {
    //ToDo handle invalid session scenario
  }
})
```

Note: Please refer to the *Fields & Attributes Definition* section for more details.

g. If incase you want to update the totalDue amount call the below function

```
pywWidget.updateDueAmount("<totalDue>");
```

h. If incase you want to update the refId call the below function

```
pywWidget.updateRefId("<refId>")
```

i. If incase you want to enable or disable pyw button use the below function

```
pywWidget.setPrepareButtonEnabled(true/false)
```

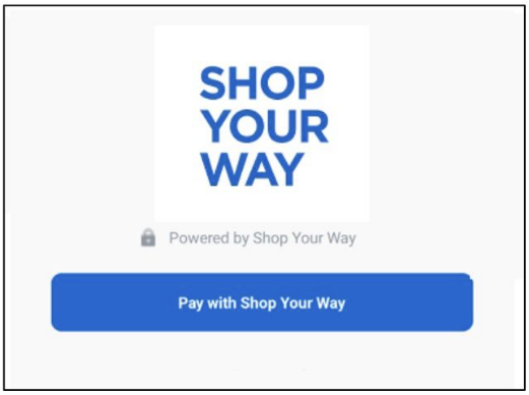
j. If in case you want to do validations before making the final payment flow, then you need to add the setPrepareListener in the config and calling the preparePayment() is mandatory in this scenario after the validations. If you don't want any validations you don't have to add this listener

```
.setPrepareListener(new PywPreparePaymentListener() {
  @Override
  public void onPywPreparePayment() {
    //Todo any validations if required
    pywWidget.preparePayment();
  }
})
```

k. In case you want to call PYW Confirmation APIs yourself, you need to set this listener in the config. Note that since the confirm api if called from your end we will not return the PywConfirmation Object in the on success. If you don't call the confirm api here then the status will go to onHold by default

```
.setStatusListener(new PywStatusListener() {
    @Override
    public void getStatus(String pywId) {
        // make confirm api call
    }
})
```

Once the PywWidget component is added into your application and the config is set properly it looks like below,

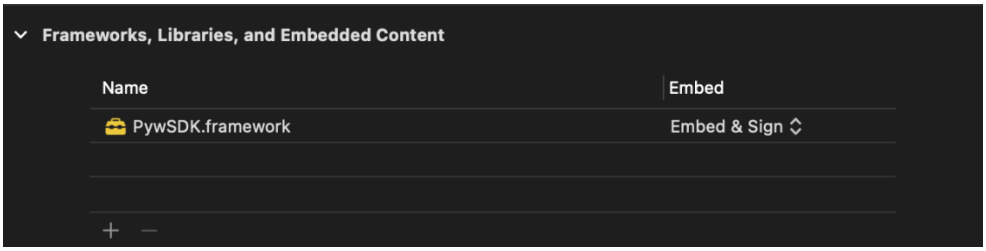


Note: This is a sample banner content and layout. Both, content and layout can be customized based on your design needs.

4. iOS Native SDK

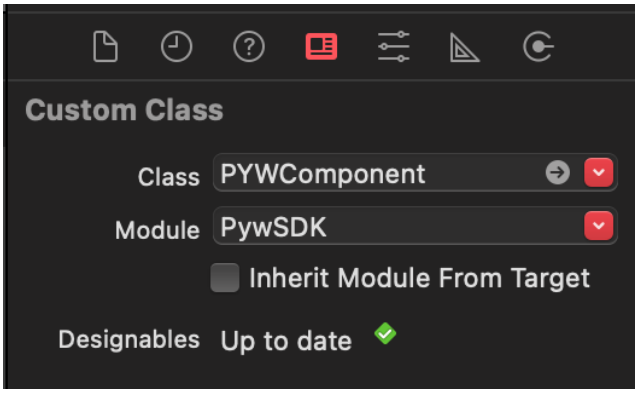
Pay with Shop Your Way (PYW) can be added to your native application as follows.

Copy PywSDK.xcframework to your project's folder via project navigator.



Note: Using custom PYWComponent view is available via storyboards / xibs and from code.
For now, recommended view height is >= 250 px.

From storyboards / xibs:
Add a regular UIView and set its custom class to PYWComponent.



From code / programmatically:

```
let component = PYWComponent()
component.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(component)

// Set frame via constraints or CGRect


//
```

Once the UI part is done, we need to create the PywConfig object and set config to PywComponent instance like shown below.

By default, PywConfig has some mandatory fields and optional fields available via builder.

```
let config = PywBuilder(
            refId: "<refId>",
            transactionId: "<transactionId>",
            environment: <.uat or .prod>,
            delegate: <self or any other object>)
            .setItems("<items>")
            .setAddresses("<addresses>")
            .setPayType("<payType>")
            .setOpenWith("<openWith>")
            .setTotalDue("<totalDue>")
            .setReturnUrl("<returnUrl>")
            .setMerchantId("<merchantId>")
            .setSourceViewController(<self or any other view controller> )
            .setPrepareHandler { <called when action button is pressed> }
            .setStatusHandler({ pywId in <called when the payment status is changed to “success”>
})
            .setEnabledPrepareButton(true / false) // if prepare button is enabled or not
            .build()
 component.setConfig(config)
```

In order to retrieve data back from the SDK using `PywCheckoutDelegate` will be handy.

```
extension ViewController: PywCheckoutDelegate {
    func onPywSuccess(pywId: String, pywConfirmation: PYWConfirmation) {
        // handle onSuccess scenario
    }

    func onPywFailure(pywId: String?) {
        // handle onFailure scenario
    }

    func onPywCanceled() {
        // handle onCanceled user-initiated scenario
    }

    func onPywHold(pywId: String) {
        // handle onHold scenario
    }

    func onPywInvalidSession() {
        // handle onInvalidSession scenario
    }
}
```

In case you want to update the totalDue amount, after the configuration is set, call the function below

```
component.updateDueAmount("<totalDue>")
```

In case you want to update the refId, after the configuration is set, call the function below

```
component.updateRefId("<refId>")
```

In case you want to change prepare's button enable state:

```
component.setPrepareButtonEnabled(true)
```

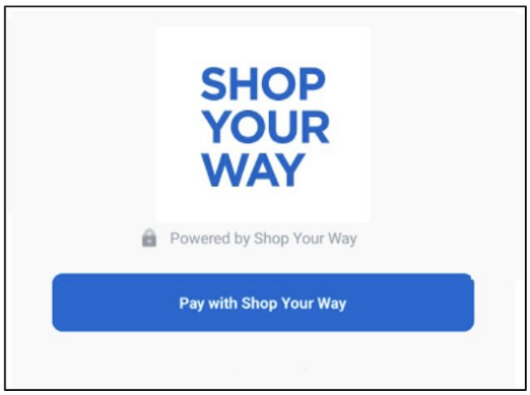
In case you want to do some extra validations before calling the SDK, you can use this PYWBuilder's item

```
.setPrepareHandler {
        // do all required validations
        // after finished to proceed with the SDK you need to call
       self.component.preparePayment()
}
```

In case you want to call PYW Confirmation APIs yourself, you need to use this item from PYWBuilder

```
.setStatusHandler({ pywId in
        // do any additional setup or call any APIs
})
```

Once the PywComponent view is added into your application and the config is set properly it should look like below:



Note: This is a sample banner content and layout. Both, content and layout can be customized based on your design needs.

5. Flutter+iOS SDK

a. Extract the zip file `FlutterPlugin_pyw_sdk.zip`, and copy that folder into your project.

b. **plugin_pyw_sdk** can be added to your flutter applications in the following way. Add the plugin_pyw_sdk in `pubspec.yaml` file inside the dependencies section.

```
dependencies:
  flutter:
    sdk: flutter
  plugin_pyw_sdk:
    path: ./plugin_pyw_sdk
```

c. Once the dependency is added, import this plugin where you want to implement the PYW payment.

```
import 'package:plugin_pyw_sdk/Models/pyw_payment_model.dart';
import 'package:plugin_pyw_sdk/plugin_pyw_sdk.dart';
```

d. Create an instance of **PYWPaymentModel** and **PluginPywSdk**.

```
final PluginPywSdk _controller = PluginPywSdk();
PYWPaymentModel paymentModel = PYWPaymentModel();
```

e. After creating the instance, just implement the UI of **PluginPywSdk** in your screen. Make sure it's height should be 250 px or more.

```
child: Container(
  child: _controller,
), // Container
```

f. Build the `refId` (refer *Fields & Attributes Definition* section for more details) and set up all the parameters which are required for initiating the payment using PYW.

```
paymentModel.isProduction = false; // for .uat false, for .prod true
paymentModel.refId = "<refId>";
paymentModel.transactionId = "<transactionId>";
paymentModel.total = "<total>";
paymentModel.merchantId = "<merchantId>";
paymentModel.items = "<items>";
paymentModel.payType = "<payType>";
paymentModel.address = "<address>";
paymentModel.openWith = "<openWith>";
paymentModel.returnUrl = "<returnUrl>";
```

g. After setting up all the configurations, load payment UI. Here, you will get all the native delegate method's response with its status with a hashable dictionary (Map<String,dynamic>) object.

```
_controller.setPaymentData(paymentModel.toJson(), (getData) {
  //
  if (kDebugMode) {
    print("Futter getData: $getData");
  }
});
```

6. Vanilla Java-Script

Implement the PYW using plain Java-script with the following steps:

a. Below is a sample Html code snippet **(for your reference)** showing a "Pay with Shop Your Way" option, once selected it renders the PYW payment component. Here the `loadPyw()` function is called with onchange event on the radio input element. You may call the same function on any other javascript event.

Check the **TODOs** marked in the below Html code and add the underlying tags into your Html.

```
<div>
  <input type="hidden" id="disablePywFlag" value="<true|false|hide value to show or disable or hide the PYW component>" />
   <input type="hidden" id="merchant" value="<client_id value used in building refid e.g. EXT_CLIENT_QA>" />

   <div>
      <div>
```

```
         <p style="color: #5F9EA0; font-family: Arial; font-size: 150%;">Select Payment
Type:</p>
      </div>
      <div style="color: #6495ED; font-family: Arial; font-size: 100%;">
         <input type="radio" id="cc" name="paymentType" value="cc" onchange="alert('Credit
Card Payment Type is selected')">
         <label for="pyw"><strong>Credit Card</strong></label>
      </div>
      <br>
      <div style="color: #6495ED; font-family: Arial; font-size: 100%;">

<!-- TODO: You may call the below loadPyw() function on any javascript event -->
         <input type="radio" id="pyw" name="paymentType" value="pyw"
                  onchange="loadPyw(document.getElementById('disablePywFlag').value,
document.getElementById('merchant').value)">

         <label for="pyw"><strong>Pay with Shop Your Way</strong></label>
       </div>
   </div>

   <div>

<!-- TODO: add the below <iframe> tag to the location where PYW component needs to be
displayed on your html page -->
       <iframe id="pywiframe" src="" width="70%" height="100%" frameborder="0"
allowtransparency="yes" scrolling="no" title="Pay with Shop Your Way"></iframe>

   </div>

<!-- TODO: add the below complete <div> tag to disable the background of the parent page after
opening the PYW in new popup window -->
   <div id="pywpopup" class="pywpopup-sdk">
       <div class="pywpopup-sdk-box">
         <section class="pywpopup-sdk-sec">
           <p class="pywpopup-sdk-text">
               Pay with Shop Your Way in progress...
           </p>
         </section>
       </div>
   </div>

<!-- TODO: add the below complete <div> tag to show a modal window with error message in case
the PYW session is invalid -->
   <div id="pywerrorpay" class="pywerrorpay-sdk">
       <div class="pywerrorpay-sdk-box">
         <section class="pywerrorpay-sdk-sec">
           <p class="pywerrorpay-sdk-text crimson">
               Invalid Payment Details or Invalid Session.
           </p>
           <div class="pywflex pywcenter">
               <button class="pywbtn pywbtn-primary" onclick="closeErrorPopup(this.value)">
                   Close
               </button>
           </div>
         </section>
       </div>
   </div>

<!-- TODO: add the below complete <div> tag to show a modal window with a message in case the
popups are blocked in the browser -->
   <div id="pywpopupblocked" class="pywerrorpay-sdk">
```

```
      <div class="pywerrorpay-sdk-box">
         <section class="pywerrorpay-sdk-sec">
            <p class="pywerrorpay-sdk-text amber">
               Window popups may be blocked in your browser, please allow popups and retry.
            </p>
            <div class="pywflex pywcenter">
               <button class="pywbtn pywbtn-primary" onclick="closePopupBlockedMessage()">
                  Ok
               </button>
            </div>
         </section>
       </div>
    </div>

</div>
```

b. Once the above Html changes are done, add the below javascript function "**pywPrecheckFetchPrepareCheckout()**" in one of your JS files which is included in the same Html file where PYW `loadPyw()` function is called.

```
function pywPrecheckFetchPrepareCheckout() {
        // derive the refid with the refid generation logic and assign it to the
        below variable.
        var refId =
        'q8apGToe7%2FzcEmP5nHfJcSLU85Sy0O8b51wnU6yV4z89kq5lhuNLc7Acbda2IEx7uK%2F8
        4tDyIwZy9XEBvHWLtmwhfEpMRuvTOatu3sZRyxrKrJ%2FxmSpy1siErxS6QVYmJ%2FyPTRV%2
        BMn5sqHf72A4IxPhkxFUjZEDThmoWh7STSRU%3D';

        // set the value of total due to be paid using PYW
        var totalDue = '0.3';

        // set the value of return url where the control is to be redirected
        after PYW payment or when the PYW popup window is closed
        var returnURL = 'https://<client-domain>/payment/confirmation';

        // set the value of client id which is registered with PYW oAuth system
        var merchantId = 'EXT_CLIENT_QA';

        // set the value as '_popup' or '_self' in case PYW app needs to be
        opened in a new popup window or in the same window of the client's app
        var openwith = '_popup';

        // set this in case you want to send checkout item details to the PYW
        app. refer Fields & Attributes Definition section for more details.
        var items = '';

        // set this in case you want to send shipping & billing address details
        to the PYW app. refer Fields & Attributes Definition section for more
        details.
        var addresses = '';

        // set this value as the name of the business entity for which the
        payment is to be done using PYW
        var payType = 'insurance';

        // call the processPayment() function passing all above derived
        variables.
```

```
            processPayment(refId, totalDue, returnURL, merchantId, openwith, items,
            addresses, payType);
        }
```

c. **`Link CSS:`** please add below *css* file in your html to render the PYW UI.
*(below link is for **UAT** environment)*

```
<link href="https://pywweb.uat.telluride.shopyourway.com/pyw_library/styles/pywexternal.css"
rel="stylesheet" />
```

For production – change CSS href to,
https://pywweb.telluride.shopyourway.com/pyw_library/styles/pywexternal.css

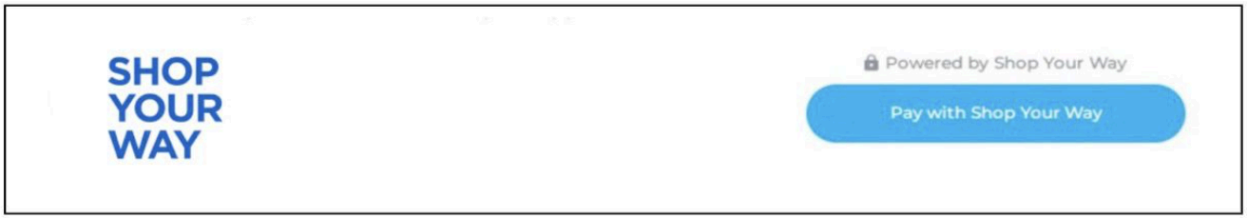
d. **`Add script tags:`** please add below *javascript* in your html for the PYW functioning.
*(below link is for **UAT** environment)*

```
<script
src="https://pywweb.uat.telluride.shopyourway.com/pyw_library/scripts/pywexternal.js"></script
>

<script
src="https://pywweb.uat.telluride.shopyourway.com/pyw_library/scripts/pywscript.js"></script>
```

For production – change above scripts src to,
https://pywweb.telluride.shopyourway.com/pyw_library/scripts/pywexternal.js
https://pywweb.telluride.shopyourway.com/pyw_library/scripts/pywscript.js

Once the PYW component is added into your application it looks like below,



Note: This is a sample banner content and layout. Both, content and layout can be customized based on your design needs.

7. Contactless

Use the PYW for contactless payment by calling the **Contactless Payment API**. This option can be used to initiate a PYW payment for a contactless transaction such as any payment triggered via Point of Sale (POS) or any other offline system w.r.t. the end user. You can refer to the API specs of the **Contactless Payment API** here.

8. Web

Use this method by calling the **Generate Payment Id API** and pass the payment Id over Pay with Shop your Way (PYW) URL to initiate the PYW module. This option can be used to initiate the PYW module on click of a button. You can refer to the API specs of the **Generate Payment Id API** here.

PYW URL:

UAT:
https://pywweb.uat.telluride.shopyourway.com/pmt?id=<**paymentId>**&merchant=<**merchantClientId**>

PROD:
https://pywweb.telluride.shopyourway.com/pmt?id=<paymentId>&merchant=<**merchantClientId**>

Fields & Attributes Definition

`Pyw` – root element for rendering the PYW UI

`merchant` – (mandatory) attribute of `<Pyw>` element. Value of **`client_id`** attribute assigned to each Client during the onboarding process and the same is used in building the **`refid`**.

`totalDue` – (conditional) attribute of `<Pyw>` element. Total amount to be paid in USD. If **`<Items>`** element is present then **`totalDue`** can be ignored.

`returnUrl` – (optional) attribute of `<Pyw>` element. Specify a callback url to redirect the control after payment is complete. It can be a reference path w.r.t. existing client domain (e.g. `/checkout/paymentsummary`) or a complete http(s) url (e.g. `https://www.client-domain.com/checkout/paymentsummary`).

`openWith` – (optional) attribute of `<Pyw>` element with values as `_popup` (default) or `_self`.

`profile` – (optional) attribute of `<Pyw>` element with values as `uat or prod` (default).

`payType` – (optional) attribute of `<Pyw>` element with value to distinguish payments to be used for different types of transactions.

`disable` – (optional) attribute of `<Pyw>` element with possible values as `true` or `false`. Set it to `true` if you want to disable the “Pay with Shop Your Way” button otherwise set it as `false` to make it active again.

Items – (optional) child element for <Pyw> element. To be added if items need to be shared with PYW to avail any item specific SYW offers or for displaying the item details in the PYW screens.

Set key attribute (mandatory) with value as "k102".

Item – (mandatory) child element for <Items> element. Capture item details being purchased.

id – (mandatory) attribute of <Item> element. Id of item being purchased.

name – (mandatory) attribute of <Item> element. Name of item being purchased.

imageUrl – (optional) attribute of <Item> element. Logo/image of item being purchased.

qty – (optional) attribute of <Item> element, default "1". Quantity of item being purchased.

price – (mandatory) attribute of <Item> element. Unit price of item being purchased.

unit – (optional) attribute of <Item> element, default is "usd". Currency unit for item price.

Addresses – (optional) child element for <Pyw> element. To be added if shipping or billing address needs to be shared with PYW. This helps avoid payment fraud.

Set key attribute (mandatory) with value as "k103".

Address – (mandatory) child element for <Addresses> element. Capture shipping or billing address details for the current payment.

type – (optional) attribute of <Address> element. S – for Shipping & B – for Billing, if missing value then the address can be considered as same for Shipping & Billing.

line1 – (mandatory) attribute of <Address> element. Address line 1.

line2 – (optional) attribute of <Address> element. Address line 2.

city – (mandatory) attribute of <Address> element.

state – (mandatory) attribute of <Address> element.

zip – (mandatory) attribute of <Address> element.

Refid – (mandatory) child element for <Pyw> element. It is the encrypted value of the following concatenated attributes separated by a delimiter as **~~.** Set key attribute (mandatory) with value as "k101".

client_id~~access_token~~requestorId~~timestamp~~transactionId~~userId

Example:

EXT_CLIENTNAME_QA~~d625c7e969e6996c76a1c4909a7de8a3bad7b0d4d8b24371280f3b429218972c~~RQID~~1648154523783~~15207454~~123456

Where,

e. `client_id` – assigned to each Client during the onboarding process.
f. `access_token` – computed using a `secret_key` assigned to each Client during the onboarding process *(valid for 2 hours after it has been created).*
g. `requestorId` – assigned to each Client during the onboarding process.
h. `timestamp` – long value (epoch) of the current timestamp in milliseconds.
i. `transactionId` – unique cart id or transaction id or order id maintained by the client
j. `userId` – user's unique id such as user's login id or email address

Note:

- Use **AES algorithm** to build the encrypted **`refid`** value
- Refer *Appendix* to follow the **Client Onboarding process** to get the **`client_id`** & **`secret_key`**.
- Refer *Appendix* to get & use the encryption key required to generate the **`refid`**.

`updateTotalDue` – (optional) function to be called on any element or from another reactJS function which changes the total due amount to be paid. This function updates the value of the total due to be paid using <pyw> component.

e.g. If there is a text box which is used to change the value of the total due amount then this function ***`updateTotalDue(<updated total due amount>)`*** can be called on this text box element as below,

```
<input name="totalDue" value={totalDue} onChange={(e) => updateTotalDue(<updated total due amount>)} type="text" />
```

`updateReturnURL` – (optional) react function to be called on any element or from another reactJS function which changes the Return URL to be redirected once return from pywweb popup. This function updates the value of the return URL that is used for redirection <pyw> component.

e.g. If there is a text box which is used to update the value of the return URL then this function ***`updateReturnURL(<updated Return URL>)`*** can be called on this text box element as below,

```
<input name="returnURL" value={returnUrl} onChange={(e) => updateReturnURL(<updated Return URL>)} type="text" />
```

`updateDisablePay` – (optional) react function to be called on any element or from another reactJS function which disables or enables the **Pay with Shop Your Way** button on the initial page.

e.g. **`updateDisablePay(<updated disable flag>)`** function can be called on selection/deselection of any checkbox.

```
<input name="disableFlag" value={disableFlag} onChange={(e) =>
updateDisablePay(<updated disable flag>)} type="text" />
```

Fields & Attributes Definition (Tag Javascript)

`pyw-app` – root element for rendering the PYW UI

`merchant` – (mandatory) props of `<pyw-app>` element. Value of `client_id` attribute assigned to each Client during the onboarding process and the same is used in building the `refid`.

`total-due` – (conditional) props of `<pyw-app>` element. Total amount to be paid in USD. If **`Items`** props are present then **`totalDue`** can be ignored.

`return-url` – (optional) props of `<pyw-app>` element. Specify a callback url to redirect the control after payment is complete. It can be a reference path w.r.t. existing client domain (e.g. `/checkout/paymentsummary`) or a complete http(s) url (e.g. `https://www.client-domain.com/checkout/paymentsummary`).

`open-with` – (optional) props of `<pyw-app>` element with values as `_popup` (default) or `_self`.

`profile` – (optional) props of `<pyw-app>` element with values as `uat or prod` (default).

`pay-type` – (optional) props of `<pyw-app>` element with value to distinguish payments to be used for different types of transactions.

E.g. For subscription type transaction , we need to pass the value as **subscription-**<subscription number>

Note: Same subscription number should be used while calling subscription renewal payment ßAPI.

`disable` – (optional) props of `<pyw-app>` element with possible values as `true` or `false`. Set it to `true` if you want to disable the payment button/link otherwise set it as `false` to make it active again.

`items` – (optional) props (as a JSON array) for `<pyw-app>` element. To be added if items need to be shared with PYW to avail any item specific SYW offers or for displaying the item details on the PYW screens.

`id` – (mandatory) Id of item being purchased.

`name` – (mandatory) Name of item being purchased.

`imageUrl` – (optional) Logo/image url of item being purchased.

`qty` – (optional) default `"1"`. Quantity of item being purchased.

`price` – (mandatory) Unit price of item being purchased.

`unit` – (optional) default is `"usd"`. Currency unit for item price.

JavaScript

```
[
    {
        "itemId": "1001",
        "itemName": "Amazon Voucher",
        "itemDesc": "Amazon Voucher value of 1000",
        "imageUrl": "",
        "price": "4"
    }
]
```

addresses – (optional) props (as a JSON array) for `<pyw-app>` element. To be added if shipping or billing address needs to be shared with PYW.

type – (optional) value of `Item` property. S – for Shipping & B – for Billing, if missing value then the address will be considered same for Shipping and Billing.

line1 – (mandatory) attribute Address line 1.

line2 – (optional) attribute .Address line 2.

city – (mandatory) attributet.

state – (mandatory) attribute.

zip – (mandatory) attribute

JavaScript

```
  addresses='[{
                "type": "S",
                "line1": "1250 Waters Pl",
                "line2": "",
                "city": "Bronx",
                "state": "NY",
                "zip": "10461"
            }]'
```

refid – (mandatory) props for `<pyw-app>` element. It is the encrypted value of the following concatenated attributes separated by a delimiter as **~~.**

`client_id~~access_token~~requestorId~~timestamp~~transactionId~~userId`

`Example:`

`EXT_CLIENTNAME_QA~~d625c7e969e6996c76a1c4909a7de8a3bad7b0d4d8b24371280f3b429218972c~~RQID~~1648154523783~~15207454~~123456`

Where,

a. `client_id` – assigned to each Client during the onboarding process.
b. `access_token` – computed using a `secret_key` assigned to each Client during the onboarding process *(valid for 2 hours after it has been created).*
c. `requestorId` – assigned to each Client during the onboarding process.
d. `timestamp` – long value (epoch) of the current timestamp in milliseconds.
e. `transactionId` – unique cart id or transaction id or order id maintained by the client
f. `userId` – user's unique id such as user's login id or email address

Note:

- Use **AES algorithm** to build the encrypted **`refid`** value
- Refer *Appendix* to follow the **Client Onboarding process** to get the **`client_id`** & **`secret_key`**.
- Refer *Appendix* to get & use the encryption key required to generate the **`refid`**.

`version` – (mandatory) props of `<pyw-app>` element. To be added for version number.

`params` – (optional) props (JSON) of `<pyw-app>` element. To be added to share user information with PYW.

`usertype` – (optional) value. Type of the customer.
R – Registered Member
G – Guest

`useremail` – (optional) value. Email of the user.

`Multiple parameters can be added.`

JavaScript

```
params='{
           "usertype":"R",
           "useremail":"test@test.com",
        }'
```

`button` – (optional) props (JSON) `<pyw-app>` element. To be added to customize button name.

`name` – (mandatory) value. Provided as button name.

JavaScript

```
button='{
         "name" : "Continue to payment",
       }'
```

link – (optional) props (JSON) `<pyw-app>` element. To be added to customize link name.
name – (mandatory) value. Provided as linkname

```javascript
link ='{
          "name" : "Continue as a guest user",
       }'
```

(Note - Button OR Link, one of them is required. If both are provided then only Button will be shown**)**

API Specifications

1. Contactless Payment API

This API can be used to initiate a PYW payment for a contactless transaction e.g. any payment triggered via POS or any other offline system w.r.t. the end user.

Environment: Available environments are UAT & Production. The details are,

Environment	URL
UAT	https://checkout.uat.telluride.shopyourway.com
Production	https://checkout.telluride.shopyourway.com

Endpoint: **POST** /tell/api/checkout/v1/contactlesspay

Headers:

- Content-Type: application/json
- Accept: application/json
- platform: PYW
- merchantClientId: registered client id on Telluride oAuth service
- transactionId: client's transaction id or cart id or order id
- refid: encrypted value of client_id~~access_token~~requestorId~~timestamp~~transactionId~~userId

Note:

- Use **AES algorithm** to build the encrypted *refid* value using the same secret key received during client onboarding process
- requestorId can be hardcoded to *PSYW*

- `userId` is optional
- Refer *Fields & Attributes Definition* section for more details about `refid`

Request structure: See appendix for sample json.

```
{
    "totalDue": (conditional) Total amount to be paid in USD. If "items" element is
    present then totalDue can be ignored,

    "returnUrl": (conditional) Specify a page url to which the redirection can be
    done after payment is completed in PYW. PYW will append payment status and id
    to this url. It should be a complete http(s) path,

    "postbackUrl": (mandatory) Specify a url to pass the payment id, status &
    merchant transaction id after the payment is complete. It can be an API
    endpoint. It should be a complete http(s) path,

    "mobile": (optional) pass valid mobile number if payment link is to be sent
    over the SMS,

    "email": (optional) pass valid email address if payment link is to be sent over
    the email,

    "payType": (optional) transaction/payment type - if merchant wants to
    distinguish multiple transactions to be processed with PYW,

    "items": (optional) [ - array: To be added if the cart item details need to be
    shared with PYW.
        {
            "id": (mandatory) Id of item being purchased,
            "name": (mandatory) Name of item being purchased,
            "imageUrl": (optional) Logo/image of item being purchased,
            "qty": (optional) default "1". Quantity of item being purchased,
            "price": (mandatory) Unit price of item being purchased,
            "unit": (optional) default is "usd". Currency unit for item price

        }
    ],
    "addresses": (optional) [ - array: To be added if shipping or billing address
    needs to be shared with PYW; The address details can be used for fraud
    prevention.
        {
            "type": (optional) Set "S" - for Shipping & "B" - for Billing, default is
            considered as same address for Shipping & Billing,
            "line1": (mandatory) Address line 1,
            "line2": (optional) Address line 2,
            "city": (mandatory) city of address,
            "state": (mandatory) state of address,
            "zip": (mandatory) zipcode of address

        }
    ]
}
```

Response structure & attributes' details:

a) Success Flow

```
{
   "status": "success",
   "pywid":  unique payment id e.g. "d13fe30a-9ece-49b1-bc0e-fa89c9f45c9b",
   "merchantTxnId": transaction id (or a reference id) that is received from
   client within an encrypted refid.
}
```

b) Error Flow:

```
{
   "status": "error",
   "merchantTxnId": transaction id (or a reference id) that is received from
   client within encrypted refid.,
   "errors": {
      "code": error code (String)
      "message": error message (String)
   }
}
```

2. Generate Payment Id API

This API can be used to initiate a PYW payment for a transaction triggered from any button click on a checkout page. E.g. any payment triggered from a mobile application.

Environment: Available environments are UAT & Production. The details are,

Environment	URL
UAT	https://checkout.uat.telluride.shopyourway.com
Production	https://checkout.telluride.shopyourway.com

Endpoint: **POST** /tell/api/checkout/v1/generatepaymentid

Headers:

- Content-Type: application/json
- Accept: application/json
- platform: PYW
- merchantClientId: registered client id on Telluride oAuth service
- transactionId: client's transaction id or cart id or order id
- refid: encrypted value of client_id~~access_token~~requestorId~~timestamp~~transactionId~~userId

Note:

- Use **AES algorithm** to build the encrypted *refid* value using the same secret key received during client onboarding process
- `requestorId` can be hardcoded to *PSYW*
- `userId` is optional
- Refer *Fields & Attributes Definition* section for more details about `refid`

Request structure: See appendix for sample json.

```
{
    "totalDue": (conditional) Total amount to be paid in USD. If "items" element is
    present then totalDue can be ignored,

    "returnUrl": (conditional) Specify a page url to which the redirection can be
    done after payment is completed in PYW. PYW will append payment id & status to
    this url. It should be a complete http(s) path,

    "postbackUrl": (conditional) Specify a url to pass the payment id, status &
    merchant transaction id after the payment is complete. It can be an API
    endpoint. It should be a complete http(s) path,

    "payType": (optional) transaction/payment type - if merchant wants to
    distinguish different transactions to be processed with PYW,

    "items": (optional) [ - array: To be added if the cart item details need to be
    shared with PYW.
        {
            "id": (mandatory) Id of item being purchased,
            "name": (mandatory) Name of item being purchased,
            "imageUrl": (optional) Logo/image of item being purchased,
            "qty": (optional) default "1". Quantity of item being purchased,
            "price": (mandatory) Unit price of item being purchased,
            "unit": (optional) default is "usd". Currency unit for item price

        }
    ],
    "addresses": (optional) [ - array: To be added if shipping or billing address
    needs to be shared with PYW; The address details can be used for fraud
    prevention.
        {
            "type": (optional) Set "S" - for Shipping & "B" - for Billing, default is
            considered as same address for Shipping & Billing,
            "line1": (mandatory) Address line 1,
            "line2": (optional) Address line 2,
            "city": (mandatory) city of address,
            "state": (mandatory) state of address,
            "zip": (mandatory) zipcode of address

        }
    ]
}
```

Response structure & attributes' details:

c) Success Flow

```
{
    "status": "success",
    "pywid":  unique payment id e.g. "d13fe30a-9ece-49b1-bc0e-fa89c9f45c9b",
    "merchantTxnId": transaction id (or a reference id) that is received from
    client within encrypted refid,
    "pmtid":  unique alphanumeric code (of 8 to 12 chars)
}
```

d) Error Flow:

```
{
    "status": "error",
    "merchantTxnId": transaction id (or a reference id) that is received from
    client within encrypted refid.,
    "errors": {
        "code": error code (String)
        "message": error message (String)
    }
}
```

Note:

Use the value of "pmtid" attribute from /tell/api/checkout/v1/generatepaymentid API response to initiate the PYW payment using web url .

Pass value of "pmtid" to the "id" parameter and "merchant" parameter with value of merchantClientId over the URL for initiating PYW payment.

UAT:
https://pywweb.uat.telluride.shopyourway.com/pmt?id=3ZCVQU56E2&merchant=<merchantClientId>

PROD:
https://pywweb.telluride.shopyourway.com/pmt?id=3ZCVQU56E2&merchant=<merchantClientId>

3. Payment Confirmation API

Order payment confirmation API is the handshake API which securely provides the payment details; it can also be used to reverse or cancel the payment.

On successful payment, the PYW returns the control to the client's returnUrl path – (which PYW received from the PYW component or from contactless API request) by passing a few parameters over the URL such as pywid, transactionid and pywmsg.

- pywid: unique reference id (UUID) for successful PYW payment. This param is not sent if payment is failed/declined i.e. when pywmsg=N
- transactionid: an id (a reference id, maybe clients order id) that is received from client within an encrypted refid.
- pywmsg: Y if the payment is successful otherwise N

Environment: Available environments are UAT & Production. The details are,

Environment	URL
UAT	https://checkout.uat.telluride.shopyourway.com
Production	https://checkout.telluride.shopyourway.com

Endpoint: GET /tell/api/checkout/v1/orderpaymentconfirm

Headers:

- Accept: application/json
- channel: ONLINE
- platform: PYW
- pywid: returned over the returnUrl passed from PYW on successful payment e.g. e.g. 64a12176-8821-4467-9b4d-c8c157a56422
- transactionId: client's transaction id or cart id or order id
- actionType: READONLY (default) / CONFIRM / CANCEL
  - READONLY: use it to get PYW payment details to decide whether to confirm/cancel the payment.
  - CONFIRM: use it to confirm the PYW payment.
  - CANCEL: use it to cancel the PYW payment.
- transactionType: 1P / 2P
  - 1P: 1st party checkout – when making payment for the merchant's own cart
  - 2P: 2nd party checkout – when making payment on behalf of other merchant's cart
- merchantClientId: the same client_id attribute used in building the *refid*
- refid: refer to **refid** related documentation under "Fields & Attributes Definition" section below
  - Encrypted value of the combination of client_id~~access_token~~requestorId~~timestamp~~transactionId~~userId
  - Use **AES algorithm** to build the encrypted refid value

Note:

- First call this API with **actionType=READONLY** which will provide the payment details including the `orderDate` and the `paymentTotal` which can be verified by the merchant with the *totalDue* amount for the given transaction.
- If the client's *totalDue* value matches with the paymentTotal from the READONLY API response, then call the same API with **actionType=CONFIRM**
- Otherwise call the same API with **actionType=CANCEL**
- This will avoid any risks of the amount getting modified while calling the PYW.

Response structure & attributes' details: See appendix for sample json.

a) Response when **actionType=READONLY** (request header)

```
{
    "authCode": unique payment authorization code e.g. 22112098011020292 or
AAB2C6 (any String),
    "transactionId": client's transaction id or cart id or order id (from
refid),
    "orderDate": date of order e.g. 09-30-2021 09:02:10,
    "paymentStatus": payment status e.g. CONFIRMED, CANCELLED
    "paymentTotal": payment amount in USD,
    "paymentDetails": (Optional - if merchant needs the payment distribution
details) [ --> one or many payments are supported
       {
            "id": payment instance id,
            "type": payment type e.g. SYWR, CREDITCARD,
            "amount": payment amount in USD paid using this payment type,
            "status": payment status for this payment type,
            "currency": payment currency,
            "cardLastFour": credit card last 4 digits - if type=CREDITCARD,
            "cardType": credit card type e.g. Visa - if type=CREDITCARD,
       }
    ],
    "additionalInfo": (Optional) Will use if any additional info needs to be
passed on to the merchant.
}
```

b) Response when **actionType=CONFIRM** (request header)

```
{
    "authCode": unique payment authorization code e.g. 22112098011020292 or
AAB2C6 (any String),
    "transactionId": client's transaction id or cart id or order id (from
refid),
    "orderDate": date of order e.g. 09-30-2021 09:02:10,
    "paymentStatus": payment status e.g. CONFIRMED, CANCELLED
    "paymentTotal": payment amount in USD,
    "paymentDetails": (Optional - if merchant needs the payment distribution
details) [ --> one or many payments are supported
       {
            "id": payment instance id,
            "type": payment type e.g. SYWR, CREDITCARD,
            "amount": payment amount in USD paid using this payment type,
            "status": payment status for this payment type,
            "currency": payment currency,
            "cardLastFour": credit card last 4 digits - if type=CREDITCARD,
            "cardType": credit card type e.g. Visa - if type=CREDITCARD,
       }
    ],
    "additionalInfo": {
        "address": {   -> user's address details
            "addressLine1": "17213 Cole Rd",
            "addressLine2": "",
            "state": "MD",
            "city": "Hagerstown",
            "postalCode": "21740",
            "shcAssociateInd": "N",
            "pinNumber": "93536",
```

```
                "zipCodeExtension": ""
            }
        }
    }
```

4. Return Payment API

This API can be used to return the full or partial payment for the given payment authcode.

Environment: Available environments are UAT & Production. The details are,

Environment	URL
UAT	https://payment.uat.telluride.shopyourway.com
Production	https://payment.telluride.shopyourway.com

Endpoint: **POST** /tell/api/payment/v1/return

Headers:

- Content-Type: application/json
- Accept: application/json
- channel: ONLINE
- platform: PYW
- merchantClientId: registered client id on oAuth service
- transactionId: Client transaction ID e.g. 10001
- refid: client_id~~access_token~~requestorId~~timestamp~~transactionId~~userId
  *-- Use **AES algorithm** to build the encrypted refid value using the same secret key received during client onboarding process*

  Note:

  - requestorId – PSYW
  - userId – can be empty or ignored
  - Rest all fields from refid are mandatory
  - Refer *Fields & Attributes Definition* section for more details about refid

Request structure: See appendix for sample json.

```
{
   "authCode": from orderpaymentconfirm API response e.g. 22112098011020292 or AAB2C6 (any String),
   "returnAmount": amount to be returned for already made payment
}
```

Response structure & attributes' details: See appendix for sample json

e) Success Response

```
{
     "returnCode": "22112098011021556",
     "returnDate": datetime on which the payment is returned
     "returnPaymentStatus" : [ - array: one or many returned payment instances here
       {
         "id": payment id which is returned,
          "type": payment type e.g. SYWR|CREDITCARD,
          "amount": payment amount in USD paid using this payment type,
          "currency": payment currency,
          "cardLastFour": credit card last 4 digits - Only if type=CREDITCARD,
          "cardType": credit card type e.g. Visa - Only if type=CREDITCARD
        }
    ],
    "status": "success"
}
```

f) Error Response

```
    "status": "error",
    "errors": {
           "code": error code (String)
           "message": error message (String)
    }
```

5. Subscription renewal payment API

Subscription renewal payment API can be used to process the payment for the subscription renewal.

Environment: Available environments are UAT & Production. The details are,

Environment	URL
UAT	https://checkout.uat.telluride.shopyourway.com
Production	https://checkout.telluride.shopyourway.com

Endpoint: **POST** /tell/api/subscription/v1/authorize

Headers:

- Content-Type: application/json
- Accept: application/json
- platform: PYW
- merchantClientId: registered client id on Telluride oAuth service
- refid: encrypted value of
  client_id~~access_token~~requestorId~~timestamp~~transactionId~~userId

Note:

- Use **AES algorithm** to build the encrypted *refid* value using the same secret

key received during client onboarding process

- `requestorId` can be hardcoded to *PSYW*
- `userId` is optional
- Refer to *Fields & Attributes Definition (Tag Javascript)* section for more details about `refid`

Request structure: See appendix for sample json.

```
{
   "orderAuthCode": Auth code return in payment confirm API while processing the
                    first payment for subscription.
   "subscriptionId": This is the subscriptionID provided during the 1st
                     subscription payment under pay-type / payType node.
   "price": payment amount ($) to be charged to CC for subscription renewal.
}
```

Response structure & attributes' details:

a) Success Flow

```
{
    "responseCode": API response code. 200 is success response code,
    "responseMessage": API response message. "SUCCESS" for success
                       response
    "authCode": unique payment authorization code,
    "transactionId": client's transaction id or cart id or order id
                     (from refid),
    "orderDate": date of order e.g. 09-30-2021 09:02:10,
    "paymentStatus": payment status e.g. CONFIRMED,
    "paymentTotal": payment amount in USD,
    "paymentDetails": Payment distribution details
    [ --> one or many payments are supported
       {
            "id": payment instance id,
            "type": payment type e.g. SYWR, CREDITCARD,
            "amount": payment amount in USD paid using this payment type,
            "status": payment status for this payment type,
            "currency": payment currency,
            "cardLastFour": credit card last 4 digits - if
            type=CREDITCARD,
            "cardType": credit card type e.g. Visa - if type=CREDITCARD,
       }
    ],
    "additionalInfo": {
        "address": { -> user's address details
            "addressLine1": "17213 Cole Rd",
            "addressLine2": "",
            "state": "MD",
            "city": "Hagerstown",
            "postalCode": "21740",
            "shcAssociateInd": "N",
            "pinNumber": "93536",
            "zipCodeExtension": ""
        }
    },
   "subscriptionId": "Subscription Id",
   "subscriptionText": "Subscription renewed Successfully"
```

```
}
```

6. Update Order API (Applicable for TYPE-A integration)

Update Order API can be used to correct the total due amount paid earlier within 30 mins.

Environment: Available environments are UAT & Production. The details are,

Environment	URL
UAT	https://checkout.uat.telluride.shopyourway.com
Production	https://checkout.telluride.shopyourway.com

Endpoint: **POST** /tell/api/checkout/v1/updateorder

Headers:

- Accept: application/json
- channel: ONLINE
- platform: PYW
- pywid: returned over the returnUrl passed from PYW UI on successful payment e.g. e.g. 64a12176-8821-4467-9b4d-c8c157a56422
- transactionId: client's transaction id or cart id or order id
- merchantClientId: the same client_id attribute used in building the *refid*
- refid: refer to **refid** related documentation under "Fields & Attributes Definition" section below
  - Encrypted value of the combination of client_id~~access_token~~requestorId~~timestamp~~transactionId~~userId
  - Use **AES algorithm** to build the encrypted refid value

Request structure: See appendix for sample json.

```
{
   "modifiedTotalDue": payment amount to be corrected for the given pywid,
   "addresses": [ (optional) - array of addresses if changed
                              viz. shipping & billing address
      {
          "type": S-for Shipping & B-for Billing (if missing then same address is
                 considered for Shipping & Billing),
          "line1": shipping/billing address line 1,
          "line2": (Optional) shipping/billing address line 2,
          "city": shipping/billing address city,
          "state": shipping/billing address state,
          "zip": shipping/billing address zipcode
      }
   ]
}
```

Response structure & attributes' details:

a) Success Response

```
{
    "pywid": reference id (UUID) for successful PYW payment. This param is not sent
    if payment is failed/declined i.e. when pywmsg=N
    "pywmsg": Y if the payment is successful otherwise N
    "transactionId": client's transaction id or cart id or order id (from refid),

}
```

b) Error Response

```
{
    "pywmsg": "N"
    "transactionId ": client's transaction id or cart id or order id (from refid),
    "errors": {
           "code": error code (String)
           "message": error message (String)
    }
}
```

Appendix

1. Client onboarding document `(Merchant_Onboarding_v1.6.pdf - refer from enclosing zip file)`
2. Document for building & encrypting the `refid (Refid_Encryption_Key_v1.0.pdf - refer from enclosing zip file)`
3. To add a new Public Key / Certificate renewal
   a. Login to OAuth Portal.
      - If you don't have a password please use the "forgot password" option to reset the password using your client_id/merchant_id & registered email.

      - OAuth Portal URL,
        - UAT: https://oauthweb.uat.telluride.shopyourway.com/oauthUI/login
        - Prod : https://oauthweb.telluride.shopyourway.com/oauthUI/login

   b. After Login you will see (+) sign next your existing certificate rows. Click the (+) sign to add a new Public Key.

4. Sample request and response for Contactless API.

```
Request :
{
  "totalDue": "10.00",
  "postbackUrl": "https://merchant.com/paymentstatus ",
  "returnUrl": "https://merchant.com/paymentsummary"
  "mobile": "8474013322",
  "email": "member@merchant.com",
  "payType": "typeA",
  "items": [
    {
      "id": "12345abcde",
      "name": "shoes",
      "imageUrl": "https:///merchant.com/item/image/12345abcde",
      "qty": "1",
      "price": "15.59",
      "unit": "USD"
    }
  ],
  "addresses": [
    {
      "type": "S",
      "line1": "3333 Beverly Road",
      "line2": "Suite 1234",
      "city": "Hoffman Estates",
      "state": "IL",
      "zip": "60179"
    }
  ]
}


Response:
{
  "status": "success",
  "pywid": "d13fe30a-9ece-49b1-bc0e-fa89c9f45c9b",
```

```
  "transactionId": "545345345345"
}
```


5. Sample response for order Payment Confirmation API – READONLY


```
{
  "authCode": "0918020220208910874884",
  "transactionId ": "545345345345",
  "orderDate": "02-08-2022 11:19:56",
  "paymentStatus": "CONFIRMED",
  "paymentTotal": "4.00",
  "paymentDetails": [
    {
      "id": "16443407",
      "type": "SYWR",
      "amount": "0.01",
      "status": "SUCCESS",
      "currency": "USD"
    },
    {
      "id": "ab6730d4-e873-47c4-9b79-de7edbf59f76",
      "type": "CREDITCARD",
      "amount": "3.99",
      "status": "SUCCESS",
      "currency": "USD",
      "cardLastFour": "9535",
      "cardType": "Sears Master"
    }
  ],
  "additionalInfo":
"wdsaw%2B9efrC%2BHysderdBD6XSF3b0ZL6SIZo9HK26ubrVE0VOHFD6Jf33wwsskVRGVy"
}
```


6. Sample response for payment Confirmation API – CONFIRM


```
{
  "authCode": "0918020220208910874884",
  "transactionId ": "545345345345",
  "orderDate": "02-08-2022 11:19:56",
  "paymentStatus": "CONFIRMED",
  "paymentTotal": "4.00",
  "paymentDetails": [
    {
      "id": "16443407",
      "type": "SYWR",
      "amount": "0.01",
      "status": "SUCCESS",
      "currency": "USD"
    },
    {
      "id": "ab6730d4-e873-47c4-9b79-de7edbf59f76",
      "type": "CREDITCARD",
      "amount": "3.99",
      "status": "SUCCESS",
      "currency": "USD",
      "cardLastFour": "9535",
      "cardType": "Sears Master"
    }
  ],
  "additionalInfo": {
    "address": {
      "city": "Kinley",
```

```
      "postalCode": "60168",
      "addressLine1": "Water cross Rd",
      "pinNumber": "00144",
      "addressLine2": "",
      "state": "IL",
      "zipCodeExtension": "3344",
      "shcAssociateInd": "N"
    }
  },
  "responseMessage": "SUCCESS",
  "responseCode": "200"
}
```


7. Return Payment API – Sample request and response.

Request:


```
{
  "authCode": "22112098011020292",
  "returnAmount": "100.00"
}
```


Response:


```
{
  "returnCode": "22112098011021556",
  "transactionId ": "545345345345",
  "returnDate": "02-08-2022 11:19:56",
  "returnPaymentStatus": [
    {
      "id": "ab6730d4-e873-47c4-9b79-de7edbf59f76",
      "type": "CREDITCARD",
      "amount": "80.00",
      "currency": "USD",
      "cardLastFour": "4455"
      "cardType": "Sears Master"
    },
    {
      "id": "6879a9ea-7d7a-4965-ae38-46314d28af3a",
      "type": "SYWR",
      "amount": "20.00",
      "currency": "USD"
    }
  ],
  "status": "success"
}
```


8. Sample request and response for subscription renewal payment API.

Request :


```
{
    "orderAuthCode": "YCRLUP",
    "subscriptionId": "S13",
    "price": "99.00"
}
```


Response:


```
{
    "authCode": "JX2PS9",
    "orderDate": "09-22-2022 04:12:27",
    "paymentStatus": "CONFIRMED",
    "paymentTotal": "99.00",
    "paymentDetails": [
        {
```

```
            "id": "1CEEC0E1000DCB721939312540682C60",
            "type": "CREDITCARD",
            "amount": "99.00",
            "status": "SUCCESS",
            "currency": "USD",
            "cardLastFour": "7110",
            "cardType": "VISA"
        }
    ],
    "additionalInfo": {
        "address": {
            "addressLine1": "Address1",
            "addressLine2": "",
            "state": "IL",
            "city": "HE",
            "postalCode": "88011",
            "shcAssociateInd": "N",
            "pinNumber": "3586",
            "zipCodeExtension": ""
        }
    },
    "responseCode": "200",
    "responseMessage": "SUCCESS",
    "transactionId": "123",
    "subscriptionId": "S13",
    "subscriptionText": "Subscription renewed Successfully"
}
```


9. Update Order Api Sample request.

Request :


```
{
  "modifiedTotalDue": "10.90",
  "addresses": [
    {
      "type": "S",
      "line1": "3333 Beverly Road",
      "line2": "Suite 1234",
      "city": "Hoffman Estates",
      "state": "IL",
      "zip": "60179"
    },
    {
      "type": "B",
      "line1": "18 High Hill Rd",
      "line2": "Apt 111",
      "city": "Tesco Estates",
      "state": "IL",
      "zip": "60193"
    }
  ]
}
```


Response:


```
{
  "pywid": "ab6730d4-e873-47c4-9b79-de7edbf59f76",
  "transactionId ": "545345345345",
  "pywmsg": "Y"
}
```

10. Sample GeneratePaymentId Request and Response

Request:

```
{
    "totalDue": "1.00",
    "returnUrl": "https://shopify.com"
}
```

Response:

```
{
    "status": "success",
    "pywid": "f1cd29cb-10a2-42dc-94d4-3c435244517b",
    "merchantTxnId": "123456",
    "pmtid": "95EMGG5ENH"
}
```