iFrames integration

Benefits of iFrame

This option allows the buyer to stay in a familiar and trusted environment.

The payment tunnel is smooth and cart abandonment is less frequent.

It is possible to use a custom Template for the card form which will be integrated into the payment page. For more details: Create a custom template

The iframe is web responsive and can be integrated into a mobile application.

Table of Contents

Summary of the iframe implementation

- Implementation of the iframe in the *merchant's payment page*
- Configuration of URLs passed in the "urls" object of the payment request
  - Preparation of the `urls.return` and `urls.cancel` parameters with the URLs for successful and abandoned payment.
  - Placement of these 2 parameters in the Data parameter hash
  - Passing Data as a parameter in the URL provided to the src attribute of the iframe
  - Addition of the `CustomField14=iframe` parameter to display only the payment block
- Creation of the 'Success' and 'Failure' pages
  - Preparation of the HTML code for redirect-success and redirect-failure
- Implementation of JavaScript redirection
  - Integration of the postMessage method in redirect-success and redirect-failure
  - Integration of the event listener (Event Listener) in the *merchant's payment page*

Implementation of the iframe and configuration of the URL passed to the src attribute

Iframe tag & customField14

An iframe is an HTML tag that allows embedding an HTML page within another.

To do this, you need to use the <iframe> tag and specify the URL of the page you want to display in the src attribute. The URL of the payment page must include the following value: customField14=iframe.

Example of integrating the card form within an HTML page

```
<!DOCTYPE html>
<HTML lang="en">
  <HEAD>
    <META charset="UTF-8" />
    <META http-equiv="X-UA-Compatible" content="IE=edge" />
    <META name="viewport" content="width=device-width, initial-scale=1.0" />
    <TITLE>Merchant website - Order</TITLE>
  </HEAD>
  <BODY>
    ...
    <IFRAME
      src="https://paymentpage.axepta.bnpparibas/payssl.aspx?token=YOUR_PAYMENT_TOKEN"
    />
    ...
  </BODY>
</HTML>
```

Settings: Height & Width

In pixels

```
<IFRAME
  src="https://paymentpage.axepta.bnpparibas/payssl.aspx?token=YOUR_PAYMENT_TOKEN"
  height="600"
  width="800"
/>
```

In CSS

```
#payment-iframe {
  height: 600px;
  width: 800px;
}

// or for a fullscreen display
#payment-iframe {
  height: 100vh;
  width: 100vw;
}
<IFRAME
  id="payment-iframe"
  src="https://paymentpage.axepta.bnpparibas/payssl.aspx?token=YOUR_PAYMENT_TOKEN"
/>
```

Creation of 'Return' pages

When the bearer submits the payment form, a request is sent to the Axepta BNP Paribas server.

Depending on whether the user proceeds with the payment or abandons it, the response is returned to the return URL or the cancel URL.

If the payment goes to the end (succesfully or not), a response is sent to the urls.return address.
If the cardholder cancels the payment processing a response is sent to the urls.cancel address.

The goal is to display a success or failure page instead of the iFrame depending on the payment result.

Remember: if you receive a return callback, you still need to check if the payment have been approved or not.

You need to request the transaction details from the Axepta server, using the getByPayId API.

Returns or cancel urls are detailled in 'Callback and Payment Notification Management' documentation.

Note: the server returns an HTTP response with the POST method. Therefore, a route must be created on the merchant site's server to handle this response.

If you want to redirect the user to a page on the merchant site displayed instead of the iframe, you will need to provide a redirect page.

Depending on the result of the transaction details, we will have:

Axepta return url	urls.return		urls.cancel
getByPayId	OK	ERROR	n/a
redirection to the merchant's site	Successful payment	Failed payment	Failed payment

The merchant creates these pages which will be displayed during the bearer's redirection.

For example,

- Successful payment: https://mon-site.com/redirect-success
- Failed payment: https://mon-site.com/redirect-failure

Each page will run a script that will redirect the user to the payment success or error page.

The setup of this script is described in the following section.

Setting up redirection in JavaScript

Integration of the postMessage method in redirect-success and redirect-failure

The postMessage solution "allows secure cross-domain communication."

The goal is to "communicate" with the parent page. This communication will take place through a MessageEvent.

Handle returnUrl (success - failed) backend integration

```
const express = require('express');

const app = express();
const PORT = 3000;

const API_BASE_URL = 'https://paymentpage.axepta.bnpparibas';
const BEARER_TOKEN = 'YOUR_BEARER_TOKEN_HERE';

// Template function to generate HTML response based on payment status
function renderPaymentPage(payId, status, isSuccess) {
  const title = isSuccess ? 'Payment Successful' : 'Payment Failed';
  const message = isSuccess ? 'Payment Successful!' : 'Payment Failed';
  const eventType = isSuccess ? 'PAYMENT_SUCCESS' : 'PAYMENT_FAILED';
  const redirectUrl = isSuccess
```

```
    ? `/order/confirmation?payId=${payId}`
    : `/payment/retry?payId=${payId}`;

  return `
    <!DOCTYPE html>
    <html>
    <head>
      <title>${title}</title>
    </head>
    <body>
      <h1>${message}</h1>
      <p>Transaction ID: ${payId}</p>
      <script>
        // If page is loaded inside an iframe, notify the parent window
        if (window.parent !== window) {
          window.parent.postMessage({
            type: '${eventType}',
            payId: '${payId}',
            status: '${status}'
          }, '*');
        }

        // Redirect user after 3 seconds
        setTimeout(() => {
          window.location.href = '${redirectUrl}';
        }, 3000);
      </script>
    </body>
    </html>
  `;
}

// Return URL endpoint - This is where Axepta redirects customers after payment
app.get('/payment/return', async (req, res) => {
  try {
    // Step 1: Extract PayID from URL query parameters
    // Example: /payment/return?PayID=abc123
    const payId = req.query.PayID;

    // Validate that PayID exists
    if (!payId) {
      return res.status(400).send('PayID is missing');
    }

    // Step 2: Call Axepta API to retrieve payment details
    const response = await fetch(
      `${API_BASE_URL}/api/v2/payments/getByPayId/${payId}`,
      {
        method: 'GET',
        headers: {
          'Authorization': `Bearer ${BEARER_TOKEN}`,
          'Content-Type': 'application/json'
        }
      }
    );

    // Check if API request was successful
    if (!response.ok) {
      throw new Error(`API returned status ${response.status}`);
    }

    // Parse JSON response from API
    const paymentData = await response.json();
    const status = paymentData.status;

    // Step 3: Determine if payment was successful
    const isSuccess = status === 'SUCCESS' || status === 'AUTHORIZED';

    // Step 4: Send HTML response to customer
    res.send(renderPaymentPage(payId, status, isSuccess));
```

```
  } catch (error) {
    // Log error for debugging
    console.error('Error retrieving payment details:', error.message);

    // Return simple error message to customer
    res.status(500).send('Server error');
  }
});

// Start the server
app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```

Integration of the (Event Listener) in the payment page

In the page that contains the iframe, we will set up an event listener. It will be responsible for executing a function when it receives an event of type message.

Modification of the page code that integrates the iframe - Addition of EventListener

```
<!DOCTYPE html>
<HTML lang="en">
  <HEAD>
    <META charset="UTF-8" />
    <META http-equiv="X-UA-Compatible" content="IE=edge" />
    <META name="viewport" content="width=device-width, initial-scale=1.0" />
    <TITLE>Merchant website - order</TITLE>
  </HEAD>
  <BODY>
    <SCRIPT>
      // on the current page
      // we place a listener with the addEventListener method
      window.addEventListener(
        // event type to listen to
        "message",
        // callback function that will be executed
        // when an event of type "message" will be emitted
        (event) => {
          // If the origin is not the same as the one placed in the message
          // we stop the execution of the function
          if (event.origin !== "https://mon-site.com") return;

          // we redirect the user to the desired page
                        }
                      );
          window.location.href = "https://mon-site.com/payment-success";
        }
      );
    </SCRIPT>
    ...
    <IFRAME
      src="IFRAME src="https://paymentpage.axepta.bnpparibas/payssl.aspx?token=YOUR_PAYMENT_TOKEN"
    />
    ...
  </BODY>
</HTML>
```

Dynamic customization

Additional custom fields can be added to the request to dynamically change the iframe.

Please find below the available fields:

* CustomField100: background color
* CustomField101: text color of the buttons when they are active
* CustomField102: background color of the buttons when they are active
* CustomField103: hover color of the text of the buttons when the button is active
* CustomField104: hover color of the background of the buttons when the button is active
* CustomField105: text color of the buttons when they are inactive
* CustomField106: background color of the buttons when they are inactive

Hexadecimal color values must be used in the CustomField fields, and the symbol **'#' must be encoded**, it will be replaced by the value '%23'.
Thus, to display a red button (when it is activated), you will need to value the CustomField102 as follows: CustomField102=%23FF0000.