Telesales API SDK

Learn how to integrate and use the Telesales API SDK with the Affirm Telesales UI.

Overview

The telesales API SDK provides a streamlined interface to aid sales agents over the phone. With this API, agents can send emails to customers and assist them with using Affirm. This will enable your sales agents to monitor customers' application processes in real-time and assist them in completing their Affirm transactions.

How it works

The Telesales API SDK works in several stages:

1. The sales agent initiates the application by clicking Send Checkout and triggering an email and/or text to the customer with instructions to complete their Affirm application.
2. The application uses webhooks to update the agent's interface, depending on where the customer is in the Affirm application.
3. If the customer doesn't receive the initial email, the agent can click Resend Checkout to send another email.
4. Once the agent sees the Confirmed status, the Affirm system creates the order.

To collect payment information for the transaction, you will need to implement additional logic to authorize and capture the payment for the transaction. You can repurpose this logic by updating the API keys to the Telesales-specific keys if you implemented Affirm in your e-commerce channel. See our Telesales documentation for more details.

Logic Flow

1. POST the Affirm `checkout_object` to the `[checkout/store](https://docs.affirm.com/developers/reference/store)` endpoint.
2. Webhooks send the event status updates to the server.

3. The application listens for server updates and provides relevant information to the agent.

Implementation

Step 1: Open and update the app.js file

Open the app.js file and update the relevant HTML class/element names on to ensure that the correct item information is pulled from the telesales agent's user interface. Note: You may need to update placeholders within the `checkout_object` variable, such as `item_URL` or `shipping_type`.

JavaScript

```javascript
// Grab form values so we can map them to the Affirm checkout_object
var billingFirst = document.getElementById('firstName').value,
billingLast = document.getElementById('lastName').value,
billingAddress1 = document.getElementById('addressLine1').value,
billingAddress2 = document.getElementById('addressLine2').value,
billingCity = document.getElementById('city').value,
billingState = document.getElementById('state').value,
billingEmail = document.getElementById('email').value,
billingZip = document.getElementById('zipcode').value,
billingPhone = document.getElementById('phoneNumber').value,
productName = document.getElementById('productName').value,
productQuantity = document.getElementById('productQuantity').value,
unitPrice = document.getElementById('unitPrice').value,
productName2 = document.getElementById('productName2').value,
productQuantity2 = document.getElementById('productQuantity2').value,
unitPrice2 = document.getElementById('unitPrice2').value,
tax = document.getElementById('tax').value,
shipping = document.getElementById('shipping').value,
discount = document.getElementById('discount').value,
discount2 = document.getElementById('discount').value,
total = document.getElementById('total').value;
order_id = document.getElementById('order_id').value;
```

Step 2: Pass the values as cents

Note that all the values within the `checkout_object` must be passed to Affirm as cents (e.g., $100 is 10000 cents). The `toInteger` function on handles this conversion.

JavaScript

```javascript
// Money helper - Affirm calculates all values in cents e.g. $100 is 10000
function toInteger(a) {
       var b;
       a = a.replace(/[$,]/g,"");
       if (a.indexOf('.') > 0) {
              b = a.replace(/[.]/g,"");
       }
       else {
              b = a * 100;
       }
       return b
}
```

Affirm checkout function

The `app.js` file also contains the `affirmCheckout` function. This function sends the `checkout_object` to Affirm and updates the agent's Affirm UI upon receipt of successful webhook responses. The webhook responses are stored in a database and pinged every three seconds by the `getWHR` function. The agent’s interface will only update if the customer’s `order_id` matches that in the webhook response.

Understand webhook event updates

Webhook event updates include the following:

- `Sent`: Confirmed that the email has been sent to the customer.
- `Opened`: Confirms that the customer has opened their email.
- `Approved`: Indicates that the customer’s application was approved.
- `Not_approved`: Indicates that the customer’s application was declined.
- `Confirmed`: Confirms that the customer has completed their Affirm loan and payment for the order is complete.

📘 Note

Note that we require a data-sharing agreement in place to share webhook events. A webhook URL will also need to be configured to complete the integration. Please reach out to Affirm through your Account team or the support widget.

About telesalesserver.js

The telesalesserver.js file represents the endpoint where the webhook data will be sent. Its function is to route the webhook event data to the client side, allowing the agent's interface to update correctly.

Update the HTML code for the agent

The HTML code for the agent is located within the `telesales-orderform.html`. You can style the agent window using the `style.css` file.

HTML

```html
<div class="max-width">
    <h3 class="header">Affirm payment details</h3>
    <div class="steps-container">
      <button class="sendcheckout">Send Checkout</button>
      <button class="resendcheckout inactive">Re-send checkout</button>
    </div>
    <div class="progress-container">
      <div class="progress-bar"></div>
    </div>
    <p class="helper-text visually-hidden">A link to start the Affirm loan
application has been emailed to the customer. They can start the
application on their own device</p>
  </div>

<script async="" src="affirm.js"></script>
<script src="app.js" type="text/javascript"></script>
```