Installation – (HTML/JavaScript)

This page assumes that you have completed the initial setup and read the Widgets Overview .

As explained in "Widget Overview," depending on the payment method selected, an "application" may be made rather than a "payment." Therefore, on this page, we refer to "payment or application" made through the widget as "processing."

Install with HTML

To place a widget in HTML, use a single <script> tag and a single <form> tag.

Within the <form> tag, parameters (various fields and their values) are written using the <span> element.

Since the App Token is a very long and irregular string, we recommend that you write it down in advance on the admin screen or copy it from a generator.

Sample Code

Below is an example of how to set up a widget to pay 1,000 yen with a button that says "Pay."

```html
<script src="https://widget.univapay.com/client/checkout.js"></script>
<form action="<任意のURL>">
 <span data-app-id="<アプリトークンID>"
      data-txt="支払う"
      data-checkout="payment"
      data-amount="1000"
      data-currency="jpy"
      data-auto-submit="true"
 ></span>
</form>
```

Actions after processing is complete

After the process is complete, submit the following to the "<any URL>" specified in form action=.

- Transaction token (`data-checkout-type="token"` if specified `univapayTokenId`)
- Payment number (`data-checkout-type="payment"` if `univapayChargeId` specified) `data-token-type="subscription"` `univapaySubscriptionId`

If you want to control the consumer's screen transition based on the processing results, create a program to obtain these.

As explained in the parameter (basic operation), by using , you can set `data-auto-submit` the form containing the widget to automatically close when the done button is pressed to close the widget . By using the above, you can also transition to the URL specified in the action attribute of the form tag placed outside the span tag. `submit`

Installing with JavaScript

When setting up a widget using Javascript, you can also embed the JavaScript code in an HTML file using the <script> tag to set up the widget.

Sample Code

Below is an example of how to set up a widget to pay 1,000 yen with a button that says "Pay."

```
<script src="https://widget.univapay.com/client/checkout.js"></script>

<form id="payment-form" action="https://docs.univapay.com/docs/guide/implement/widge
  <button id="univapay-payment-checkout">
   支払う
  </button>
</form>

<script>
  var checkout = UnivapayCheckout.create({
    appId: "<TOKEN>",
    checkout: "payment",
    amount: 100,
    currency: "jpy",
    cvvAuthorize: true,

    onSuccess: function () {
      var form = document.querySelector("#payment-form");
      form.submit();
    }
  });

  var button = document.querySelector("#univapay-payment-checkout");
  button.onclick = function () {
    checkout.open();
  };

</script>
```

First, the page containing the widget must include the following line, as set in HTML:

```
<script src="https://widget.univapay.com/client/checkout.js"></script>
```

Unlike when setting up in HTML, a button is not automatically created on the page, so you need to write the code to submit using the <form> tag and <button> tag.
When the submit event is fired, it will transition to the URL specified in the action.

```
<form id="payment-form" action="https://docs.univapay.com/docs/guide/implement/widge
  <button id="univapay-payment-checkout">
   支払う
  </button>
</form>

<script>
<!--
タグ内にJavaScriptの記述をします
内容は以下で説明
-->
</script>
```

Next, write a call to UnivapayCheckout.create.

For example, to create a widget object that will process a 1000 yen payment, you need to write the following:

```
var checkout = UnivapayCheckout.create({
  appId: "<TOKEN>",
  checkout: "payment",
  amount: 1000,
  currency: "jpy",
```

Next, write the submit process to transition to the URL specified in the action.

This time, we will use onSuccess to write a code that will submit the form when the process is successful.

```
    onSuccess: function () {
      var form = document.querySelector("#payment-form");
      form.submit();
    }
  });
```

Finally, write the onclick function of the <button> tag. Open the widget by calling the open function on the object returned by UnivapayCheckout.create.

```
  var button = document.querySelector("#univapay-payment-checkout");
  button.onclick = function () {
    checkout.open();
  };
```

Parameters that control the widget design

The widget design can be changed using the following parameters.

For details on other operations such as what kind of processing will be performed, please refer to the various parameters from the left menu and implement them.

About each field name

- `data-` Field names written in kebab-case are for HTML.
- The field name written in lower camel case on the second line is for JavaScript.
- If only one is listed, you can use only one of them.

field	Acceptable values	remarks	Default value
data-text	Text (unlimited)	Text in the button to expand the widget	Pay with UnivaPay
data-size	Half-width English	Button size for expanding the widget. Possible values: `small` , `normal` , `large`	
data-class	Half-width alphanumeric characters	The value to assign to the class attribute of the button to expand the widget. Can be written on the website. `style`	
data-style	Half-width alphanumeric characters	Value assigned to the style attribute of the button for expanding the widget. Used when you want to write styles inline without applying the style from the website . `style`	
data-hover-style	Half-width alphanumeric characters	Style for the button to expand the widget when hovered (inline description)	

field	Acceptable values	remarks	Default value
data-header header	Text (unlimited)	The text you want to display in the widget header	Service name
data-title title	Text (unlimited)	Store name you want to display in the widget subheader	Depends on the settings in the admin panel
data-description description	Text (unlimited)	The text you want to display under the store name in the widget subheader.	Depends on the settings in the admin panel
data-submit-button-text submitButtonText	Text (unlimited)	The text you want to display instead of the "Pay" button on the widget	pay

Please see the disclaimer for description rules

Webhooks

The processing result will be POSTed as JSON data to the URL specified in the "Webhook" section of the admin panel.

Please refer to "Reference > Webhooks" to create a script that retrieves the POSTed data and updates the order status, etc.

Callbacks

By defining the following function in the source code within the widget tag, a notification including the content of each event will be sent when it occurs.

If you want to control the display content of the original page or the transition destination depending on the processing result, use this function or the value submitted from the form tag.

event	Content
Opened (**univapay:opened**)	An event triggered by the opening of the payment form. It does not contain any other information.
Closed (**univapay:closed**)	This event is triggered when the payment form is closed. It does not contain any other information.
Success (**univapay:success**)	This event is raised when the payment form has been processed. It contains the IDs of the resources created and details about the transaction.
Error (**univapay:error**)	This event occurs when the processing fails at any stage and contains the ID of the resource that was created.
Token created (**univapay:token-created**)	This event occurs when a transaction token is created, and contains detailed information about the token. This event does not occur when making a payment using an already created transaction token, such as for periodic or recurring charges.
Charge created (**univapay:charge-created**)	This event occurs when a charge is created, and contains the details of the charge. It occurs regardless of whether the charge is successful.
Subscription created (**univapay:subscription-created**)	This event occurs when a recurring charge is created, and contains details about the recurring charge. It occurs regardless of whether the charge was successful.
Validation error (**univapay:validation-error**)	This event occurs if there are errors in each field. If there are no errors or if each field has not yet been selected, the object will be blank.

event	Content
3DS Start (**univapay:three-ds-authorization**)	This event occurs when 3-D Secure authentication is initiated. It does not include any other information.
3DS Open (**univapay:three-ds-authorization-modal-opened**)	This event occurs when a popup appears on the form when 3-D Secure authentication is performed. It does not contain any other information.
3DS Success (**univapay:three-ds-authorization-success**)	This event occurs when 3-D Secure authentication is successful. It contains the target resource ID and detailed processing information. It does not notify you that payment has been successful.
3DS Failed (**univapay:three-ds-authorization-failure**)	This event occurs when 3-D Secure authentication fails. It includes the ID of the target resource and details of the operation.

Callback Sample

```
<script>
    window.addEventListener("univapay:success", (e) => { console.info("Success event
    window.addEventListener("univapay:token-created", (e) => { console.info("Token e
    window.addEventListener("univapay:charge-created", (e) => { console.info("Charge
    window.addEventListener("univapay:subscription-created", (e) => { console.info("
    window.addEventListener("univapay:validation-error", (e) => { console.error("Val
</script>
```