

# Monetra® Payment Software

---

## PaymentFrame Guide

Revision: 2.0.0

Publication date April 27, 2022

# PaymentFrame Guide

Monetra Technologies, LLC

Revision: 2.0.0

Publication date April 27, 2022

Copyright © 2022 Monetra Technologies, LLC

## Legal Notice

The information contained herein is provided *As Is* without warranty of any kind, express or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. There is no warranty that the information or the use thereof does not infringe a patent, trademark, copyright, or trade secret.

Monetra Technologies, LLC. SHALL NOT BE LIABLE FOR ANY DIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, WHETHER RESULTING FROM BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, OR OTHERWISE, EVEN IF MONETRA TECHNOLOGIES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. MONETRA TECHNOLOGIES RESERVES THE RIGHT TO MAKE CHANGES TO THE INFORMATION CONTAINED HEREIN AT ANYTIME WITHOUT NOTICE. NO PART OF THIS DOCUMENT MAY BE REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS, ELECTRONIC OR MECHANICAL, FOR ANY PURPOSE, WITHOUT THE EXPRESS WRITTEN PERMISSION OF Monetra Technologies, LLC.

# Table of Contents

- 1. Revision History ..... 1
- 2. Monetra PaymentFrame ..... 2
  - 2.1. Introduction ..... 2
  - 2.2. How It Works ..... 2
  - 2.3. HMAC Parameters ..... 3
  - 2.4. Styling the Payment Form ..... 5
  - 2.5. Using an External Submit Button ..... 8
  - 2.6. Enabling the Iframe Submit Button ..... 8
  - 2.7. Code Example (PHP) ..... 9

# 1 Revision History

Version	Date	Changes
v1.0	2017-08-30	<ul style="list-style-type: none"><li>Initial document.</li></ul>
v1.1	2017-10-31	<ul style="list-style-type: none"><li>Documented the ability to split cardholder name field into first and last.</li></ul>
v1.2	2018-02-12	<ul style="list-style-type: none"><li>This update applies to the "HMAC Parameters" section. It enhances the description for the 'hmac-css-url' parameter and adds the three new ones below:<ul style="list-style-type: none"><li>* hmac-include-cardholdername</li><li>* hmac-include-street</li><li>* hmac-include-zip</li></ul></li></ul>
v1.3	2018-07-18	<ul style="list-style-type: none"><li>This update applies to the "HMAC Parameters" section. It describes how to use the new 'auto-load' feature.</li></ul>
v1.4	2018-10-04	<ul style="list-style-type: none"><li>This update applies to the "HMAC Parameters" section. It describes how to use the new 'autocomplete' feature.</li></ul>
v1.5	2019-10-11	<ul style="list-style-type: none"><li>This update describes how to let Monetra/TranSafe host the javascript files.</li></ul>
v1.6	2019-12-09	<ul style="list-style-type: none"><li>Support for suppressing the submit button to allow a submit button external to the iframe to submit.</li></ul>
v1.6.1	2020-04-02	<ul style="list-style-type: none"><li>Explicitly note that only HTTPS is supported.</li></ul>
v1.7.0	2021-08-02	<ul style="list-style-type: none"><li>Add ACH support.</li></ul>
v2.0.0	2022-04-27	<ul style="list-style-type: none"><li>Authentication now uses API Keys</li></ul>

## 2 Monetra PaymentFrame

2.1. Introduction .....	2
2.2. How It Works .....	2
2.3. HMAC Parameters .....	3
2.4. Styling the Payment Form .....	5
2.5. Using an External Submit Button .....	8
2.6. Enabling the Iframe Submit Button .....	8
2.7. Code Example (PHP) .....	9

### 2.1 Introduction

PaymentFrame is a Monetra feature that allows you to embed a secure iframe containing a payment form on an HTTPS-enabled web page. This prevents credit card data from ever touching your systems, while allowing your customers to complete seamless ecommerce transactions on your site.



Note: The PaymentFrame can not be embedded into a non-secured HTTP site, even for testing and development purposes, due to content security policies and mixed active content, the browser will silently fail to load the page.



PCI Notice: This approach to Ecommerce integration follows the PCI Council's best practices as outlined in [https://www.pcisecuritystandards.org/pdfs/best\\_practices\\_securing\\_ecommerce.pdf](https://www.pcisecuritystandards.org/pdfs/best_practices_securing_ecommerce.pdf), and qualifies for SAQ-A.

### 2.2 How It Works



Note: In order to use the PaymentFrame, you will need to create a non-expiring merchant profile API Key on the Monetra (or TranSafe) instance you are connecting to. This API Key needs to at a minimum contain the `action_admin=TICKETREQUEST`, however if using the same API Key for the follow-up actions, it must contain any necessary permissions for the follow-up actions.

Examples for PaymentFrame implementation and styling are available <https://www.monetra.com/developers> with a live demo at <https://iframe.test.transafe.com>.

1. *Prepare your credentials:* Before rendering your payment page (the page that will host the iframe), you will need to generate a 256-bit keyed-hash message authentication code (HMAC-SHA256) See Section 2.3: HMAC Parameters.
2. *Add iframe element and components:* Your html payment page must include an empty iframe element (with no `src` attribute specified) with a unique id which will be referenced by the JavaScript where you would like the payment form to appear. The HMAC message components (along with the HMAC itself from step 1) must be included as `data-` attributes on the iframe element. Note: The order in which the data attributes are concatenated for the HMAC message must match the order of the parameters in the table in Section 2.3: HMAC Parameters. The ordering is crucial for Monetra/TranSafe to be able to properly verify the HMAC.

3. *Load script:* Your payment page will need to load `https://{monetra_url}/PaymentFrame/PaymentFrame.js`, which provides the client-side logic for displaying the iframe. For example, if you are using `test.transafe.com` as your payment server, the URL would be `https://test.transafe.com/PaymentFrame/PaymentFrame.js`.
4. *Load Payment Form:* From here you'll just need to write a few lines of your own Javascript to instantiate a `PaymentFrame` object and request the iframe. You will need to define a callback function to be executed after the user's payment information has been submitted. The full sequence would be to instantiate `PaymentFrame` with the `iframe id` and URL for the Monetra Server, define callback using the `.setPaymentSubmittedCallback()` method, and call the `.request()` method to load the `iframe`. You can find an example of this sequence in the code example below.
5. *Submit Cardholder Data:* Once the `iframe` has loaded, it will render a payment form into which the user can enter their payment information. When the user submits the form, their information will be sent directly to the Monetra/TranSafe server, which will return a `CardShield` ticket.
6. *Process response:* At this point, the callback function that you defined in your Javascript will be executed, receiving a JSON object representing the Monetra/TranSafe server's response (including the `CardShield` ticket) as a parameter.
7. *Process the ticket:* Once you have a valid ticket then you can communicate directly with the Monetra/TranSafe system to run a standard transaction such as as `SALE` or a `PRE-AUTH` as defined in the Monetra Application Interface Guide.

## 2.3 HMAC Parameters



Note: Please ensure these `iframe` attributes are prefixed with `data-` when using our standard `PaymentFrame` javascript to display the `iframe` as the javascript library uses this prefix to determine the attributes used by the `PaymentFrame` and sent in the request to the Payment Server.

Legend:

Req = Required or not [Y=Yes C=Conditional O=Optional]

Key	Req	Description
<code>hmac-hmacsha256</code>	Y	Generated HMAC-SHA256 using the API Key as the key. The message must be a concatenation (with no delimiters) of all of the other <code>hmac-</code> values you are providing, in the same order in which they appear in this table (with the exception of this attribute)
<code>hmac-timestamp</code>	Y	Standard Unix timestamp. Must be within 15 minutes of server's time.
<code>hmac-domain</code>	Y	The domain of the page embedding the <code>iframe</code> . This must match exactly what would show in the browser URL bar, including <code>'https://'</code> (and port, if it is explicitly set). Please note that using <code>'http://'</code> will NOT work due to Content Security Policies and Mixed Active Content.

Key	Req	Description
hmac-sequence	Y	Merchant-specified sequence number. May be alphanumeric.
hmac-auth_apikey_id	Y	The API Key ID returned by the Monetra server when creating the merchant profile API Key.
hmac-css-url	O	URL of CSS file the iframe should load. It must be served from the same domain as `hmac-domain`. If no CSS URL is provided, the payment form will be rendered with default styling (see " <a href="#">Styling the Payment Form</a> " for more information).
hmac-include-cardholdername	O	Indicates whether the cardholder name field should be present on the payment form. Value can be "yes" or "no". Defaults to "yes" if not provided. Only provide "no" here if cardholder name is collected elsewhere during your checkout process, and sent along with the ticket in the Monetra sale request.
hmac-include-street	O	Indicates whether the street address field should be present on the payment form. Value can be "yes" or "no". Defaults to "yes" if not provided. Only provide "no" here if street address is collected elsewhere during your checkout process, and sent along with the ticket in the Monetra sale request.
hmac-include-zip	O	Indicates whether the zip code field should be present on the payment form. Value can be "yes" or "no". Defaults to "yes" if not provided. Only provide "no" here if zip code is collected elsewhere during your checkout process, and sent along with the ticket in the Monetra sale request.
hmac-expdate-format	O	Format of expiration date the iframe should use. Choices: * single-text: Free-form text entry. Default if not specified. * separate-selects: Two drop downs for month and year. * coupled-selects: Two drop downs for month and year in a container.
hmac-cardholdername-format	O	Single, or split cardholder name. Choices: * separate-firstlast * combined-firstlast separate-firstlast will split the card holder name field into first and last name fields. This field must be part of the hmac if present. It comes after 'expdate-format' in the hmac data.
hmac-auto-reload	O	Indicates whether checkout page containing payment form should auto-reload every 15 minutes in order to avoid the HMAC becoming invalid due to the time it was generated. Value can be 'yes' or 'no'. Defaults to 'no' if not provided. If 'yes', users will see a message on the

Key	Req	Description
		checkout form (starting approximately 5 minutes before the reload) indicating that the page will be reloaded soon for security purposes. The message will include a timer that counts down the time remaining until the reload occurs.
hmac-autocomplete	O	Indicates whether browser autocomplete should be enabled on the payment form. Value can be 'yes' or 'no'. Defaults to 'yes' if not provided. If 'yes', payment form fields will allow autocomplete (as long as the user's browser settings are configured to allow it, which is generally the default). If 'no', autocomplete will be disabled on the payment form fields.
hmac-include-submit-button	O	Indicates whether the payment form should include a submit button within the iframe. Defaults to 'yes' if not provided. If you would like to use a submit button on your own page outside of the iframe, set this value to 'no'. You would then use Javascript to send a message to the iframe to trigger a form submit. See <a href="#">Section 2.5: Using an External Submit Button</a> .
hmac-ach-enabled	O	Whether to display an ACH payment form in the iframe. Value can be 'yes' or 'no'. Defaults to 'no' if not provided (meaning a credit card payment form will be displayed).
hmac-ach-accounttype	O	Which ACH account type options should be allowed, and the order in which they will appear. Should be provided as a comma-separated list of values. Options are "business", "personal", "checking", and "savings". Options will be displayed to user in two separate `select` elements, one for selecting "business" or "personal" and another for selecting "checking" or "savings". The order of these options (and therefore which options are the default choices) will match the order in which they are provided in this list. If ACH is enabled and this value is not provided, the allowed options will default to "personal"/"business" and "checking/savings". If ACH is not enabled, this value has no effect and should not be provided.



Note: Response Parameters are the same as returned by the Monetra POST Protocol ticket request. The hmac-timestamp and hmac-sequence will be used as part of the POST response to allow the hmac-hmacsha256 response from the ticket request to be validated.

## 2.4 Styling the Payment Form

The `iframe` content can be styled using custom CSS. You can use the classes and IDs documented below in your CSS to style the payment form and the elements it contains.



You can also provide custom text for the form's labels by specifying appropriate CSS rules. Each label contains an empty span element, so that you can use the `::before` selector in conjunction with the `content` property to insert your desired text into the label. For example, in order to provide custom text for the ZIP code label, you would use a CSS rule similar to the following:

```
#payment-zip-label span::before {
  content: "Custom text here";
}
```



Note: If you do not provide custom text for any label, it will default to the value referenced in the "IDs" table below.

CLASSES	
Class	Description
<code>payment-form-label</code>	Applied to all label elements in the payment form. Each of these elements contains an input or select element into which payment data will be entered.
<code>payment-expdate-label</code>	If <code>hmac-expdate-format</code> is set to <code>separate-selects</code> or <code>coupled-selects</code> , this will be applied to the two label elements that contain the select drop-downs for expiration date month and year.
<code>payment-cardholdername-label</code>	If <code>hmac-cardholdername-format</code> is set to <code>separate-firstlast</code> , this will be applied to the two label elements that contain the first and last name text entry fields.
<code>payment-ach-accounttype-container</code>	<code>div</code> element containing a label and <code>select</code> for selecting ACH account type

ID's	
ID	Description
<code>monetra-payment-form</code>	The payment form element.
<code>payment-account-label</code>	The label containing the account (credit card number) input element. Label text defaults to 'Card Number'.
<code>payment-expdate-container</code>	If <code>hmac-expdate-format</code> is set to <code>coupled-selects</code> , this is the <code>div</code> that contains the two select elements.
<code>payment-expmonth-label</code>	If <code>hmac-expdate-format</code> is set to <code>separate-selects</code> or <code>coupled-selects</code> , this is the label that contains the expiration month select element. Label text defaults to 'Expiration Month'.
<code>payment-expyear-label</code>	If <code>hmac-expdate-format</code> is set to <code>separate-selects</code> or <code>coupled-selects</code> , this is the label that contains the expiration year select element. Label text defaults to 'Expiration Year'.
<code>payment-expdate-label</code>	If <code>hmac-expdate-format</code> is set to <code>single-text</code> , this is the label that contains the 'expdate' (expiration date) input element. Label text defaults to 'Expiration Date'.

ID's	
ID	Description
payment-cardholdername-label	The label containing the 'cardholdername' input element. Label text defaults to 'Cardholder Name'.
payment-cardholdernamefirst-label	If hmac-cardholdername-format is set to separate-firstlast, this is the label that contains the first name field. Label text defaults to Cardholder First Name.
payment-cardholdernamelast-label	If hmac-cardholdername-format is set to separate-firstlast, this is the label that contains the last name field. Label text defaults to Cardholder Last Name.
payment-street-label	The label containing the 'street' (street address) input element. Label text defaults to 'Street Address'.
payment-zip-label	The label containing the 'zip' (ZIP code) input element. Label text defaults to 'ZIP Code'.
payment-cv-label	The label containing the 'cv' (card verification value) input element. Label text defaults to 'CV'.
payment-submit-button	The button element used to submit the form.
payment-ach-account-label	label element containing ACH account number input
payment-ach-account	ACH account number input
payment-ach-account-confirm-label	label element containing ACH account number confirmation input
payment-ach-account-confirm	ACH account number confirmation input
payment-ach-routing-label	label element containing ACH routing number input
payment-ach-routing	ACH routing number input
payment-ach-cardholdername-label	label element containing ACH account holder name input
payment-ach-cardholdername	ACH account holder name input
payment-ach-accounttype-owner-label	label element containing select element for selecting Business or Personal ACH account type
payment-ach-accounttype-owner-select	select element for selecting Business or Personal ACH account type
payment-ach-accounttype-purpose-label	label element containing select element for selecting Checking or Savings ACH account type

ID's	
ID	Description
payment-ach-accounttype-purpose-select	select element for selecting Checking or Savings ACH account type

## 2.5 Using an External Submit Button

Depending on the layout of your payment page, you might wish to use your own submit button for the payment form rather than use the one shown in the iframe by default. Here are the steps required for this option:

- When generating the iframe, send the parameter `hmac-include-submit-button="no"`. This will hide the iframe's default submit button.
- To trigger a submit of the iframe payment form, use the `PaymentFrame` object's `submitPaymentData` method as demonstrated in the below example:

```

1 var paymentFrame = new PaymentFrame("my-iframe-id", "https://test.transafe.com");
2 var submitButton = document.getElementById("my-submit-button");
3
4 submitButton.addEventListener("click", function() {
5   paymentFrame.submitPaymentData();
6 });

```

## 2.6 Enabling the Iframe Submit Button

By default, once the iframe-generated payment form is submitted, its submit button is disabled. This is to prevent users from clicking the submit button multiple times and potentially generating multiple payment requests. Some integrations may require the submit button to be re-enabled in certain circumstances, such as a user's card declining, in which case the user would need to enter a different card and try again.

To re-enable the iframe's submit button after submission, use the `PaymentFrame` object's `enableSubmitButton` method as demonstrated in the below example:

```

1 var paymentFrame = new PaymentFrame("my-iframe-id", "https://test.transafe.com");
2 var checkoutForm = document.getElementById("my-checkout-form-id");
3
4 paymentFrame.setPaymentSubmittedCallback(function(response) {
5
6   if (response.code === "AUTH") {
7
8     // Ticket request was successful, prepare to process transaction
9
10    var formData = new FormData(checkoutForm);
11    formData.append('ticket', response.ticket);
12
13    fetch('./transaction.php', {
14      method: 'POST',
15      body: formData
16    }).then(function(response) {
17      response.json().then(function(responseJson) {

```

```

18     if (responseJson.code === "AUTH") {
19         // Transaction was successful, no need to re-enable the submit button
20     } else {
21         // Something went wrong when processing the transaction,
22         // re-enable the iframe's submit button so user can retry if necessary
23         paymentFrame.enableSubmitButton();
24     }
25 });
26 });
27 }
28
29 });

```

## 2.7 Code Example (PHP)

```

1  <?php
2
3  /* Values that will be needed for generating the HMAC */
4  $host_domain = "https://your.website.com";
5  $auth_apikey_id = "P004E346922321910";
6  $auth_apikey_secret = "WrRMpJBU6hGZss45Fv7GJwIFhFo7gApD2L1AblqF63A=";
7
8  $hmac_fields = [];
9
10 /* "timestamp", "domain", "sequence", and "auth_apikey_id" are the required HMAC fields. */
11
12 /* Current Unix timestamp */
13 $hmac_fields["timestamp"] = time();
14
15 /* Domain of the website that will host the iframe */
16 $hmac_fields["domain"] = $host_domain;
17
18 /* Merchant-specified alphanumeric value for tracking/verification purposes.
19  * In production this should be dynamically generated.
20  */
21 $hmac_fields["sequence"] = "abc123";
22
23 /* Merchant Profile API Key ID that will be used to request the iframe
24  * and generate the ticket
25  */
26 $hmac_fields["auth_apikey_id"] = $auth_apikey_id;
27
28 /* Optional field. This is the URL of the CSS file that will be used to style the
29  * iframe's contents. */
30 $hmac_fields["css-url"] = $host_domain . "/css/iframe.css";
31
32 /* Optional field. This will direct Monetra to generate the form with separate
33  * select elements for the expiration date month and year, rather using than
34  * a single text element for the expiration date.
35  */
36 $hmac_fields["expdate-format"] = "separate-selects";
37
38 /* Concatenate all of the defined HMAC fields into a string with no delimiters */
39 $data_to_hash = implode("", $hmac_fields);
40
41 /* Generate the HMAC, using the Monetra Merchant User's password as the key */
42 $hmac = hash_hmac('sha256', $data_to_hash, $auth_apikey_secret);

```

```

43
44 /* Assemble a string containing the "data-" attributes for the iframe element.
45 * This will consist of the HMAC itself and all of the fields included in the HMAC.
46 */
47 $iframe_attributes = [
48   'data-hmac-hmacsha256=' . $hmac . ''
49 ];
50 foreach ($hmac_fields as $key => $value) {
51   $iframe_attributes[] = 'data-hmac-' . $key . '=' . $value . '';
52 }
53 $iframe_attribute_string = implode(" ", $iframe_attributes);
54
55 /* Render the payment page HTML. */
56 ?>
57 <!DOCTYPE html>
58 <html>
59   <head>
60     <meta name="viewport" content="width=device-width, initial-scale=1">
61     <title>Example Shopping Site</title>
62     <link rel="stylesheet" type="text/css" href="./css/host.css" />
63   </head>
64   <body>
65     <main>
66       <h1>Checkout Page</h1>
67       <p>
68         Please fill out your payment information below.
69       </p>
70       <iframe id="myPaymentFrameId" <?php echo $iframe_attribute_string; ?></iframe>
71     </main>
72     <!-- Load the Javascript file containing the PaymentFrame helper object -->
73     <script src="https://test.transafe.com/PaymentFrame/PaymentFrame.js"></script>
74     <script>
75
76     /* Instantiate the PaymentFrame object. The constructor accepts two parameters:
77     * (iframeElementId) The ID of the iframe element on your page that will
78     * contain the PaymentFrame (iframeURL) The URL of the payment server you are
79     * using to generate the PaymentFrame In this case, the iframe element has
80     * an ID of "myPaymentFrameId", and our payment server is https://test.transafe.com.
81     */
82     var paymentFrame = new PaymentFrame(
83       "myPaymentFrameId",
84       "https://test.transafe.com"
85     );
86
87     /* You can use the "setPaymentSubmittedCallback" method of the PaymentFrame
88     * object to set a callback function that will be executed once the payment
89     * form has been submitted. This function will receive a "response" object
90     * containing details about the payment form submission. This won't include
91     * any sensitive data.
92     */
93     paymentFrame.setPaymentSubmittedCallback(function(response) {
94       if (response.code === 'AUTH') {
95         /* If the response code is "AUTH" (meaning the ticket request was
96         * successful), the response object will contain the CardShield ticket,
97         * which can be used in place of card data for the payment
98         * transaction. At this point, you would use the ticket to continue
99         * your checkout/payment flow.
100        */

```

```
101     console.log("The CardShield ticket is " + response.ticket);
102   } else {
103     /* If the response code is "DENY", there was a problem generating the
104      * ticket. In this case, the response object will contain a "verbiage"
105      * property with a brief error message.
106      */
107     console.error(response.verbiage)
108   }
109 });
110
111 /* The "request" method requests a payment form to be loaded into your
112  * iframe from the payment server. Calling this method is the last step to
113  * rendering the PaymentFrame.
114  */
115 paymentFrame.request();
116
117 </script>
118 </body>
119 </html>
120
```