How to Integrate with Atlas Interpreter API

This guide will show how a merchant could integrate their own system with Expitrans Payments Gateway.

First, the merchant should get an Expitrans API Token. This token **must be provided with any request to Expitrans servers.**
The token can be attached to the request in two ways:
1. As a User Name in [Basic Authorization](https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication#basic_authentication_scheme) scheme; in this case, the token appears in Authorization HTTP header, as a **key param in URL (e.x. `https://devel01.expitrans.com/webapis?key=<TOKEN>`)**
2. Or a merchant has the ability to provide finish and cancelation urls for Secure Pay mechanism.

Atlas Interpreter Integration

Expitrans provides a javascript library that is similar to the widely used [Stripe.js](https://stripe.com/docs/js) library.
There are 3 steps to complete the integration:

Step 1: Payment Intent Generation

This step **must occur on the server side. Below there is an example in [PHP](https://www.php.net/) but you could use any methods you like to create and get Token Intent at Expitrans Webapi Server.**

```php
$intent_ch = curl_init("https://devel01.expitrans.com/webapis/v1/payment_intents");
curl_setopt_array($intent_ch, [
    CURLOPT_POST => true,
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_POSTFIELDS => http_build_query(['amount' => 10]),
    CURLOPT_USERPWD => "$token:",
    CURLOPT_HTTPAUTH => CURLAUTH_BASIC,
]);
$intent_response = curl_exec($intent_ch);
curl_close($intent_ch);
```

Here the payment intent request sends the payment amount in the body of [POST HTTP request](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST)

Expitrans Webapi Server will respond with JSON-formatted data about the created Payment Intent. To process the payment generated Payment Intent, there should be confirmation by the payer.

Step 2: Create Payment HTML Page

This page will be shown to the payer and should get payment data like card details and billing details. At this step the user should input all the data and submit the form to confirm the Payment Intent created in the step above. Integrator could use the Stripe.js alike javascript library to process all data to the Expitrans Webapi Server.
To have an opportunity to use such a library, the integrator should include a javascript library to HTML code of the page.

```html
<script src="https://devel01.expitrans.com/webapis/lib/v3/lib.js" async></script>
```

Step 3: Confirm Payment Intent

In this step the integrator should allow the user to input all payment data and submit them into the server. All errors or mistakes that the user makes while filling out the form could be displayed at the same page.
There are two requirements at this step:
1. `merchant_token` merchant webapi token -- is needed to create an instance of `Stripe` object that would communicate with Expitrans Webapi Server on behalf of the merchant.
2. `client_secret` unique string for every token generated by Expitrans Server as a response of *Step 1* above -- needed by Expitrans Server to identify early generated Payment Intent.

```javascript
// create instance of Stripe alike lib with merchant token
const stripe = Stripe(merchant_token);
const elements = stripe.elements();
// create dom node that will hold card num and cvv code
const card = elements.create('card', { hidePostalCode: true });

// node handles errors; like wrong card number
const err = document.querySelector('#card-errors');

// this function will run when user types in card element;
// if there are any error in card number or cvv, them will be shown in
// #card-errors node
function errHandler({ error }) {
    err.hidden = !error;
    err.textContent = error ? error.message : ''
}

// wait payment details form to be loaded
document.addEventListener('DOMContentLoaded', () => {
    card.addEventListener('change', errHandler);

    // try to confirm the payment intent
    // when payor submit the form
```

```
    document.querySelector('#submit').addEventListener('click', async () => {
        // get all fields with billing details
        // like payor address, zip, first name, last name, phone, email
        const flds = document.querySelectorAll('#billing_details [name]');
        const billing_details = {};

        flds.forEach(f => {
            billing_details[f.name] = f.value;
        });

        // make a http request to confirm the payment intent
        const re = await stripe.confirmCardPayment(
            // secret token string given from response at step 1
            client_secret,
            {
                payment_method: {
                    card, // card node
                    billing_details
                }
            },
            { handleActions: false }
        );

        // payment intent cannot be confirmed, show the error message
        if (re.error) {
            errHandler(re);
            return;
        }

        // clear the card num & cvv node
        card.clear();
        // the paymentIntent object holds information
        // about confirmed payment
        alert(`
        Payment status: ${ re.paymentIntent.status }
        Text: ${ re.paymentIntent.charges.data[0].outcome.seller_message }`);
        location.reload();
    });

    // mount card node to enter card num & cvv
    card.mount('#card-element');
});
```

Please refer to [Stripe.js documentation](https://stripe.com/docs/js/payment_intents) for detailed reference and examples of using javascript library.