# Briqpay M2 extension

Developer guide


PAYMENTS UNLEASHED

# Introduction

**What is Briqpay**
Briqpay offers a payment integration module embedded as an iframe. This module allows merchants to add or remove payment providers without additional technical investment, as the configuration is managed in Briqpay's backoffice.

By translating information from various payment providers into a unified format, Briqpay streamlines the backend integration process. The payment module can be integrated into existing Magento checkout systems, either replacing all payment integrations or just specific ones.

**Note** that address handling, thank you pages, and other surrounding elements are managed outside of Briqpay.

**Briqpay for Magento Overview**

Welcome to the **Briqpay_Payments** Magento 2 extension! This powerful extension seamlessly integrates the Briqpay payment  into your Magento store, providing a comprehensive suite of features to streamline your payment integrations.

With Briqpay_Payments, you'll have access to essential functionalities including:

- Rendering the Briqpay Iframe: Handling of session creation and rendering of the briqpay iframe.
- Order Creation: Handling of order creation from a Briqpay order to a Magento 2 Order.
- Order Management: Full support of order management features such as captures and refunds.

While the extension offers a solid foundation, some customization is necessary to fully leverage Briqpay's capabilities. This guide will walk you through the necessary steps to get everything up and running, ensuring a seamless integration into your Magento store.

Let's get started!

## Prerequisites

- Magento 2.4.x or higher
- PHP 8.1 or higher
- Basic understanding of Magento 2 module development

# Installation

### 1. Install via Composer

To install the Briqpay Payments module, use Composer. Run the following commands in your Magento 2 root directory:

```
composer require briqpay/module-payments

bin/magento module:enable Briqpay_Payments

bin/magento setup:upgrade

bin/magento setup:di:compile

bin/magento setup:static-content:deploy
```

### 2. Manual Installation

Download the latest version of the module from the repository.
https://github.com/Briqpay-Extensions/Briqpay-m2-extension

Extract the contents of the zip file into the app/code/Briqpay/Payments directory of your Magento installation.

Enable the module and run the Magento setup commands:

```
bin/magento module:enable Briqpay_Payments

bin/magento setup:upgrade

bin/magento setup:di:compile

bin/magento setup:static-content:deploy
```

# Configuration

After installing the Briqpay_Payments extension, you can configure it to suit your needs by navigating to Stores > Configuration > Sales > Payment Methods > Briqpay. Below are the details for each configurable field:

### Enable

- **Description:** Enable or disable the Briqpay payment method in the checkout.
- **Options:** Yes / No
- **Usage:** Toggle this setting to "Yes" to activate Briqpay in your store's checkout process. Set it to "No" to deactivate it.

### Testmode

- **Description:** Choose between Briqpay's production or playground environments for API calls.
- **Options:** Yes (Playground) / No (Production)
- **Usage:** Toggle this setting to "Yes" to direct API calls to Briqpay's playground environment for testing purposes. Set it to "No" to use the production environment for live transactions.

### Client ID and Client Secret

- **Description:** API keys supplied by Briqpay to authenticate your integration.
- **Usage:** Obtain your Client ID and Client Secret by logging into your Briqpay account at Briqpay. Navigate to Developers > Keys and generate a set of API keys. Make sure to generate separate keys for the production and playground environments.
- **Note:** The Client ID and Client Secret must be set correctly to establish a connection with Briqpay's services.

### Checkout Type

- **Description:** Specify the type of customers your store serves.
- **Options:** Consumer / Business
- **Usage:** Select "Consumer" if your store primarily serves individual customers. Select "Business" if your store caters to businesses. Briqpay supports both customer types, but the extension requires you to specify the appropriate type for your storefront to ensure proper handling of transactions and customer data.

## Steps to Configure

1. **Navigate to Configuration:**
    - Go to Stores > Configuration > Sales > Payment Methods > Briqpay in the Magento admin panel.
2. **Enable Briqpay:**
    - Set the Enable field to "Yes".
3. **Set Testmode:**
    - Choose "Yes" for the playground environment or "No" for the production environment.
4. **Enter Client ID and Client Secret:**
    - Log in to your Briqpay account.
    - Go to Developers > Keys and generate the API keys.
    - Enter the Client ID and Client Secret in the corresponding fields in the configuration.
5. **Select Checkout Type:**
    - Choose the appropriate option (Consumer or Business) for your store under Checkout Type.
6. **Save Configuration:**
    - Click Save Config to apply the changes.

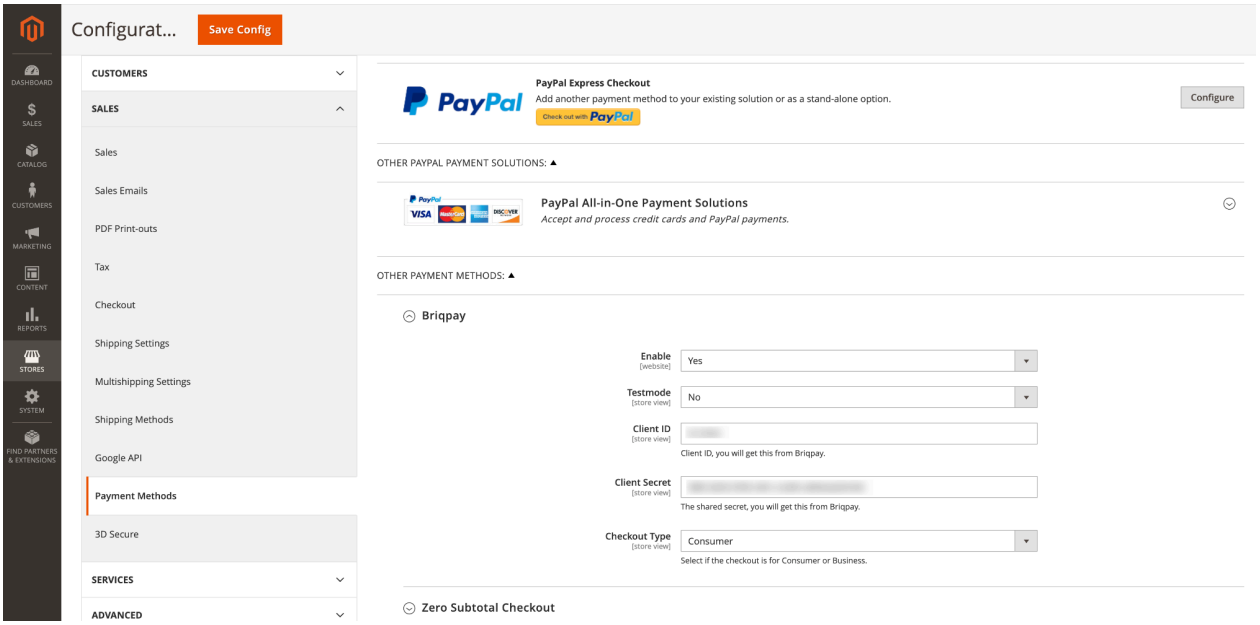



*Image displays expected view of the Briqpay configuration view.*

# Usage

## Usage Guide Introduction

The **Briqpay_Payments** Magento 2 extension integrates Briqpay into your Magento store, streamlining the checkout process for both consumers and businesses. This guide outlines the various steps involved in the purchase flow, highlighting key configurations and functionalities. We will dive deeper into each step to ensure a comprehensive understanding of the extension's capabilities.

### Purchase Flow Overview

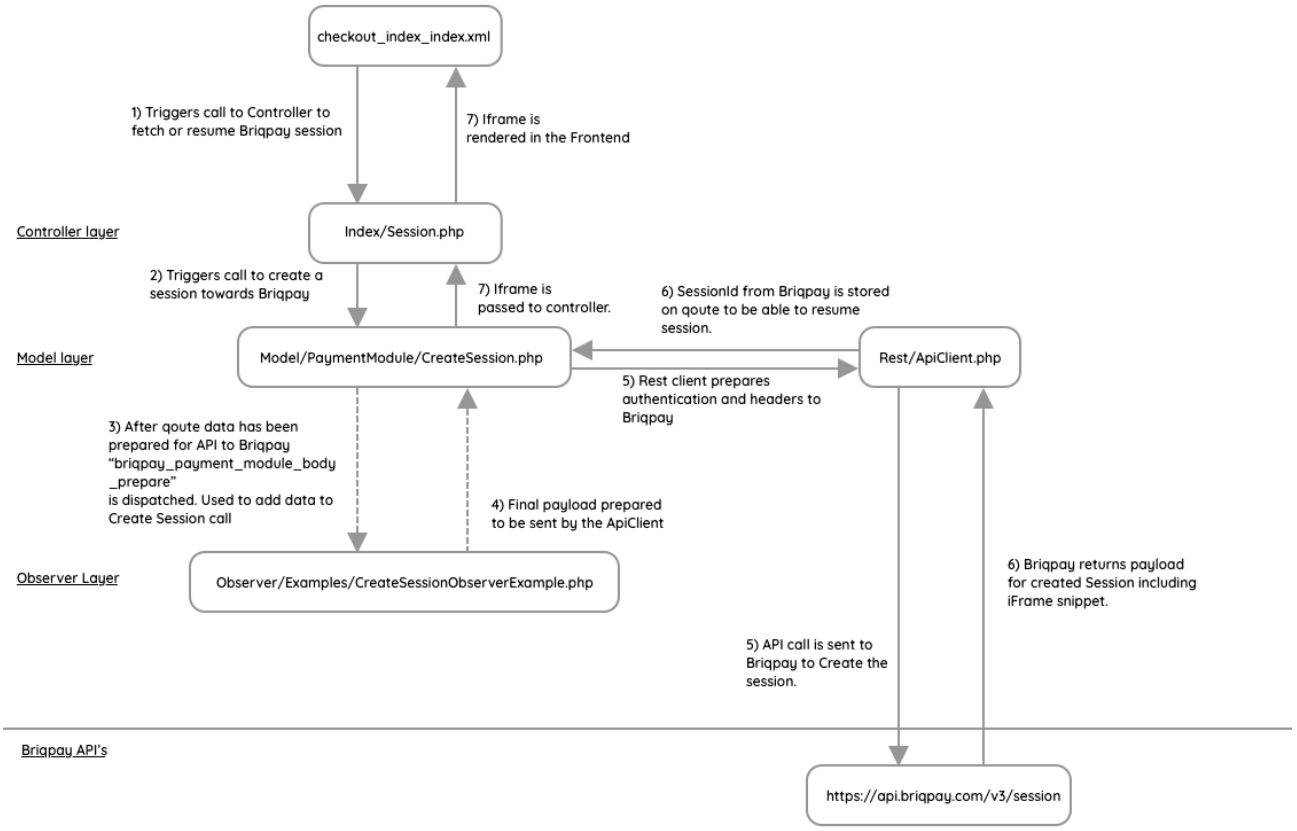
1. **Session Creation**
    - **Triggering Session Creation:** When the user proceeds to checkout, the Briqpay session is initiated. This is configured in the module's layout XML, which includes scripts to load Briqpay's JavaScript library and the Briqpay_Payments custom JavaScript file. The configuration sets up the checkout steps and the Briqpay payment method renderer.
2. **Rendering the Iframe**
    - **Iframe Rendering:** Once the session is created, Briqpay returns an iframe that is embedded in the checkout page. This iframe handles presentation of payment methods as well as the buy button.
3. **Purchase Decision**
    - **User Actions:** When the user clicks the "buy" button, the Briqpay Decision is triggered. This step allows you to decide whether to proceed with the purchase or not, based on Briqpay's evaluation as well as potential validation performed by the observer.
4. **Order Creation**
    - **Controller Redirection:** If the purchase decision is approved, the user is redirected to a custom Briqpay controller (order/success). This controller handles the creation of the order in Magento.
    - **Order Status:** Initially, the order is created with a status of "pending."
5. **Order Success Page**
    - **Redirection:** After the order is successfully created, the user is redirected to the Magento checkout/onepage/success page, confirming the order placement.
6. **Order Status Update**
    - **Webhook Notification:** Briqpay sends a webhook to update the order status from "pending" to "processing" once the payment is confirmed.
    - **Processing Status:** An order in the "processing" status indicates that it is ready for further handling, such as fulfillment or shipping.

**Order Management Features**

- **Cancel Order:** Orders in the "processing" state can be canceled, which will cancel the entire order.
- **Invoice Creation:** Generating an invoice triggers a capture call to Briqpay. Both full and partial captures are supported.
- **Credit Memo Creation:** Creating a credit memo triggers a refund call to Briqpay. The extension supports both full and partial refunds.

# Detailed description

## 1) Creating a session




Creating a payment session with Briqpay involves multiple components and steps, including configuring layout XML, JavaScript components, and backend controllers in Magento 2. This documentation outlines the process sequentially, focusing on the technical flow and the interaction between various components.

### *1. Frontend Layout Configuration*

The layout configuration is defined in checkout_index_index.xml. This XML file specifies the inclusion of necessary JavaScript files and sets up the structure for the payment component in the Magento checkout process.

- **Script Inclusions:** The Briqpay JavaScript library is included directly from an external source, and a local JavaScript file (briqpay.js) is included from the Briqpay_Payments module.
- **Checkout Structure:** The layout XML extends the default Magento checkout structure to include a new payment method renderer for Briqpay.

### *2. JavaScript Component*

The JavaScript logic for handling the Briqpay payment method is defined in briqpay.js. This file sets up the component, handles initial loading, and observes changes in the checkout process.

- **Component Initialization:** The component is initialized with a default template and an observable message for loading states.
- **Session Loading:** An AJAX request is made to the server to create or fetch a Briqpay session. The response updates the observable message with the session's HTML snippet.
- **Quote Change Handling:** The component observes changes in the shipping and billing addresses and the totals object. If the totals change, the Briqpay session is temporarily suspended and then resumed after fetching updated session information.

### *3. Backend Controller*

The backend logic for creating and managing Briqpay sessions is handled by a Magento controller in Session.php. This controller is responsible for creating a new session or reinitializing an existing one.

- **Session Management:** The controller checks if a session ID exists in the user's session. If it does, it reinitializes the session; otherwise, it creates a new session.
- **Response Handling:** The controller returns a JSON response containing the HTML snippet for the Briqpay session. If an error occurs, an error message is returned instead.

## *4. Session Creation Logic*

The logic for creating a Briqpay session is encapsulated in the CreateSession.php class. This class prepares the necessary data and interacts with the Briqpay API to create a session.

- **Configuration and Cart Data:** The class gathers setup configuration, cart details, billing and shipping data, and quote information.
- **API Request:** A request body is constructed with the gathered data, including dynamically generated webhook URLs. The request is sent to the Briqpay API to create a session.
- **Event Dispatching:** An event is dispatched to allow other components to modify the request body before it is sent.
- **Response Handling:** The response from the API is processed, and the session ID is stored in the quote.

## *5. Utility Classes*

Utility classes in the Briqpay integration serve to prepare and provide necessary data for creating a payment session. They ensure that the correct configuration, cart details, and address information are formatted and sent to the Briqpay API. Here's a detailed explanation of their purposes and impacts on the system:

***SetupConfig generation***

***Purpose:*** Provides essential configuration data needed to initialize a Briqpay session.

***Impact on the System:***

- Ensures that the session setup aligns with the locale, country, and currency settings of the Magento store.
- Generates and supplies URLs for redirecting customers after payment and for webhooks (notifications from Briqpay).

***Key Configuration Data:***

- **Locale:** The language and regional settings.
- **Country:** The country code where the store operates.
- **Currency:** The currency code used for transactions.
- **Redirect URLs:** URLs to redirect customers upon completing or canceling the payment.
- **Webhook URLs:** URLs for receiving asynchronous updates from Briqpay.

***Cart generation***

***Purpose:*** Generates and structures the cart data required by the Briqpay API.

***Impact on the System:***

- Transforms the Magento quote data into a format that Briqpay can understand, ensuring accurate representation of the cart contents and totals.

***Key Cart Data:***

- **Item Details:** Includes product name, SKU, quantity, price, and tax information.
- **Totals:** Calculates and includes the subtotal, discounts, shipping costs, taxes, and grand total.
- **Discounts:** Includes any applied discounts or promotions.
- **Shipping:** Adds shipping costs as a separate line item if applicable.

***Address Collection***

***Purpose:*** Provides and formats the billing and shipping address information for the Briqpay API.

***Impact on the System:***

- Ensures that the address data from the Magento quote is accurately transferred to Briqpay, facilitating proper billing and delivery compliance with address requirements.

***Key Address Fields:***

- **Billing Address Fields:**
    - **First Name:** The customer's first name.
    - **Last Name:** The customer's last name.
    - **Street:** The street address, including house number and street name.
    - **City:** The city where the customer resides.
    - **Postal Code:** The postal code for the billing address.
    - **Country:** The country code for the billing address.
    - **Email:** The customer's email address.
    - **Phone Number:** The customer's contact number.
- **Shipping Address Fields:**
    - **First Name:** The recipient's first name.
    - **Last Name:** The recipient's last name.
    - **Street:** The street address, including house number and street name.
    - **City:** The city where the recipient resides.
    - **Postal Code:** The postal code for the shipping address.
    - **Country:** The country code for the shipping address.
    - **Phone Number:** The recipient's contact number.

## *5. Important information regarding session creation*

### *Using Briqpay for B2B*

If you are planning to use the Briqpay Payment module for selling to businesses (B2B), you need to use the observer to provide the business specific data. This is shown in the "CreateSessionObserverExample".

Expected business data to be passed:

- **CIN:** Company identification number (this could be a VATnumber or Organization number)
- **Name:** The name of the company

### *Using Briqpay for Consumer sales:*

If you are planning to use the Briqpay Payment module for selling to consumers (B2C), you may need to send more information, this is dependent on what payment methods you are planning to use. This is shown in the "CreateSessionObserverExample".

Consumer specific data:

- **identificationNumber:** The consumers identification number, this could be SSN or PNO (this will be country specific but gathered generically in this param)
- **dateOfBirth:** Date of birth
- **name:** The name of the consumer

For additional fields that can be passed visit the Briqpay developer page.

# 2) Order creation

In the Briqpay integration, creating an order can be achieved through two primary methods. Understanding these methods is crucial to ensure a seamless checkout process and to handle any potential issues that may arise during the transaction.

## *Method 1: Customer-Initiated Order Creation*

The first and expected way for an order to be created is through user interaction with the Briqpay iFrame during the checkout process.

1. **User Presses Buy:**
    - When the customer completes their purchase by pressing the "Buy" button within the Briqpay iFrame, the payment process is finalized.
2. **Redirection to Success URL:**
    - After a successful payment, the Briqpay iFrame will automatically redirect the customer to the URL: /checkout/order/success/.
3. **Order Creation in Magento:**
    - At this point, the order is created within the Magento system. All the necessary order details, including customer information, cart contents, and payment confirmation, are processed to generate a new order entry in Magento.
4. **Redirection to Confirmation Page:**
    - Upon successful order creation, the customer is then redirected to the default Magento order confirmation page: /checkout/onepage/success/. This page confirms that the order has been successfully placed and provides the user with order details and a summary of their purchase.

This method ensures a seamless and straightforward experience for the customer, with real-time feedback and immediate confirmation of their order.

## *Method 2: Webhook-Initiated Order Creation*

In some cases, the expected customer-initiated order creation process might not complete as intended. This can occur due to various reasons such as network issues, user navigation errors, or other unforeseen interruptions. To handle such scenarios, Briqpay provides a fallback mechanism using webhooks.

1. **Failure to Redirect:**
    - If the customer does not get redirected to /checkout/order/success/ after completing their payment in the Briqpay iFrame, or if an error occurs during this redirection, the order might not be created immediately in Magento.
2. **Async Flow with Webhooks:**
    - Briqpay's webhook system comes into play to recover from such disruptions. Webhooks are automated messages sent from Briqpay to the Magento backend, containing information about the payment status and transaction details.
3. **Automatic Order Creation:**
    - Upon receiving a webhook notification, Magento processes the data and creates the order asynchronously. This ensures that even if the user's front-end experience is interrupted, the backend order creation process can still complete successfully.
4. **Recovery of User Experience:**
    - The use of webhooks helps in maintaining the integrity of the order creation process, ensuring that valid transactions result in order creation within Magento, even without direct user intervention.

This asynchronous flow provides a robust mechanism to handle edge cases and ensure that all successful transactions are appropriately processed into orders, maintaining a consistent and reliable checkout experience for customers.

# 3) Webhook management

Webhook management is a critical component in the integration of Briqpay with Magento 2, enabling seamless and automated handling of various order statuses and payment confirmations. Briqpay employs webhooks for multifunctional use, ensuring that the order lifecycle is accurately reflected within the Magento system.

## Webhooks Overview

Briqpay utilizes webhooks to communicate important events and updates related to order statuses. These webhooks are essential for maintaining synchronization between Briqpay and Magento, providing real-time updates on order and payment statuses. Below, we outline the primary webhook events and their implications for order management within Magento.

### *1. Order Pending Webhook*

Purpose:

- To notify Magento that an order has been placed but not yet confirmed by the selected Payment Service Provider (PSP).

Process:

- When an order is created, Briqpay sends a POST request to `/briqpay/webhooks/` with the order status `order_pending`.
- This indicates that the order is in a pending state, awaiting confirmation from the PSP.

Impact on Magento:

- The order is reflected in the Magento backoffice as `pending`.
- This webhook also handles the potential asynchronous order creation described in the previous section, ensuring that even if the user experience is disrupted, the order is created and logged in Magento.

### *2. Order Approved Not Captured Webhook*

Purpose:

- To notify Magento that the order has been confirmed by the selected PSP, but the payment has not yet been captured.

Process:

- Upon confirmation from the PSP, Briqpay sends a POST request to `/briqpay/webhooks/` with the status `order_approved_not_captured`.
- This triggers the system to update the order status in Magento.

Impact on Magento:

- The order status moves from `pending` to `processing`.
- The Briqpay custom `transaction` field in the Magento backoffice under the order is updated to `approved`.
- This update signals to the merchant that the order is ready for further action, such as capturing the payment or potentially canceling the order.

### *3. Capture Handling Webhook*

Purpose:

- To update the invoice status to paid, indicating that the payment settlement has been completed.

Process:

- When the payment provider confirms that the settlement of the payment has been completed, Briqpay sends a specific POST request to `/briqpay/webhooks/` for capture handling.

Impact on Magento:

- The invoice status is updated to `paid` in Magento.
- This ensures that the financial status of the order is accurately reflected, allowing the merchant to proceed with order fulfillment with the assurance that payment has been received.

# 4) Order Management

Order management in the Briqpay integration involves handling various operations such as canceling, capturing, and refunding orders. These operations ensure that merchants can efficiently manage their orders directly from the Magento 2 backoffice while keeping the payment status synchronized with Briqpay. The order management process is broken down into three key operations:

## *1 Cancel an Order*

Once an order reaches the "processing" status in the Magento 2 backoffice, the merchant has the option to cancel the order. This action triggers a cancel request to Briqpay, which subsequently triggers a cancel call to the payment provider. The key points are as follows:

- Trigger: The merchant initiates a cancel action from the Magento 2 backoffice when the order is in "processing" status.
- Briqpay Interaction: A cancel request is sent to Briqpay, which then attempts to cancel the transaction with the payment provider.
- Graceful Failure: If the payment provider does not support cancellations, Briqpay will fail gracefully. Despite the failure, the Magento 2 system will still cancel the order.
- Limitation: Canceling an order is only possible before any captures are performed. The order must be in "processing" status for the cancel call to be successful.

## *2 Capture an Order*

When a user creates an invoice in the Magento 2 backoffice, this triggers a capture request to Briqpay. The extension supports both full and partial captures, but the support for partial captures may vary depending on the payment provider. Key points include:

- Trigger: The merchant creates an invoice in the Magento 2 backoffice.
- Briqpay Interaction: A capture request is sent to Briqpay. The extension supports both full and partial captures.
- Partial Captures: Support for partial captures depends on the payment providers integrated with Briqpay. Merchants should consult with Briqpay to confirm if their payment mix supports partial captures.
- Webhook Update: Once the capture is completed at the payment provider, Briqpay sends a webhook to the Magento 2 system. The invoice status is then updated to "paid" in Magento.

### *3 Refund an Order*

After an invoice has been created in the Magento 2 system, it is possible to create a credit memo. This action triggers a refund request to Briqpay. Similar to captures, Briqpay supports both full and partial refunds. Key considerations include:

- Trigger: The merchant creates a credit memo in the Magento 2 backoffice.
- Briqpay Interaction: A refund request is sent to Briqpay. The extension supports both full and partial refunds.
- Partial Refunds: Support for partial refunds depends on the payment providers integrated with Briqpay. Merchants should consult with Briqpay to confirm if their payment mix supports partial refunds.

These operations ensure that the order management process remains seamless and synchronized between Magento 2 and Briqpay, providing merchants with a reliable and efficient workflow.

# Events and Observers

In the Briqpay integration with Magento 2, events and observers play a crucial role in providing flexibility and customization. These mechanisms allow developers to extend and modify the default behavior of the payment module. The following is a conceptual overview of the different events and observers available in the Briqpay integration.

## 1. CreateSessionObserverExample

Event: `briqpay_payment_module_body_prepare`

Purpose: The `CreateSessionObserverExample` provides an example of how to use the `briqpay_payment_module_body_prepare` event. This event allows developers to add more data or modify the payload before it is sent to Briqpay for session creation. By leveraging this event, developers have full control over the create session call, enabling them to utilize the complete functionality of Briqpay.

Use Case:

- Adding custom fields or data to the create session payload.
- Modifying existing data to meet specific business requirements.
- Ensuring that all necessary information is included in the session creation process.

For more information on what data can be sent to Briqpay, refer to the Briqpay Create Session API Documentation.

## 2. DecisionObserverExample

Event: `briqpay_payment_module_decision_prepare`

**Purpose:** The `DecisionObserverExample` allows developers to perform validations before accepting a payment. This event is crucial for merchants who need to ensure that certain conditions are met before a transaction is completed. If the developer returns `true`, the validation passes, and the payment process continues. If `false`, the transaction is stopped after the customer has pressed "buy" in the checkout.

**Use Case:**

- Stock validation to ensure items are available.
- Shipping validation to confirm delivery options are correct.
- Additional custom validations required by the business.

Standard validations performed by Briqpay include:

- Ensuring billing and shipping data match between the quote and the Briqpay session.
- Verifying that the cart contents are the same between the quote and the Briqpay session.

## 3. UpdateSessionObserverExample

Event: `briqpay_payment_module_update_prepare`

**Purpose:** The `UpdateSessionObserverExample` is triggered when an ongoing order is being updated. Similar to the create session event, this allows developers to provide additional data or modify the payload when a session update occurs. This ensures that any changes in the order are accurately reflected in the Briqpay session.

**Use Case:**

- Updating session data with new shipping information.
- Modifying cart contents if changes occur after the initial session creation.
- Adding additional information required for the updated session.

For more details on what can be modified during an update, refer to the Briqpay Update Session API Documentation.

### 4. CustomPaymentMethodName

Event: `email_order_set_template_vars_before`

Purpose: The `CustomPaymentMethodName` observer overrides the payment method name in the email sent to users after a purchase. Instead of displaying "Briqpay," it replaces it with the name of the selected payment provider. This helps avoid confusion, as Briqpay is an integration layer, not a payment method.

Use Case:

- Ensuring that the payment method name in the order confirmation email matches the user's selected payment provider.
- Enhancing user experience by providing clear and accurate information.

# Testing on localhost

If you are planning to develop on localhost you will need to handle the webhook management manually as they require a public endpoint. In the following section we will provide some tips and tricks on how this can be achieved.

## 1 Receive webhooks on localhost

To receive webhooks on your localhost you will need to use an external service that can receive the webhooks and display its content. We suggest that you use a free service such as https://webhook.site/ that allows you to set up a public endpoint and record the payload of the webhook.

Once you have received the public url, you can manually set the url in Briqpay/Payments/Model/Config/SetupConfig on line 83.

**Note:** Do not forget revert this change before going live

## 2 Manually send webhooks on localhost

Once you have received a webhook and would like to manually trigger the webhook towards your local magento 2 installation you can use the following curl command.

```
curl -X POST http://127.0.0.1:8080/briqpay/webhooks \
    -H "Content-Type: application/json" \
    -d '{ "event": "order_status", "status": "replace_with_session_status",
"sessionId": "replace_with_session_id", "quoteId": "replace_with_qoute_id" }'
```

*Replace the payload with the payload from the webhook. Don't forget to change the URL to your localhost url.*