Creating a custom payment form with Components requires four steps:

1. Set up Monri Components
2. Create your payment form
3. Create a token to securely transmit card information
4. Submit the token and the rest of your form to your server

HTTPS requirements

All submissions of payment info using Components are made via a secure HTTPS connection. However, to protect yourself from certain forms of man-in-the-middle attacks, and to prevent your customers from seeing Mixed Content warnings in modern browsers, you must serve the page containing the payment form over HTTPS as well.

In short, the address of the page containing Components must start with **https://** rather than just **http://**. If you are not familiar with the process of buying SSL certificates and integrating them with your server to enable a secure HTTPS connection, check out our [security](security link) documentation for more information.

Step 1: Set up Monri Components

Components is available as part of Monri.js. To get started, include the following script on your pages. This script must always load directly from **monri.com** in order to remain PCI compliant—you can't include it in a bundle or host a copy of it yourself.

```
<script src="https://ipgtest.monri.com/dist/components.js"></script>
```

Next, create an instance of Components:

```
var monri = Monri('authenticity-token');
var components = monri.components("random-token", "digest", 'timestamp');
```

Replace `authenticity-token` with value provided in merchant dashboard.
Timestamp should be valid ISO date string.
Generate `random-token` as random UUID, digest is calculated as `digest = SHA512(#{merchant.key}#{random-token}#{timestamp})`.

When you're ready to accept live card payments, replace the test authenticity token and merchant key with your production authenticity token and merchant key.

Step 2: Create your payment form

To securely collect card details from your customers, Components creates UI components for you that are hosted by Monri. They are then placed into your payment form, rather than you creating them directly.

To determine where to insert these components, create empty DOM elements (containers) with unique IDs within your payment form. We recommend placing your container within a `<label>` or next to a `<label>` with a `for` attribute that matches the unique `id` of the Element container. By doing so, the Element automatically gains focus when the customer clicks on the corresponding label.

For example:

```
<form action="/charge" method="post" id="payment-form">
  <div class="form-row">
    <label for="card-element">
      Credit or debit card
    </label>
    <div id="card-element">
      <!-- A Monri Component will be inserted here. -->
    </div>

    <!-- Used to display Component errors. -->
    <div id="card-errors" role="alert"></div>
  </div>

  <button>Submit Payment</button>
</form>
```

When the form above has loaded, create an instance of an Component and mount it to the Component container created above:

```
// Custom styling can be passed to options when creating an Component.
var style = {
  base: {
    // Add your base input styles here. For example:
    fontSize: '16px',
    color: "#663399",
  }
};
// Create an instance of the card Component.
var card = components.create('card', {style: style});
// Add an instance of the card Component into the `card-element` <div>.
card.mount('#card-element');
```

The `card` Component simplifies the form and minimizes the number of fields required by inserting a single, flexible input field that securely collects all necessary card details.

Components validates user input as it is typed. To help your customers catch mistakes, you should listen to `change` events on the `card` Component and display any errors:

```
card.onChange(function(event) {
  var displayError = document.getElementById('card-errors');
  if (event.error) {
    displayError.textContent = event.error.message;
  } else {
    displayError.textContent = '';
  }
});
```

Step 3: Create a token to securely transmit card information

The payment details collected using Components can then be converted into a token. Create an event handler that handles the `submit` event on the form. The handler sends the fields to Monri for tokenization and prevents the form's submission (the form is submitted by JavaScript in the next step).

```
// Create a token or display an error when the form is submitted.
var form = document.getElementById('payment-form');
form.addEventListener('submit', function(event) {
  event.preventDefault();
  monri.createToken(card).then(function(result) {
    if (result.error) {
      // Inform the customer that there was an error.
      var errorElement = document.getElementById('card-errors');
      errorElement.textContent = result.error.message;
    } else {
      // Send the token to your server.
      monriTokenHandler(result.token);
    }
  });
});
```

Call createToken and pass it the card component.
TBD: The function also accepts an optional second parameter containing additional payment information collected from the customer, which is not used in this example.
The function returns a `Promise` which resolves with a `result` object. This object has either:

- `result.token` : a Token was created successfully.
- `result.error` : there was an error. This includes client-side validation errors. Refer to the API reference for all possible errors.

Step 4: Submit the token and the rest of your form to your server

The last step is to submit the token, along with any additional information that has been collected, to your server.

```
function monriTokenHandler(token) {
  // Insert the token ID into the form so it gets submitted to the server
  var form = document.getElementById('payment-form');
  var hiddenInput = document.createElement('input');
  hiddenInput.setAttribute('type', 'hidden');
  hiddenInput.setAttribute('name', 'monriToken');
  hiddenInput.setAttribute('value', token.id);
  form.appendChild(hiddenInput);
  // Submit the form
  form.submit();
}
```

Viewport meta tag requirements

In order to provide a great user experience for 3D Secure on all devices, you should set your page's viewport `width` to `device-width` with the the viewport meta tag. There are several other viewport settings, and you can configure those based on your needs. Just make sure you include `width=device-width` somewhere in your configuration.

For instance the following will work with Components:

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

If you already have this tag on your page and it includes `width=device-width`, then you are all set.

Components

All Components accept a common set of options, and then some Component-specific options.

- style - customize appearance using CSS properties. Style is specified as an object for any of the variants below.
  – `base`, base style—all other variants inherit from this style
  – `complete`, applied when the Element has valid input
  – `empty`, applied when the Element has no customer input
  – `invalid`, applied when the Element has invalid input
  For each of the above, the properties below can be customized.
  – `fontSize`
  – `color`
  – `fontFamily`
  – `fontSmoothing`
  – `fontVariant`
  – `fontWeight`
  – `letterSpacing`
  – `textDecoration`
  – `textShadow`
  – `textTransform`
  – `border`

Backend integration

Example, php, yii2 framework, action for rendering checkout form:

```
// actionMonriComponents
// authenticity_token => merchant dashboard
// key => merchant dashboard
$sec = new Security();
$token = $sec->generateRandomString(10);
$timestamp = (new \DateTime())->format('c');
return $this->renderPartial("monri-components", [
  'authenticity_token' => $authenticity_token,
  'token' => $token,
  'digest' => hash('sha512', $key . $token . $timestamp),
  'timestamp' => $timestamp
]);
```

Content of `monri-components.php` file

```
<form action="" method="post" id="payment-form">
 <div class="form-row">
 <label for="card-element">
  Credit or debit card
        </label>
 <div id="card-element">
  <!-- A Monri Component will be inserted here. -->
  </div>

  <!-- Used to display Element errors. -->
  <div id="card-errors" role="alert"></div>
 </div>
 <button>Submit Payment</button>
</form>

<footer>
 <script src="https://ipgtest.monri.com/dist/components.js"></script>
 <script>  var monri = Monri("<?= $authenticity_token ?>");
 var components = monri.components("<?= $token ?>", "<?= $digest ?>", "<?=

 var card = components.create("card", {
            style: {}
        });

  card.mount("card-element");

  window.card = card;

 var form = document.getElementById('payment-form');
  form.addEventListener('submit', function (event) {
            event.preventDefault();
```

```
   monri.createToken(card).then(function (result) {
                 if (result.error) {
                     // Inform the customer that there was an error.
   var errorElement = document.getElementById('card-errors');
   errorElement.textContent = result.error.message;
   } else {
                     // Send the token to your server.
   monriTokenHandler(result.result);
   }
             });
   });

  function monriTokenHandler(token) {
             // Insert the token ID into the form so it gets submitted to th
   var form = document.getElementById('payment-form');
  var hiddenInput = document.createElement('input');
   hiddenInput.setAttribute('type', 'hidden');
   hiddenInput.setAttribute('name', 'monriToken');
   hiddenInput.setAttribute('value', token.id);
   form.appendChild(hiddenInput);

   form.submit();
   }

     </script>
 </footer>
```

After successful card tokenization `monriToken` is submitted to your server with rest of your form.

Next steps

Congrats! You now have a custom payment form to accept card payments with Monri. Once you've sent your form to your server, you'll be able to use the token to perform a charge or to save to a customer.

Authorize example

Example of action handling form submit:

```php
function transactionData($transactionType) {
    $sec = new Security();
    $amount = '100';
    $currency = 'EUR';
    $order_number = "monri-components" . $sec->generateRandomString(10);
    //digest = SHA512(key + order_number + amount + currency)
    $digest = hash('sha512', $key . $order_number .     $amount . $currency
    return  [
        "transaction_type" => $transactionType,
        "amount" => $amount,
        "ip" => '10.1.10.111',
        'order_info' => 'Monri components trx',
        'ch_address' => 'Adresa',
        'ch_city' => 'Grad',
        'ch_country' => 'BIH',
        'ch_email' => 'test@test.com',
        'ch_full_name' => 'Test',
        'ch_phone' => '061 000 000',
        'ch_zip' => '71000',
        'currency' => $currency,
        'digest' => $digest,
        'order_number' => $order_number,
        'authenticity_token' => $authenticity_token,
        'language' => 'en',
        // This part is important! Extract monriToken from post body
        'temp_card_id' => Yii::$app->request->post('monriToken')
     ];
}

function transaction($url, $data) {
    $data_string = Json::encode(['transaction' => $data]);
// Execute transaction
    $ch = curl_init($url . './v2/transaction');
      curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
      curl_setopt($ch, CURLOPT_POSTFIELDS, $data_string);
      curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
      curl_setopt($ch, CURLOPT_HTTPHEADER, array(
      'Content-Type: application/json',
      'Content-Length: ' . strlen($data_string))
     );
    // TODO: handle transaction result
    $result = curl_exec($ch);
    return $result;
}
```

```
function transactionExample() {
    $url = 'https://ipgtest.monri.com'; // change for production env
    // Prepare transaction payload, include monriToken as `temp-card-id` fi
    $data = transactionData('authorize');
    return transaction($url, $data);
}
```

Purchase example

```
function transactionExample() {
    $url = 'https://ipgtest.monri.com'; // change for production env
    // Prepare transaction payload, include monriToken as `temp-card-id` fi
    $data = transactionData('purchase');
    return transaction($url, $data);
}
```

Purchase + tokenize card example

```
function transactionExample() {
    $url = 'https://ipgtest.monri.com'; // change for production env
    // Prepare transaction payload, include monriToken as `temp-card-id` fi
    $data = transactionData('purchase');
    // Add tokenize-pan-until field to `$data`
    $data['tokenize_pan_until'] = '2110'; // YYMM
    return transaction($url, $data);
}
```