Praln FinTech Co., Ltd. (HQ)
SJ Infinite I Business Complex,
21st Floor Vibhavadi Rangsit Road,
Chompol, Chatuchak, Bangkok, Thailand. 10900
Tax ID: 0105535090912

V.1.0.2

CHILLCREDIT MERCHANT INTEGRATION

MANUAL DOCUMENT [ENGLISH]

+66.2.107.7788	@chillpay	chillpay payment gateway
CS.CHILLPAY	@ChillPay_VI	help@chillpay.co

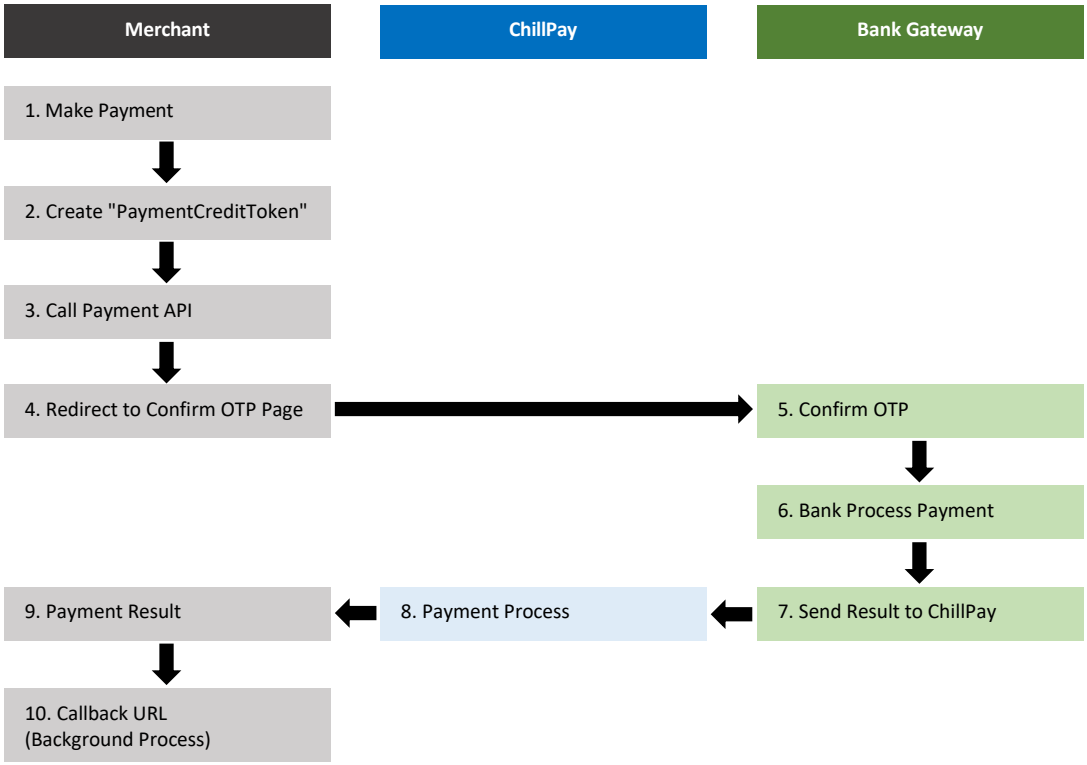
Content

A. System Development Guide: How to connect with ChillCredit

1. Connecting via Credit Card Inline method (supports payment by Credit Card only)

It is a type of connection format on the front end of a website store that allows for designing or customizing the UI to be able to embed a credit card input page directly on the store's website (without redirecting to ChillPay's card input page).

1.1 System Flow of Credit Card Inline Connection



1. The customer (End User) makes a payment transaction, fills in credit card information at the store front, and presses the payment button.
2. . Before the customer transaction data is sent to the store's server, the ChillPay system script embedded on the store's website will shoot an API to send the card data to be stored in the Credit Card Inline system and create a Token (called PaymentCreditToken) to be used in the payment process, along with sending this Token to the store along with the customer transaction data.
3. The customer (End User) makes a payment transaction, fills in credit card information at the store front, and presses the payment button.
4. Before the customer transaction data is sent to the store's server, the ChillPay system script embedded on the store's website will shoot an API to send the card data to be stored in the Credit Card Inline system and create a Token (called PaymentCreditToken) to be used in the payment process, along with sending this Token to the store along with the customer transaction data.
5. When the server of the store receives transaction data from the customer along with a Token, it will use this information to call the API Payment to the ChillPay system, which is an API specifically for Inline Payment (referred to as ChillPay DirectCreditApi). The store will receive a response back with a URL to be used for

redirecting to the OTP input page (which is the bank's page).

6. The merchant's system redirects to the OTP entry page
7. Customers confirm the payment transaction by entering the OTP and submitting it to proceed with the payment process.
8. Bank Payment Process
9. The bank sends the results back to ChillPay's side, ready to redirect back to the ChillPay system.
10. The ChillPay system sends the results back to the merchants through 2 methods: redirecting to the payment confirmation page of the store and shooting a callback to send the payment details back to the merchants (as a background process).
11. The store displays payment options for customers to see.
12. The shop received payment details from the Callback sent back by the ChillPay system.

1.2 Connecting to the Credit Card Inline system.

Connecting to the Credit Card Inline system is divided into 2 parts. The first part involves embedding the Credit Card Inline system script onto the merchant's website so that customers can enter their credit card information directly on the website. The second part involves the merchant's server sending API requests to process payments with the ChillPay system through a specific API for the Credit Card Inline system. The details are as follows:

1.2.1 Embedding the Credit Card Inline system script onto the store's website page by using the ChillPay JavaScript. The payment page of the store must be modified as follows.

1.2.1.1 Add a necessary div tag to serve as a container for displaying the input section of the credit card, which will consist of a total of 5 digits.

```
<div id=""ccdinline-card-name""></div>
<div id=""ccdinline-card-number""></div>
<div id=""ccdinline-card-expiry""></div>
<div id=""ccdinline-card-cvv"" ></div>
<div id=""ccdinline-card-remember""></div>
<div id=""ccdinline-card-select""></div>
```

A. These tags do not need to be in the same location, but the important part is that there must be div ids "ccdinline-card-name", "ccdinline-card-number", "ccdinline-card-expiry", "ccdinline-card-cvv" and "ccdinline-card-remember". These are the required divs for the Credit Card Inline system script to function properly.

B. The div with id "ccdinline-card-select" is optional and may or may not be included, depending on whether it is needed. This div is used for a UI that allows customers (End Users) to select a previously saved card. *For more details, see section 1.2.4 on Using the Card Select UI

C. Merchants can customize the HTML properties of this div or externally as needed.

1.2.1.2 Add Tag Script to call the script of the Credit Card Inline system. This script should be placed within the Payment Form of your own store so that when customers make a payment, the script can generate a token and attach it to the form to send back to the store's server.

```
<script
   src="https://sandbox-bankdemo3.chillpay.co/js/ccdpayment.js"
   data-merchant-code="XXX-MERCHANTCODE-XXX"
   data-api-key="XXXXXXX-API-KEY-XXXXXXX" >
</script>
```

A. By default, the script will perform the operation when the form is submitted, by sending an API request to ChillPay's Credit Card system to create a PaymentCreditToken, which is a token storing card information. This token is added as an input in the form to be sent to the merchant's server, allowing the merchant's server to use the API to continue making a payment.

B. If the script is not in the form, it may prevent the use of the Auto Create PaymentCreditToken system and you may need to manage the event yourself. For more details, refer to section 1.2.2 Managing Events of the Credit Card Inline system and section 1.2.3 Manual PaymentCreditToken Creation.

The src value of the script will change according to the desired environment to connect to, as follows:

Sandbox : https://sandbox-bankdemo3.chillpay.co/js/ccdpayment.js

Production : https://cdn.chill.credit/js/ccdpayment.js

Example of a simple HTML Payment page (file name: payment.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Payment Page</title>
</head>
<body>
<div id="CreditCardData">
        <br>
        <div style="font-size:30px;vertical-align:center;">Payment Page</div>
        <br>
        <div>
                <label>Cardholder Name</label>
                <div id="ccdinline-card-name"></div>
                <label>Card Number</label>
                <div id="ccdinline-card-number"></div>
                <label>Expiry Date</label>
                <div id="ccdinline-card-expiry"></div>
                <label>Security Code</label>
                <div id="ccdinline-card-cvv" ></div>
                <label>Remember Card</label>
```

```
                    <div id="ccdinline-card-remember"></div>
                    <br>
                    <br>
                    <form id="payment-form" method="POST" action="payment_post.php">
                            <script
                                    src="https://sandbox-bankdemo3.chillpay.co/js/ccdpayment.js"
                                    data-merchant-code="XXX-MERCHANTCODE-XXX"
                                    data-api-key="XXXXXXX-API-KEY-XXXXXXX">
                            </script>
                            <label>Payment Details</label>
                            <div><span> Order No : </span><input id="OrderNo" name="OrderNo"
type="text" value="my order" maxlength="20" ></div>
                            <div><span> Customer Id : </span><input id="CustomerId" name="CustomerId"
type="text" value="customer id" maxlength="100" ></div>
                            <div><span> Amount : </span><input id="Amount" name="Amount"
type="number" step="0.01" value="1500.00"  ></div>
                            <button>Submit</button>
                    </form>
            </div>
</div>
</body>
</html>
```

If the script works correctly, when the submit form is pressed, the script will attach the input to the form with all 3 elements.

- **PaymentCreditToken**: This token represents the credit card information that customers input. It will be used to make API payment requests, which the store will send to the ChillPay system for processing.
- **RememberCard**: Has a value of Y or N, which is based on whether the customer chooses to remember the card from the UI of the Credit Card Inline system. It will be used for triggering the payment API as well.
- **CreditToken:** A token value used in the ChillPay system for storing card information. Merchants will receive this value when customers choose to store their card information. Merchants can keep this CreditToken value as an option for customers to use when they want to access card information, they have previously stored.

The script invocation for the store can set UI attributes created by the ChillPay system at a certain level. There are attributes that can be set as shown in Table 1.1

Table 1.1: Attributes that can be set script for the Credit Card Inline system

No.	Attribute Name	Mandatory /Optional	Data Type	Default	Description
1	src	M	string, url		This is the URL value of the ChillPay script needed for use (value varies depending on the connected environment).
2	data-merchant-code	M	string		The MerchantCode provided by the ChillPay system to the merchant.
3	data-api-key	M	string		The ApiKey for the ChillPay system provided to the merchant.
4	data-lang	O	string, langc	th	Set the language display in the UI of the Credit Card Inline system.
5	data-callback-event-receiver	O	string		Provide the name of the JavaScript function that needs to receive events from the Credit Card Inline system. *For further details, refer to section 1.2.2 Event Handling of the Credit Card Inline system.
6	data-auto-adjust-height	O	string, boolean	true	Set the Credit Card Inline system script to automatically adjust the height value of the iframe it creates to match the actual height of the UI inside that iframe precisely (performed once during the creation of the UI script).
7	data-show-error-message	O	string, boolean	true	Set the script to display default error messages for each input.
8	data-auto-create-payment-credit-token-on-submit	O	string, boolean	true	Set the script to insert functionality during form submission by sending a request to ChillPay to create a PaymentCreditToken and automatically attach the data to the form before submission. If this feature is not desired, manual creation of the PaymentCreditToken can be handled. * *For further details, refer to section 1.2.3 Manual PaymentCreditToken Creation.

9	data-frame-bg-color	O	string, color		Set attributes for the iframes created by the script (applied to every iframe).
10	data-frame-width	O	string, element size	100%	Set attributes for the iframes created by the script (applied to every iframe).
11	data-frame-height	O	string, element size	150px	Set attributes for the iframes created by the script (applied to every iframe).
12	data-body-bg-color	O	string, color		Set attributes for the body of the HTML within the iframe (excluding "cardremember").
13	data-body-border-style	O	string, border style		Set attributes for the body of the HTML within the iframe (excluding "cardremember").
14	data-body-border-width	O	string, dimension size		Set attributes for the body of the HTML within the iframe (excluding "cardremember").
15	data-body-border-color	O	string, color		Set attributes for the body of the HTML within the iframe (excluding "cardremember").
16	data-input-bg-color	O	string, color		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
17	data-input-border-style	O	string, border style		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
18	data-input-border-width	O	string, dimension size		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
19	data-input-border-color	O	string, color		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
20	data-input-padding	O	string, dimension size		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
21	data-input-margin	O	string, dimension size		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
22	data-input-font-family	O	string, font family		เซ็ต Attribute ที่ input ของ HTML ที่อยู่ใน iframe (ยกเว้น "card-remember")

23	data-input-font-style	O	string, font style		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
24	data-input-font-weight	O	string, font weight		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
25	data-input-font-size	O	string, font size		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
26	data-input-font-color	O	string, color		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
27	data-input-text-align	O	string, text align		เซ็ต Attribute ที่ input ของ HTML ที่อยู่ใน iframe (ยกเว้น "card-remember")
28	data-input-width	O	string, element size		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
29	data-input-height	O	string, element size		Set attributes for the input elements of the HTML within the iframe (excluding "cardremember").
30	data-error-bg-color	O	string, color		Set attributes for the div error elements of the HTML within the iframe (excluding "card-remember").
31	data-error-border-style	O	string, border style		Set attributes for the div error elements of the HTML within the iframe (excluding "card-remember").
32	data-error-border-width	O	string, dimension size		Set attributes for the div error elements of the HTML within the iframe (excluding "card-remember").
33	data-error-border-color	O	string, color		Set attributes for the div error elements of the HTML within the iframe (excluding "card-remember").
34	data-error-padding	O	string, dimension size		Set attributes for the div error elements of the HTML within the iframe (excluding "card-remember").
35	data-error-margin	O	string, dimension size		เซ็ต Attribute ที่ div error ของ HTML ที่อยู่ใน iframe (ยกเว้น "card-remember")
36	data-error-font-family	O	string, font		Set attributes for the div error elements

			family		of the HTML within the iframe (excluding "card-remember").
37	data-error-font-style	O	string, font style		Set attributes for the div error elements of the HTML within the iframe (excluding "card-remember").
38	data-error-font-weight	O	string, font weight		Set attributes for the div error elements of the HTML within the iframe (excluding "card-remember").
39	data-error-font-size	O	string, font size		Set attributes for the div error elements of the HTML within the iframe (excluding "card-remember").
40	data-error-font-color	O	string, color		Set attributes for the div error elements of the HTML within the iframe (excluding "card-remember").
41	data-error-text-align	O	string, text align		Set attributes for the div error elements of the HTML within the iframe (excluding "card-remember").
42	data-card-name-frame-width	O	string, element size		Set attributes for the iframe with the "card-name" created by the script.
43	data-card-name-frame-height	O	string, element size		Set attributes for the iframe with the "card-name" created by the script.
44	data-card-name-input-width	O	string, element size		Set attributes for the input elements of the HTML within the iframe with the "cardname".
45	data-card-name-input-height	O	string, element size		เซ็ต Attribute ที่ input ของ HTML ที่อยู่ใน iframe "card-name"
46	data-card-number-frame-width	O	string, element size		Set attributes for the input elements of the HTML within the iframe with the "cardname".
47	data-card-number-frame-height	O	string, element size		Set attributes for the input elements of the HTML within the iframe with the "cardname".
48	data-card-number-input-width	O	string, element size		Set attributes for the input elements of the HTML within the iframe with the "cardnumber".
49	data-card-number-input-height	O	string, element size		Set attributes for the input elements of the HTML within the iframe with the "cardnumber".

50	data-card-expiry-frame-width	O	string, element size		Set attributes for the iframe with the "card-expiry" created by the script.
51	data-card-expiry-frame-height	O	string, element size		Set attributes for the iframe with the "card-expiry" created by the script.
52	data-card-expiry-input-width	O	string, element size		Set attributes for the input elements of the HTML within the iframe with the "cardexpiry"ใ
53	data-card-expiry-input-height	O	string, element size		Set attributes for the input elements of the HTML within the iframe with the "cardexpiry"
54	data-card-cvv-frame-width	O	string, element size		Set attributes for the iframe with the "card-cvv" created by the script.
55	data-card-cvv-frame-height	O	string, element size		Set attributes for the iframe with the "card-cvv" created by the script.
56	data-card-cvv-input-width	O	string, element size		Set attributes for the input elements of the HTML within the iframe with the "cardcvv".
57	data-card-cvv-input-height	O	string, element size		Set attributes for the input elements of the HTML within the iframe with the "cardcvv".
58	data-card-remember-frame-width	O	string, element size		Set attributes for the iframe with the "card-remember" created by the script.
59	data-card-remember-frame-height	O	string, element size		Set attributes for the iframe with the "card-remember" created by the script.
60	data-card-remember-checkbox-style	O	string, checkbox style	rect-1	Set attributes for the iframe with the "card-remember" created by the script.
61	data-card-consent-font-family	O	string, font family		Set attributes for the div consent elements of the HTML within the iframe with the "card-remember".
62	data-card-consent-font-style	O	string, font style		Set attributes for the div consent elements of the HTML within the iframe with the "card-remember".
63	data-card-consent-font-weight	O	string, font weight		Set attributes for the div consent elements of the HTML within the iframe with the "card-remember".
64	data-card-consent-font-size	O	string, font size		Set attributes for the div consent elements of the HTML within the iframe with the "card-remember".

65	data-card-consent-font-color	O	string, color		Set attributes for the div consent elements of the HTML within the iframe with the "card-remember"
66	data-card-consent-text-align	O	string, text align		Set attributes for the div consent elements of the HTML within the iframe with the "card-remember".
67	data-card-select-frame-width	O	string, element size	100%	Set the attribute of the "card-select" iframe that the script creates
68	data-card-select-frame-height	O	string, element size	150px	Set the attribute of the "card-select" iframe that the script creates.
69	data-card-select-show-card-type	O	string, boolean	true	Set the display information for each data box (selectable card information) to show whether to display the Card Type or not.
70	data-card-select-show-card-number	O	string, boolean	true	Set the display information for each data box (selectable card information) to show whether to display the Card Number or not.
71	data-card-select-show-card-name	O	string, boolean	false	Set the display information for each data box (selectable card information) to show whether to display the Card Name or not.
72	data-card-select-show-card-expiry	O	string, boolean	true	Set the display information for each data box (selectable card information) to show whether to display the Card Expiry or not.
73	data-card-select-container-display	O	string, display type		Set the attribute of the div container in the HTML within the "card-select" iframe.
74	data-card-select-container-flex-direction	O	string, flex direction		Set the attribute of the div container in the HTML within the "card-select" iframe.
75	data-card-select-container-flex-wrap	O	string, flex wrap		Set the attribute of the div container in the HTML within the "card-select" iframe.
76	data-card-select-container-padding	O	string, dimension size		Set the attribute of the div container in the HTML within the "card-select" iframe.

77	data-card-select-container-margin	O	string, dimension size		Set the attribute of the div container in the HTML within the "card-select" iframe
78	data-card-select-container-width	O	string, element size		Set the attribute of the div container in the HTML within the "card-select" iframe.
79	data-card-select-container-height	O	string, element size		Set the attribute of the div container in the HTML within the "card-select" iframe.
80	data-card-select-box-bg-color	O	string, color		Set the attribute of the div containing the card information (box) in the HTML within the "cardselect" iframe.
81	data-card-select-box-padding	O	string, dimension size	8px 16px	Set the attribute of the div containing the card information (box) in the HTML within the "cardselect" iframe.
82	data-card-select-box-margin	O	string, dimension size		Set the attribute of the div containing the card information (box) in the HTML within the "cardselect" iframe.
83	data-card-select-box-width	O	string, element size		Set the attribute of the div containing the card information (box) in the HTML within the "cardselect" iframe.
84	data-card-select-box-height	O	string, element size		Set the attribute of the div containing the card information (box) in the HTML within the "cardselect" iframe.
85	data-card-select-box-font-family	O	string, font family		Set the attribute of the div containing the card information (box) in the HTML within the "cardselect" iframe.
86	data-card-select-box-font-style	O	string, font style		Set the attribute of the div containing the card information (box) in the HTML within the "cardselect" iframe.
87	data-card-select-box-font-weight	O	string, font weight		Set the attribute of the div containing the card information (box) in the HTML within the "cardselect" iframe.
88	data-card-select-box-font-size	O	string, font size		Set the attribute of the div containing the card information (box) in the HTML within the "cardselect" iframe.
89	data-card-select-box-font-color	O	string, color		Set the attribute of the div containing the card information (box) in the HTML

					within the "cardselect" iframe.
90	data-card-select-box-text-align	O	string, text align		Set the attribute of the div containing the card information (box) in the HTML within the "cardselect" iframe.
91	data-card-select-box-hover-bg-color	O	string, color	rgba(160, 160, 160, 0.3)	Set the attribute for the div containing the card information (box) in the HTML within the "cardselect" iframe for the hover event.
92	data-card-select-box-hover-font-style	O	string, font style		Set the attribute for the div containing the card information (box) in the HTML within the "cardselect" iframe for the hover event.
93	data-card-select-box-hover-font-weight	O	string, font weight		Set the attribute for the div containing the card information (box) in the HTML within the "cardselect" iframe for the hover event.
94	data-card-select-box-hover-font-size	O	string, font size		Set the attribute for the div containing the card information (box) in the HTML within the "cardselect" iframe for the hover event.
95	data-card-select-box-hover-font-color	O	string, color		Set the attribute for the div containing the card information (box) in the HTML within the "cardselect" iframe for the hover event.
96	data-card-select-box-selected-bg-color	O	string, color	rgba(102, 153, 255, 0.4)	Set the attribute for the div containing the card information (box) in the HTML within the "cardselect" iframe for the selected event.
97	data-card-select-box-selected-font-style	O	string, font style		Set the attribute for the div containing the card information (box) in the HTML within the "cardselect" iframe for the selected event.
98	data-card-select-box-selected-font-weight	O	string, font weight		Set the attribute for the div containing the card information (box) in the HTML within the "cardselect" iframe for the selected event.
99	data-card-select-box-selected-font-size	O	string, font size		Set the attribute for the div containing the card information (box) in the HTML

					within the "cardselect" iframe for the selected event.
100	data-card-select-box-selected-font-color	O	string, color	white	Set the attribute for the div containing the card information (box) in the HTML within the "cardselect" iframe for the selected event.

The details of the Data Types are as follows:

- **string :** the value entered for script attributes will always be enclosed in double quotation marks ("").
- **url** : the value is a valid URL.
- **langcode** : the acceptable values are "en" for English and "th" for Thai.
- **boolean** : the acceptable values are "true" or "false".
- **color** : acceptable values typically include:
    - **13 basic colors** : The colors include transparent, black, blue, brown, gray, green, orange, pink, purple, red, violet, white, yellow.
    - **hex color codes :** They begin with "#" followed by a 6-digit hexadecimal number. For example, "#0F0F0F".
    - **rgb color codes :** They are in the format rgb(<N>, <N>, <N>), where <N> is a number from 0 to 255. For example, "rgb(255, 255, 255)".
    - **rgba color codes :** They are in the format rgba(<N>, <N>, <N>, <D>), where <N> is a number from 0 to 255, and <D> is a decimal number ranging from 0 to 1 with a maximum of 6 decimal places. For example, "rgba(255, 100, 100, 0.5)".
    - **hsl color codes :** They are in the format hsl(<H>, <P>%, <P>%), where is a number from 0 to 359, and <P> is a number from 0 to 100. For example, "hsl(9, 100%, 64%)"
    - **hsla color codes :** They are in the format hsla(<H>, <P>%, <P>%, <D>), where is a number from 0 to 359, <P> is a number from 0 to 100, and <D> is a decimal number ranging from 0 to 1 with a maximum of 6 decimal places. For example, "hsla(9, 100%, 64%, 0.5)".
- **element size** : You can specify values in two formats: pixel (px) or percentage (%). Pixel values are always integers, while percentage values can be decimals with up to 6 decimal places. For example, "300px", "50%", "60.02%".
- **border style** : Acceptable values include dotted, dashed, solid, double, groove, ridge, inset, outset, none, hidden.
- **dimension size** : This is for setting the size of the border, which only accepts pixel (px) values. You can set it in 4 different ways:
    - Setting a single value will apply the same size to all four sides (top, bottom, right, left). For example, "4px".
    - Setting two values will apply the first value to the top and bottom, and the second value to the right and left. For example, "2px 4px".

- Setting three values will apply the first value to the top, the second value to the right and left, and the third value to the bottom. For example, "2px 4px 2px".
- Setting four values will apply the first value to the top, the second value to the right, the third value to the bottom, and the fourth value to the left. For example, "2px 4px 2px 4px".

- **font family** : Acceptable values include "times new roman", "georgia", "garamond", "arial", "verdana", "courier new", "lucida console", "monaco", "brush script mt", "lucida handwriting", "copperplate", "papyrus".
- **font style** : Acceptable values include "normal", "italic".
- **font weight** : Acceptable values include "normal", "bold".
- **font size** : Specify values in two formats: pixel (px) or em. For em values, you can include up to 6 decimal places. For example, "22px", "1.12345em".
- **text align** : Acceptable values include "center", "left", "right".
- **checkbox style** : Acceptable values include "rect-1", "rect-2", "rect-3", "circle-1", "circle-2", "circle3", "slide-1", "slide-2", "slide-3", "slide-4", "slide-5".
- **display type** : Acceptable values include block, inline-block, flex, inline-flex
- **flex direction** : Acceptable values include row, row-reverse, column, column-reverse
- **flex wrap** : Acceptable values include nowrap, wrap, wrap-reverse

1.2.1.3 The merchant's server will execute an API call to process the payment transaction with the ChillPay system through the dedicated API for the Credit Card Inline system.

Sandbox : https://sandbox-api-directcredit.chillpay.co/api/v1/payment

Production : https://api-directcredit.chillpay.co/api/v1/payment

The API Payment call will include headers as specified in Table 1.2, parameters according to the details in Table 1.3, and the response will follow the details in Table 1.4. Request parameters should be sent in JSON format, and the response received will also be in JSON format.

When making the API call, if the operation is successful, the response will return with HTTP Code 200 (OK). However, if an error occurs, the response will return with HTTP Code 400 (Bad Request).

Table 1.2 : Headers for calling the ChillPay Credit Card Inline Payment API

No.	Header Name	Data Type	Mandatory/Optional	Description
1	CHILLPAY-MerchantCode	String	M	The MerchantCode provided by the ChillPay system to the merchant.
2	CHILLPAY-ApiKey	String	M	The ApiKey for the ChillPay system provided to the merchant.

Table 1.3 : Request Parameters for calling the ChillPay Credit Card Inline Payment API

No.	Request Parameter	Data Type	Length	Mandatory/ Optional	Description	Remark
1	OrderNo	String	20	M	Order number or reference code for the transaction.	It should be a unique value generated by the store, consisting only of numbers or a combination of numbers and English letters. Special characters such as +-*/-#$_ or others are not allowed.
2	CustomerId	String	100	M	Customer reference code (End User) or customer name.	For example, CUS000001 or Mr. Bank Spaces are allowed only in the middle, not at the beginning or end, and special characters such as +-*/-#$_ or others are not allowed.
3	Amount	Number	12	M	The payment amount for the transaction. *For currencies JPY and KRW, there will be no decimal places.	For example, if the input value is 55025, the system will interpret it as 550.25. For currencies JPY and KRW, the system will interpret it as 55025.00.
4	PhoneNumber	Number	10	O	The mobile phone number of the customer (End User).	
5	Description	String	100	O	Payment details information.	For example, "Product: iPhone X 64GB Color: Black".
6	ChannelCode	String	30	M	The reference code representing the receiving bank.	For example, "creditcard". Details can be found in the appendix E. Currently, the Credit Card Inline system only supports "creditcard" and "installment_kbank".
7	Currency	String	3	M	The currency code data should be set as a currency number.	For example, "764". Details can be found in Appendix D.
8	LangCode	String	2	O	The language code to be displayed on the	TH: Thai language EN: English language

					bank's payment page.	
9	RouteNo	Number	2	M	Route number is a value provided by the ChillPay system.	For example, "1". If the store needs to use more than one channel number, they can contact customersupport@chillp ay.co.
10	IPAddress	String	20	M	Customer's IP Address information.	For example, 203.255.255.155
11	TokenType	String	2	M	The format for using the token for payment.	DT: Payment by normal method (using PaymentCreditToken) CT: Payment using tokenized card (using PaymentCreditToken + CreditToken)
12	CreditToken	String	255	O	Tokenized card information.	It is the token provided by the ChillPay system when the customer selects to tokenize their card. If TokenType is CT, this value must be included as well.
13	DirectCreditToken	String	255	M	Token representing card data.	It is the token provided by the ChillPay system when the customer makes a payment through the Credit Card Inline channel.
14	CreditMonth	Number	2	O	The number of months for installment payments made through a credit card transaction.	For example, 3, 8, 10. *If it's an installment payment channel, this value must be specified.
15	ShopID	String	2	O	The Shop ID for processing installment payments through Credit Card transactions.	01: For merchants responsible for interest charges. 02: For customers (End Users) responsible for interest charges. *If it's an installment payment channel, this value must be

						specified
16	CustEmail	String	255	O	Email of the customer (End User).	
17	SaveCard	String	1	O	Data indicating whether there will be card tokenization from this payment or not.	N: No card tokenization Y: Card tokenization *If not specified, the default value is N: No card tokenization.
18	CheckSum	String	32	M	The result obtained from hashing all the values using the MD5 algorithm	Used to verify the correctness of the transaction.

The method for calculating the CheckSum.

1. Concatenate the values of parameters from 1 to 17 in Table 1.3 in sequence. If a parameter does not have a value, it is considered as an empty string.
2. Append the SecretKey obtained from ChillPay to the end of the concatenated string. (The SecretKey is a confidential value that should not be disclosed and is not publicly available).
3. Hash the concatenated data using the MD5 algorithm. Details can be found at https://en.wikipedia.org/wiki/MD5. After hashing, you will obtain an MD5 Hash value, which will be used as the CheckSum parameter (as described in point 18 in Table 1.3).

Example of CheckSum Calculation

Example of Parameter Data Used to Call the Payment API


```
{
    "OrderNo" : "Ord000001",
    "CustomerId" : "Cust_Test",
    "Amount" : 1500000,
    "PhoneNumber" : "0911111111",
    "Description" : "Desc Test",
    "ChannelCode" : "creditcard",
    "Currency" : "764",
    "LangCode" : "TH",
    "RouteNo" : 1,
    "IPAddress" : "127.0.0.1",
    "TokenType" : "DT",
    "CreditToken" : null,
    "DirectCreditToken" : "x1mPrAKM2kiZoQdrN7bwTbOBxvtvs3kw",
    "CreditMonth" : null,
```

```
        "ShopID" : null,
        "CustEmail" : "customer_01@a@b.com",
        "SaveCard" : "N"
}
```

Example of Other Necessary Data for the Merchant

```
Merchant SecretKey: "XXXXXXSECRETXXXXXX"
```

From the calculation method in step 1, find the concatenated data.

```
ConcatData = OrderNo + CustomerId + Amount + PhoneNumber + Description + ChannelCode + Currency + LangCode + RouteNo + IPAddress + TokenType + CreditToken + DirectCreditToken + CreditMonth + ShopID + CustEmail + SaveCard
```

Since CreditToken, CreditMonth, and ShopID have null values, they are replaced with blank values. Therefore, the result will be as follows:

```
ConcatData = "Ord000001Cust_Test150000009111111111Desc Testcreditcard764TH1127.0.0.1DTx1mPrAKM2kiZoQdrN7bwTbOBxvtvs3kwcustomer_01@a@b.comN"
```

From the calculation method in step 2, appending with the Merchant SecretKey will yield the following result:

```
ConcatData = "Ord000001Cust_Test150000009111111111Desc Testcreditcard764TH1127.0.0.1DTx1mPrAKM2kiZoQdrN7bwTbOBxvtvs3kwcustomer_01@a@b.comNXXXXXXSECRETXXXXXX"
```

From the calculation method in step 3, hashing with MD5 will yield the following result:

```
CheckSum = "2b332382ff893b8c4c18960d87499e4b"
```

Table 1.4 : Response Parameters Returned from Calling the Payment API of ChillPay's Credit Card Inline System

No.	Response Parameter	Data Type	Length	Mandatory/ Optional	Description	Remark
1	status	int		M	Transaction Status Code (Result of API Call)	See details in the appendix A.
2	message	string	255	M	Description of Transaction Status	
3	data	object		O	The result data.	If the transaction is unsuccessful, there will be no data available.
3.1	data.paymentStatus	string	30	M	Payment Status	See details in the appendix B.
3.2	data.amount	long		M	The amount paid for the	For example, if the input

					transaction (the last two digits represent the decimal part). *For JPY and KRW currencies, there will be no decimal part.	value is 55025, the system will interpret it as 550.25. For currencies JPY and KRW, the system will interpret it as 55025.00.
3.3	data.orderNo	string	20	M	Order number or reference code of the transaction.	The value received from the merchant
3.4	data.customerId	string	100	M	The URL to be displayed when the customer's transaction is successful.	It is the result page of the merchant's website (Result URL).
3.5	data.returnUrl	string	255	M	The URL to be displayed when the customer's transaction is successful.	It is the result page of the merchant's website (Result URL).
3.6	data.paymentUrl	string	255	O	It is the URL to redirect to the bank's OTP confirmation page	The merchant needs to program the redirection to this URL.
3.7	data.ipAddress	string	20	M	The customer's IP address making the transaction.	For example, 20.25.55.20
3.8	data.token	string	60	M	Token for referencing the transaction.	Used for ChillPay.
3.9	data.transactionId	long		M	Reference number for the merchant.	To be used for crossreferencing between the merchant and ChillPay, for example, 10071.
3.10	data.channelCode	string	20	M	Reference code for the bank receiving the payment.	For example, "creditcard" see details in the appendix E. Currently, the Credit Card Inline system only supports "creditcard" and "installment_kbank".
3.11	data.createdDate	string	20	M	The transaction date.	Format: YYYYMMDDHH MMSS Example: 20180712173122
3.12	data.expiredDate	string	20	M	Expiration date of the transaction	Format: YYYYMMDDHH MMSS Example:

						20180712173122

Example of calling Payment API (PHP language code) (file name: payment_post.php)

```
<?php

// Utilities Class/Function
class DirectPaymentPostData {
	public $OrderNo;
	public $CustomerId;
	public $Amount;
	public $PhoneNumber;
	public $Description;
	public $ChannelCode;
	public $Currency;
	public $LangCode;
	public $RouteNo;
	public $IPAddress;
	public $TokenType;
	public $CreditToken;
	public $DirectCreditToken;
	public $CreditMonth;
	public $ShopID;
	public $CustEmail;
	public $SaveCard;
	public $CheckSum;

	public function  getConcatString() {
		return $this->OrderNo . $this->CustomerId . $this->Amount . $this->PhoneNumber .
				$this->Description . $this->ChannelCode . $this->Currency . $this->LangCode .
				$this->RouteNo . $this->IPAddress . $this->TokenType . $this->CreditToken .
				$this->DirectCreditToken . $this->CreditMonth . $this->ShopID . $this->CustEmail .
				$this->SaveCard;
	}
	public function  calculateCheckSum($secretKey) {
		$this->CheckSum = md5($this->getConcatString() . $secretKey);
	}
}
function IsNullOrEmptyString($str){
```

```
    return (!isset($str) || $str === null || trim($str) === '');
}

// CONFIG
$paymentUrl = ""https://sandbox-api-directcredit.chillpay.co/api/v1/payment"";
$merchantCode = ""XXX-MERCHANTCODE-XXX"";
$apiKey = ""XXXXXXX-API-KEY-XXXXXXX"";
$secretKey = ""XXXXXXX-SECRET-KEY-XXXXXXX"";

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        echo ""Inline Payment Example: <br>"";

    // from POST [Script Inline]
        $paymentCreditToken = isset($_POST['PaymentCreditToken']) ? $_POST['PaymentCreditToken'] : null;
        $creditToken = isset($_POST['CreditToken']) ? $_POST['CreditToken'] : null;
        $rememberCard = isset($_POST['RememberCard']) ? $_POST['RememberCard'] : null;

        echo ""Script POST Data: <br>"";
        echo ""  PaymentCreditToken = "" . $paymentCreditToken . ""<br>"";
        echo ""  CreditToken = "" . $creditToken . ""<br>"";
        echo ""  RememberCard = "" . $rememberCard . ""<br>"";

        // from POST [Merchant's page]
        $orderNo = $_POST['OrderNo'];
        $customerId = $_POST['CustomerId'];
        $amount = intval(floatval($_POST['Amount']) * 100);

        echo ""Merchant POST Data: <br>"";
        echo ""  OrderNo = "" . $orderNo . ""<br>"";
        echo ""  CustomerId = "" . $customerId . ""<br>"";
        echo ""  Amount = "" . $amount . ""<br>"";

        // find tokenType from creditToken
        $tokenType = IsNullOrEmptyString($creditToken) ? ""DT"" : ""CT"";

        // post data
        $postData = new DirectPaymentPostData();
        $postData->OrderNo = $orderNo;
```

```
$postData->CustomerId = $customerId;
$postData->Amount = $amount;
$postData->PhoneNumber = "0911111111";
$postData->Description = "Description";
$postData->ChannelCode = "creditcard";
$postData->Currency = "764";
$postData->LangCode = "TH";
$postData->RouteNo = 1;
$postData->IPAddress = "127.0.0.1";
$postData->TokenType = $tokenType;
$postData->CreditToken = $creditToken;
$postData->DirectCreditToken = $paymentCreditToken;
$postData->CreditMonth = null;
$postData->ShopID = null;
$postData->CustEmail = "customer@a.com";
$postData->SaveCard = $rememberCard;

echo "ConcatData = " . $postData->getConcatString() . "<br><br>";

$postData->calculateCheckSum($secretKey);
echo "CheckSum = " . $postData->CheckSum . "<br><br>";

$postDataJson = json_encode($postData);
echo "PostData = " . $postDataJson . "<br><br>";

// call direct payment api
$curl = curl_init();
curl_setopt_array($curl, array(
        CURLOPT_URL => $paymentUrl,
        CURLOPT_RETURNTRANSFER => true,
        CURLOPT_ENCODING => "",
        CURLOPT_MAXREDIRS => 10,
        CURLOPT_TIMEOUT => 30,
        CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
        CURLOPT_CUSTOMREQUEST => "POST",
        CURLOPT_POSTFIELDS => $postDataJson,
        CURLOPT_HTTPHEADER => array(
                "Cache-Control: no-cache",
```

```
                "Content-Type: application/json",
                "CHILLPAY-MerchantCode: " . $merchantCode,
                "CHILLPAY-ApiKey: " . $apiKey
            ),
        ));
        $response = curl_exec($curl);
        $err = curl_error($curl);
        curl_close($curl);

        if ($err) {
            echo "cURL Error #:" . $err;
        } else {
            echo "response #:" . $response . "<br><br>";

            $respObj = json_decode($response, true);

            if ($respObj['status'] != 200) {
                echo "API FAILED status: " . $respObj['status'] . ", message: " . $respObj['message'] . "<br><br>";
            } else {
                if ($respObj['data']['paymentStatus'] == "Paid") {
                    echo "PAIID SUCCESS <br><br>";
                } else if ($respObj['data']['paymentStatus'] == "WaitAuthorize") {
                    $redirectUrl = $respObj['data']['paymentUrl'];

                    echo "wait 10 sec to redirect to url #:" . $redirectUrl . "<br><br>";
                    header("refresh:10; url='". $redirectUrl . "'");
                } else {
                    echo "PAYMENT STATUS ERROR paymentStatus: " . $respObj['data']['paymentStatus'] . "<br><br>";
                }
            }
        }
}

?>
```

Example of Response Data (JSON) received from calling the API in the case of a successful transaction.


```
{
        "status" : 200,
        "message" : "Success",
        "data" : {
                "paymentStatus" : "WaitAuthorize",
                "amount" : 1500000,
                "orderNo" : "Ord000001",
                "customerId" : "Cust_Test",
                "returnUrl" : "https://merchant.com/result",
                "paymentUrl" : "https://bank.com/confirmOtp",
                "ipAddress" : "127.0.0.1",
                "token" : "xxxxxxx",
                "transactionId" : 123456,
                "channelCode" : "creditcard ",
                "createdDate" : "20240408133145",
                "expiredDate" : "20240409133145"
        }
}
```


Example of Response Data (JSON) received from calling the API in the case of an unsuccessful transaction.


```
{
            "status" : 3001,
            "message" : "Payment failed",
            "data" : null
}
```


1.2.2 Management of Script Event for Credit Card Inline System.

In the script system of Credit Card Inline, merchants can write additional JavaScript to handle various events of the Inline Script system. The details of all events are as listed in Table 1.5

Table 1.5 : Events for Credit Card Inline Script

No.	Event Id	Event SubId	Data Type	Data Structure	Remark
1	Initialized		string		This is an event that occurs when initializing the Inline UI is completed. It can be used in conjunction with invoking functions from the Inline

					Script system. The functions of the Inline Script system can only be executed once the UI initialization is completed.

2	CreateTokenSucceed	The value is "Success"	object	{
				PaymentCreditToken: "XXXXXX",
				CreditToken: "YYYYYY",
				RememberCard: "N"
				}
					The event occurs when the creation of PaymentCreditToken is successful, with the following details:
					PaymentCreditToken: The token created successfully.
					CreditToken: The credit token set in the Inline script system.
					RememberCard: Value from the checkbox, which can be "", "N", or "Y". The default value is "". This is equivalent to "N". If the checkbox is checked, the value will be "Y"; otherwise, it will be "N".

3	CreateTokenFailed	The value can be as follows:
		0: Success
		1: Fail
		2: Error
		3: SystemError
			object	{
				Code: 3001,
				Message: "System Error",
				}
					The event occurs when the creation of PaymentCreditToken is unsuccessful due to an error from the server side, with the following details:
					Code: The error code from the server side resulting from the unsuccessful creation of PaymentCreditToken.
					Message: The error messages.

4	CreateTokenError	The value can be as follows: - 1: Invalid Input – 2: Invalid	string	The message will vary depending on the Event SubId, which can be either "Invalid Input" or	The event occurs when the creation of PaymentCreditToken is unsuccessful due to an error

		MerchantCode or MerchantApiKey - Other Values: Represent the status value obtained from the XMLHttpRequest when an error occurs.		"Invalid MerchantCode or MerchantApiKey," or it will be the status text obtained from the XMLHttpRequest when an error occurs.	on the script side, such as failed parameter validation or encountering an error while calling an API, such as a failed API connection.
5	InputState	The value indicates which input it is, with the following possible values: "card-name": Input for the cardholder's name "card-number": Input for the card number "card-expiry": Input for the card expiry date "card-cvv": Input for the CVV (Card Verification Value)	boolean	true	The event occurs when there is a change in the data of an input within the script. The Data parameter is a boolean indicating whether the input is currently valid or not. *This event can be used in conjunction with the script attribute "data-show-errormessage" by setting the attribute to false to disable the default error message of the script system. Then, you can capture this event to allow the merchant side to display their own error message instead.

6	AddCardSelectSucceed		object	{
				CreditToken: "YYYYYY",
				CardType: "VISA"
				}
					This is an event that occurs when adding card information (from CreditToken) to the Card Select UI is successful. The details are as follows:
					CreditToken: The token that was added
					CardType: The type of card, such as VISA, MASTERCARD, JCB, UnionPay

7	AddCardSelectFailed	The value can be as follows: 0: Success	object	{ Code: 3001, Message: "System	This is an event that occurs when adding card information (from

		1: Fail 2: Error 3: SystemError		Error", }	CreditToken) to the Card Select UI fails due to a server-side error. The details are as follows: **Code**: The error code from the server when the CreditToken could not be found **Message**: The error message
8	AddCardSelectError	The value can be as follows: -2: Invalid MerchantCode or MerchantApiKey Other values: The status value from the xhr (XMLHttpRequest) when an error occurs	string	The message will vary according to the Event SubId, which can be "Invalid MerchantCode or MerchantApiKey" or the status text received from the xhr (XMLHttpRequest) when an error occurs	This is an event that occurs when adding card information (from CreditToken) to the Card Select UI fails due to a script-side error, such as a parameter validation failure or an error while calling the API, such as an inability to connect to the API.

9	RemoveCardSelectSucceed	The value will be "Success".	object	{
				CreditToken: "YYYYYY"
				}
					This is an event that occurs when removing card information (from CreditToken) from the Card Select UI is successful. The details are as follows:
					CreditToken: The token that was removed

10	UseCreditTokenSucceed	The value will be "Success".	object	{
				CreditToken: "YYYYYY",
				CardType: "VISA"
				}
					This is an event that occurs when using a CreditToken (specifying which CreditToken Inline Script should use for payment) is successful. The details are as follows:
					CreditToken: The token that was used
					CardType: The type of card, such as VISA, MASTERCARD, JCB, UnionPay

11	UseCreditTokenFailed	The possible values are: 0: Success
		1: Fail
		2: Error
		3: SystemError
			object	{
				Code: 3001,
				Message: "System Error",
				}
					This is an event that occurs when using a CreditToken (specifying which CreditToken Inline Script should use for payment) fails due to an error from the server side. The details are as follows:
					Code: The error code from the server when the CreditToken retrieval fails
					Message: The error message

12	UseCreditTokenError	The values can be as follows: - 2: Invalid MerchantCode or MerchantApiKey Other values: Status values obtained from xhr (XMLHttpRequest) when an error occurs	string	The message will vary according to the Event SubId, which can be "Invalid MerchantCode or MerchantApiKey" or the status text obtained from xhr (XMLHttpRequest) when an error occurs.	This is an event that occurs when using a CreditToken (specifying which CreditToken Inline Script should use for payment) fails due to a script-side error, such as parameter validation failure or an error while calling the API, like inability to connect to the API.
13	UnuseCreditTokenSucceed	The value will be "Success".	string		This is an event that occurs when unuse a CreditToken (canceling the selection of a CreditToken previously chosen for Inline Script usage) is successful.

14	CheckCreditTokenSucceed	The value will be "Success".	object	{
				CreditToken: "YYYYYY",
				CardType: "VISA"
				}
					This is an event that occurs when a function to check a CreditToken (verifying the status of a CreditToken on the server to determine if it can be used) is successful. The details are as follows:
					CreditToken: The token that was checked
					CardType: The type of card, such as VISA, MASTERCARD,

					JCB, UnionPay

15	CheckCreditTokenFailed	The possible values are: 0: Success
		1: Fail
		2: Error
		3: SystemError
			object	{
				Code: 3001,
				Message: "System Error",
				}
					This is an event that occurs when a function to check a CreditToken (verifying the status of a CreditToken on the server to determine if it can be used) fails due to an error from the server side. The details are as follows:
					Code: The error code from the server when the CreditToken check fails
					Message: The error message

16	CheckCreditTokenError	The value can be as follows: - 2: Invalid MerchantCode or MerchantApiKey Other values: Status values obtained from xhr (XMLHttpRequest) when an error occurs	string	The message will vary according to the Event SubId, which can be "Invalid MerchantCode or MerchantApiKey" or the status text obtained from xhr (XMLHttpRequest) when an error occurs.	This is an event that occurs when a function to check a CreditToken (verifying the status of a CreditToken on the server to determine if it can be used) fails due to a script-side error, such as parameter validation failure or an error while calling the API, like inability to connect to the API.

17	DeleteCreditTokenSucceed	The value will be "Success".	object	{
				CreditToken: "YYYYYY"
				}
					This is an event that occurs when a function to delete a CreditToken (removing CreditToken data on the server) is successful. The details are as follows:
					CreditToken: The token that was deleted

18	DeleteCreditTokenFailed	The possible values are: 0: Success
		1: Fail
		2: Error
		3: SystemError
			object	{
				Code: 3001,
				Message: "System Error",
				}
					This is an event that occurs when a function to delete a CreditToken (removing CreditToken data on the server) fails due to an error

					from the server side. The details are as follows: **Code**: The error code from the server when the attempt to delete the CreditToken fails **Message**: The error message
19	DeleteCreditTokenError	The value can be as follows: - 2: Invalid MerchantCode or MerchantApiKey Other values: Status values obtained from xhr (XMLHttpRequest) when an error occurs	string	The message will vary according to the Event SubId, which can be "Invalid MerchantCode or MerchantApiKey" or the status text obtained from xhr (XMLHttpRequest) when an error occurs.	This is an event that occurs when a function to delete a CreditToken (removing CreditToken data on the server) fails due to a scriptside error, such as parameter validation failure or an error while calling the API, such as inability to connect to the API.

The process of editing the HTML page to manage script events on your own.

1) Add a JavaScript function to the HTML page to serve as a callback function from the Inline Script system.
   Example script callback function

```
<script>
    function CCDCallbackEvent(eventId, subEventId, data) {
        if (eventId === "Initialized") {
            console.log("CCDCallbackEvent: Initialized:");
        }
        else if (eventId === "CreateTokenSucceed") {
            console.log("CCDCallbackEvent: CreateTokenSucceed: PaymentCreditToken=" +
data.PaymentCreditToken + ", CreditToken=" + data.CreditToken + ", RememberCard=" + data.RememberCard);
        }
        else if (eventId === "CreateTokenFailed") {
            console.log("CCDCallbackEvent: CreateTokenFailed: status=" + subEventId + ", code=" + data.Code + ",
message=" + data.Message);
        }
        else if (eventId === "CreateTokenError") {
            console.log("CCDCallbackEvent: CreateTokenError: status=" + subEventId + ", message=" + data);
        }
        else if (eventId === "InputState") {
```

```
                console.log("CCDCallbackEvent: InputState: input=" + subEventId + ", IsDataValid=" + data);
            }
            else if (eventId === "AddCardSelectSucceed") {
                console.log("CCDCallbackEvent: AddCardSelectSucceed: CreditToken=" + data.CreditToken + ",
CardType=" + data.CardType);
            }
            else if (eventId === "AddCardSelectFailed") {
                console.log("CCDCallbackEvent: AddCardSelectFailed: status=" + subEventId + ", code=" + data.Code +
", message=" + data.Message);
            }
            else if (eventId === "AddCardSelectError") {
                console.log("CCDCallbackEvent: AddCardSelectError: status=" + subEventId + ", message=" + data);
            }
            else if (eventId === "RemoveCardSelectSucceed") {
                console.log("CCDCallbackEvent: RemoveCardSelectSucceed: CreditToken=" + data.CreditToken);
            }
            else if (eventId === "UseCreditTokenSucceed") {
                console.log("CCDCallbackEvent: UseCreditTokenSucceed: CreditToken=" + data.CreditToken + ",
CardType=" + data.CardType);
            }
            else if (eventId === "UseCreditTokenFailed") {
                console.log("CCDCallbackEvent: UseCreditTokenFailed: status=" + subEventId + ", code=" + data.Code +
", message=" + data.Message);
            }
            else if (eventId === "UseCreditTokenError") {
                console.log("CCDCallbackEvent: UseCreditTokenError: status=" + subEventId + ", message=" + data);
            }
            else if (eventId === "UnuseCreditTokenSucceed") {
                console.log("CCDCallbackEvent: UnuseCreditTokenSucceed:");
            }
            else if (eventId === "CheckCreditTokenSucceed") {
                console.log("CCDCallbackEvent: CheckCreditTokenSucceed: CreditToken=" + data.CreditToken + ",
CardType=" + data.CardType);
            }
            else if (eventId === "CheckCreditTokenFailed") {
                console.log("CCDCallbackEvent: CheckCreditTokenFailed: status=" + subEventId + ", code=" + data.Code
+ ", message=" + data.Message);
            }
```

```
            else if (eventId === "CheckCreditTokenError") {
                console.log("CCDCallbackEvent: CheckCreditTokenError: status=" + subEventId + ", message=" + data);
            }
            else if (eventId === "DeleteCreditTokenSucceed") {
                console.log("CCDCallbackEvent: DeleteCreditTokenSucceed: CreditToken=" + data.CreditToken);
            }
            else if (eventId === "DeleteCreditTokenFailed") {
                console.log("CCDCallbackEvent: DeleteCreditTokenFailed: status=" + subEventId + ", code=" +
data.Code + ", message=" + data.Message);
            }
            else if (eventId === "DeleteCreditTokenError") {
                console.log("CCDCallbackEvent: DeleteCreditTokenError: status=" + subEventId + ", message=" + data);
            }
        }
</script>
```

2) Add an attribute to set the callback function for the script.

Example of calling a script with an added attribute to utilize a callback function.

```
<script
    src="https://sandbox-bankdemo3.chillpay.co/js/ccdpayment.js"
    data-merchant-code="XXX-MERCHANTCODE-XXX"
    data-api-key="XXXXXXX-API-KEY-XXXXXXX"
    data-callback-event-receiver="CCDCallbackEvent" >
</script>
```

Example of calling a script with an added attribute to utilize a callback function:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Payment Page</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script>
        var cardNameValid = true;
        var cardNumberValid = true;
        var cardExpiryValid = true;
        var cardCvvValid = true;
```

```
        function RefreshErrorMessage() {
            if (!cardNameValid) {
                $('#my-custom-error-msg').html("Card name is invalid!!");
            }
            else if (!cardNumberValid) {
                $('#my-custom-error-msg').html("Card number is invalid!!");
            }
            else if (!cardExpiryValid) {
                $('#my-custom-error-msg').html("Card expiry is invalid!!");
            }
            else if (!cardCvvValid) {
                $('#my-custom-error-msg').html("Card cvv is invalid!!");
            }
            else {
                $('#my-custom-error-msg').html("");
            }
        }
        function CCDCallbackEvent(eventId, subEventId, data) {
            if (eventId === "InputState") {
                if (subEventId === "card-name") {
                    cardNameValid = data;
                    RefreshErrorMessage();
                }
                else if (subEventId === "card-number") {
                    cardNumberValid = data;
                    RefreshErrorMessage();
                }
                else if (subEventId === "card-expiry") {
                    cardExpiryValid = data;
                    RefreshErrorMessage();
                }
                else if (subEventId === "card-cvv") {
                    cardCvvValid = data;
                    RefreshErrorMessage();
                }
            }
        }
    </script>
```

```
</head>
<body>
    <div id="CreditCardData">
        <br>
        <div style="font-size:30px;vertical-align:center;" >Payment Page</div>
        <br>
        <div>
            <label>Cardholder Name</label>
            <div id="ccdinline-card-name"></div>
            <label>Card Number</label>
            <div id="ccdinline-card-number"></div>
            <label>Expiry Date</label>
            <div id="ccdinline-card-expiry"></div>
            <label>Security Code</label>
            <div id="ccdinline-card-cvv" ></div>
            <label>Remember Card</label>
            <div id="ccdinline-card-remember"></div>
            <br>
            <div id="my-custom-error-msg" style="color:red; font-size: 22px; "></div>
            <br>
            <form id="payment-form" method="POST" action="payment_post.php">
                <script
                    src="https://sandbox-bankdemo3.chillpay.co/js/ccdpayment.js"
                    data-merchant-code="XXX-MERCHANTCODE-XXX"
                    data-api-key="XXXXXXX-API-KEY-XXXXXXX"
                    data-callback-event-receiver="CCDCallbackEvent"
                    data-show-error-message="false" >
                </script>

                <label>Payment Details</label>
                <div><span> Order No : </span><input id="OrderNo" name="OrderNo" type="text" value="my
order" maxlength="20" ></div>
                <div><span> Customer Id : </span><input id="CustomerId" name="CustomerId" type="text"
value="customer id" maxlength="100" ></div>
                <div><span> Amount : </span><input id="Amount" name="Amount" type="number" step="0.01"
value="1500.00"  ></div>
            <button>Submit</button>
            </form>
```

```
            </div>
        </div>
</body>
</html>
```

1.2.3 Adjusting Script Usage for Manual PaymentCreditToken Creation

In the Credit Card Inline script system, typically, the process of creating a PaymentCreditToken occurs when the form is submitted, and the token data (including CreditToken and RememberCard) is attached to the form.

However, in some cases, the merchant's webpage may not support this functionality. For instance, the payment submission might be on a different page from the one where the card details are entered, or there might not be a form present at all (the form is created dynamically upon form submission, or the submission is done via AJAX, or input names need to be changed).

In such scenarios, the merchant needs to handle the PaymentCreditToken creation process themselves. This can be achieved as follows:

1) Add an attribute to disable the option of auto-creating PaymentCreditToken on form submission (Attribute "data-auto-create-payment-credit-token-on-submit") and add a callback receiver to receive and handle events related to PaymentCreditToken creation.

   Example of calling a script with added attributes to utilize a callback function.

```
<script
        src="https://sandbox-bankdemo3.chillpay.co/js/ccdpayment.js"
        data-merchant-code="XXX-MERCHANTCODE-XXX"
        data-api-key="XXXXXXX-API-KEY-XXXXXXX"
        data-callback-event-receiver="CCDCallbackEvent"
        data-auto-create-payment-credit-token-on-submit="false" >
</script>
```

2) Add a script that will execute the process of creating PaymentCreditToken within the logic.

The operation of your store's webpage involves calling the following JavaScript function.

```
ccdinline.CreatePaymentCreditToken();
```

3) Write JavaScript code to handle the event results of PaymentCreditToken creation. There are three events that need to be handled:
   1. CreateTokenSucceed: This event occurs when the creation of PaymentCreditToken is successful. It provides three pieces of necessary information: PaymentCreditToken, CreditToken, and RememberCard. These pieces of information are essential for subsequent API calls for Payment.
   2. CreateTokenFailed: This event occurs when the creation of PaymentCreditToken fails due to an error on ChillPay's server. Error Code and Error Message data will be provided.
   3. CreateTokenError: This event occurs when the creation of PaymentCreditToken fails due to an error on the Inline script side or during the connection to the Credit Card Inline system's API. Status and Status Message data will be provided.

Example of an HTML page for a payment process with auto-create-payment-credit-token-onsubmit disabled in the Inline Script, and then manually creating PaymentCreditToken.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Payment Page</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script>
        var isOnCreate = false;
        var currentPaymentCreditToken = null;
        var timoutId;
        var isSubmit = false;

        function SetCrearPaymentCreditToken() {
            if (timoutId) {
                clearTimeout(timoutId);
            }

            timoutId = setTimeout(() => {
                currentPaymentCreditToken = null;
            }, 120000);
        }
        $(document).ready(function() {
            $('#payment-btn').on("click", function() {
                if (isOnCreate) {
                    return false;
                }
                if (currentPaymentCreditToken === null) {
                    return false;
                }
                if (isSubmit) {
                    return false;
                }

                isSubmit = true;
                $('#payment-form').submit();
```

```
                });
                $('#crate-payment-credit-token-btn').on("click", function() {
                        if (isOnCreate) {
                                return false;
                        }

                        isOnCreate = true;
                        ccdinline.CreatePaymentCreditToken();
                });
        });
        function CCDCallbackEvent(eventId, subEventId, data) {
                if (eventId === "CreateTokenSucceed") {
                        $('#PaymentCreditToken').val(data.PaymentCreditToken);
                        $('#CreditToken').val(data.CreditToken);
                        $('#RememberCard').val(data.RememberCard);

                        currentPaymentCreditToken = data.PaymentCreditToken;
                        SetCrearPaymentCreditToken();

                        isOnCreate = false;
                }
                else if (eventId === "CreateTokenFailed") {
                        console.error("CCDCallbackEvent: CreateTokenFailed: status=" + subEventId + ",
code=" + data.Code + ", message=" + data.Message);
                        isOnCreate = false;
                }
                else if (eventId === "CreateTokenError") {
                        console.error("CCDCallbackEvent: CreateTokenError: status=" + subEventId + ",
message=" + data);
                        isOnCreate = false;
                }
        }
    </script>
</head>
<body>
    <div id="CreditCardData">
        <br>
        <div style="font-size:30px;vertical-align:center; ">Payment Page</div>
```

```
            <br>
            <div>
                <label>Cardholder Name</label>
                <div id="ccdinline-card-name"></div>
                <label>Card Number</label>
                <div id="ccdinline-card-number"></div>
                <label>Expiry Date</label>
                <div id="ccdinline-card-expiry"></div>
                <label>Security Code</label>
                <div id="ccdinline-card-cvv" ></div>
                <label>Remember Card</label>
                <div id="ccdinline-card-remember"></div>
                <br>
                <br>
                <form id="payment-form" method="POST" action="payment_post.php">
                    <script
                        src="https://sandbox-bankdemo3.chillpay.co/js/ccdpayment.js"
                        data-merchant-code="XXX-MERCHANTCODE-XXX"
                        data-api-key="XXXXXXX-API-KEY-XXXXXXX"
                        data-callback-event-receiver="CCDCallbackEvent"
                        data-auto-create-payment-credit-token-on-submit="false" >
                    </script>

                    <label>Payment Details</label>
                    <div><span> Order No : </span><input id="OrderNo" name="OrderNo"
type="text" value="my order" maxlength="20" ></div>
                    <div><span> Customer Id : </span><input id="CustomerId" name="CustomerId"
type="text" value="customer id" maxlength="100" ></div>
                    <div><span> Amount : </span><input id="Amount" name="Amount"
type="number" step="0.01" value="1500.00"  ></div>
                    <input id="PaymentCreditToken" name="PaymentCreditToken" type="hidden"
value="" >
                    <input id="CreditToken" name="CreditToken" type="hidden" value="" >
                    <input id="RememberCard" name="RememberCard" type="hidden" value="" >
                <button id="payment-btn">Submit</button>
                </form>
                <button id="crate-payment-credit-token-btn">Create PaymentCreditToken</button>
            </div>
```

```
    </div>
</body>
</html>
```

*By default, PaymentCreditToken has a lifespan of 120 seconds (2 mins). If it is not used within this period, it will expire and need to be recreated. Therefore, merchants must consider this aspect when managing or creating PaymentCreditToken themselves.

1.2.4 Using the Card Select UI

The operation of the Inline Script system related to the use of CreditToken has two modes: the use of the Card Select UI, which is the built-in UI of the Inline Script system itself, and the other mode where the merchant manages the selection of the customer's CreditToken themselves (either by creating a UI for the customer to choose or by the merchant selecting it for the customer). This method allows the Inline Script system to recognize which CreditToken is being used. The merchant's website must call the script system's function to manually invoke the CreditToken (see details in section 1.2.5 on using CreditToken manually). This section covers the first mode, which is the use of the Card Select UI.

Whether to use the Card Select UI can be specified by creating a div with id "ccdinline-card-select" (details in section 1.2.1.1). If this div is present, it indicates the selection of the Card Select UI mode. If this div is absent, it indicates the selection of the manual CreditToken mode.

The usage of this mode will have the following functions available:

1. A function to add card information for selection in the Card Select UI.

```
ccdinline.AddCreditTokenToCardSelectUI(requestExpireDate, creditToken, securityCheck);
```

The details of the necessary parameters for calling the function are as follows:

- **requestExpireDate**: Specifies the expiration date of this request (used to enhance security in accessing information and limit the usage of this parameter set within a certain time frame). The value is in UTC time zone and in the format "yyyyMMddHHmmss". For example, "20240408134550" corresponds to April 8, 2024, at 13:45:50 (UTC time; to convert to Thai time, add 7 hours). If the specified time has passed, this entire set of parameters will no longer be usable and must be recreated (i.e., specify a new requestExpireDate and recalculate the merchantSecurityCheck).
- **creditToken**: This is a token representing the card information that the customer has previously paid with and instructed to save. The merchant receives this token when the customer successfully pays through the Background Url provided to the ChillPay system during service registration. The merchant must store and manage this token by mapping it to their own customer information to correctly select the customer's CreditToken.
- **merchantSecurityCheck**: This is an MD5 hash value in the specified format. The method for calculating the MerchantSecurityCheck is as follows:

1.) Concatenate the information in the following order: MerchantCode + MerchantApiKey + RequestExpireDate + CreditToken + MerchantSecretKey.

```
ConcatData = "XXX-MERCHANTCODE-XXXXXXXXXX-API-KEY-
XXXXXXX20240408134550MYCREDITTOKENXXXXXXX-SECRET-KEY-XXXXXXX"
```

2.) Take the concatenated information from step 1) and compute the MD5 hash of this data to obtain the value for MerchantSecurityCheck.

```
MerchantSecurityCheck = "573b38a80d803cd7559f4c347a08d1dc"
```

*Calculating the MerchantSecurityCheck requires the use of the MerchantSecretKey. Therefore, this logic should reside on the server side of the merchant. The merchant can implement this in two ways:
The first method is to calculate and obtain the value when the customer enters the payment page, so the information is ready from the start. This method might face issues with the parameters expiring before they can be used, and the merchant must manage this aspect.
The second method is to create an internal API within the merchant's system to generate the MerchantSecurityCheck when calling the function of the Inline Script system. This method is more flexible but might be more complex to implement and requires careful consideration of security to prevent unauthorized usage (limit usage to only the merchant's own website).

When called, the result of the data request will be sent as an event, with three event IDs that need to be handled as follows:

- AddCardSelectSucceed : : This event occurs when adding card information to the Card Select UI is successful. Details of the information received from this event can be found in Table 1**.5**

- AddCardSelectFailed : This event occurs when adding card information to the Card Select UI fails due to an error on the ChillPay server side. Details of the information received from this event can be found in Table 1.5

- AddCardSelectError : : This event occurs when adding card information to the Card Select UI fails due to an error on the Inline Script side or an error during connection to the system's API. Details of the information received from this event can be found in Table 1.5

2. A function to remove card information from the Card Select UI.

```
ccdinline.RemoveCreditTokenFromCardSelectUI(creditToken);
```

The details of the necessary parameters for calling the function are as follows:

- **creditToken**: This is a token representing the card information that the customer has previously paid with and instructed to save. The merchant receives this token when the customer successfully pays through the Background URL provided to the ChillPay system during service registration. The merchant must store and manage this token by mapping it to their own customer information to correctly select the customer's CreditToken. When called, the result of the data request will be sent as an event, with one event ID that needs to be handled as follows:
- **RemoveCardSelectSucceed**: This event occurs when removing card information from the Card Select UI is successful. Details of the information received from this event can be found in Table 1.5

*There will be no error in the deletion process. This means that if the creditToken sent does not exist in the UI, no deletion will occur, but the result will still be considered successful.

* Calling functions available in the Inline Script system requires waiting for the Inline Script system to complete initialization before they can be utilized.

Example of an HTML page for payment using the Card Select UI.

This example demonstrates the use of the Card Select UI with three cards added, each containing existing creditToken data.

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<title>Payment Page</title>
	<script>
		// sample data for add to Card Select UI
		var cardSelectList = [
			{ RequestExpireDate: "20240515133045", CreditToken: "XXX-CreditToken-1-XXX",
MerchantSecurityCheck: "1234567890abcdef1234567890abcdef" },
			{ RequestExpireDate: "20240515133045", CreditToken: "XXX-CreditToken-2-XXX",
MerchantSecurityCheck: "1234567890abcdef1234567890abcdef" },
			{ RequestExpireDate: "20240515133045", CreditToken: "XXX-CreditToken-3-XXX",
MerchantSecurityCheck: "1234567890abcdef1234567890abcdef" }
		];

		function CCDCallbackEvent(eventId, subEventId, data) {
			if (eventId === "Initialized") {
				console.log("CCDCallbackEvent: Initialized:");

				for (let i = 0; i < cardSelectList.length; i++) {

	ccdinline.AddCreditTokenToCardSelectUI(cardSelectList[i].RequestExpireDate,
cardSelectList[i].CreditToken, cardSelectList[i].MerchantSecurityCheck);
				}
			}
			else if (eventId === "AddCardSelectSucceed") {
				console.log("CCDCallbackEvent: AddCardSelectSucceed: CreditToken=" +
data.CreditToken + ", CardType=" + data.CardType);
			}
			else if (eventId === "AddCardSelectFailed") {
				console.log("CCDCallbackEvent: AddCardSelectFailed: status=" + subEventId + ",
code=" + data.Code + ", message=" + data.Message);
			}
```

```
                        else if (eventId === "AddCardSelectError") {
                                console.log("CCDCallbackEvent: AddCardSelectError: status=" + subEventId + ",
message=" + data);
                        }
                        else if (eventId === "RemoveCardSelectSucceed") {
                                console.log("CCDCallbackEvent: RemoveCardSelectSucceed: CreditToken=" +
data.CreditToken);
                        }
        }
        </script>
</head>
<body>
        <div id="CreditCardData">
                <br>
                <div style="font-size:30px;vertical-align:center;">Payment Page</div>
                <br>
                <div>
                        <label>Cardholder Name</label>
                        <div id="ccdinline-card-name"></div>
                        <label>Card Number</label>
                        <div id="ccdinline-card-number"></div>
                        <label>Expiry Date</label>
                        <div id="ccdinline-card-expiry"></div>
                        <label>Security Code</label>
                        <div id="ccdinline-card-cvv" ></div>
                        <label>Remember Card</label>
                        <div id="ccdinline-card-remember"></div>
                        <label>Select Card</label>
                        <div id="ccdinline-card-select"></div>
                        <br>
                        <br>
                        <form id="payment-form" method="POST" action="payment_post.php">
                                <script
                                        src="https://sandbox-bankdemo3.chillpay.co/js/ccdpayment.js"
                                        data-merchant-code="XXX-MERCHANTCODE-XXX"
                                        data-api-key="XXXXXXX-API-KEY-XXXXXXX"
                                        data-callback-event-receiver="CCDCallbackEvent" >
                                </script>
```

```
                                    <label>Payment Details</label>
                                    <div><span> Order No : </span><input id="OrderNo" name="OrderNo"
type="text" value="my order" maxlength="20" ></div>
                                    <div><span> Customer Id : </span><input id="CustomerId" name="CustomerId"
type="text" value="customer id" maxlength="100" ></div>
                                    <div><span> Amount : </span><input id="Amount" name="Amount"
type="number" step="0.01" value="1500.00"  ></div>
                            <button>Submit</button>
                            </form>
                    </div>
            </div>
</body>
</html>
```

1.2.5 Manual CreditToken Usage

If merchants wish to manage customer selection of CreditToken themselves (such as creating their own selection UI or directly specifying CreditToken usage for customers), they can do so by not using the Card Select UI of the Inline Script system (i.e., not creating the div with id "ccdinline-card-select"). This allows them to utilize functions that enforce the Inline Script system to use CreditToken directly.

Manual CreditToken usage provides the following functions for operation:

1. A function to enforce the Inline Script system to use the desired CreditToken.

```
ccdinline.UseCreditToken(requestExpireDate, creditToken, securityCheck);
```

The details regarding parameters will be similar to those for the function "ccdinline.AddCreditTokenToCardSelectUI" as outlined in section 1.2.4

When called, the result of the data request will be sent as an event, with three event IDs that need to be handled as follows:

- **UseCreditTokenSucceed** : This event occurs when forcing the Inline Script system to use the desired CreditToken is successful. Details of the information received from this event can be found in Table 1.5
- **UseCreditTokenFailed** : This event occurs when forcing the Inline Script system to use the desired CreditToken fails due to an error on the ChillPay server. Details of the information received from this event can be found in Table 1.5
- **UseCreditTokenError** : This event occurs when forcing the Inline Script system to use the desired CreditToken fails due to an error on the Inline Script side or during the connection with the system's API. Details of the information received from this event can be found in Table 1.5

2. A function to force the Inline Script system to cancel the usage of the currently used CreditToken.

```
ccdinline.UnuseCreditToken();
```

When called, the result of the data request will be sent as an event, with one event ID that needs to be handled

as follows:

- **UnuseCreditTokenSucceed**: This event occurs when the Inline Script system successfully cancels the usage of the CreditToken that was currently active. You can refer to table 1.5 for details on the information received from this event.

*Canceling the use of CreditToken will not result in an error. This means that even if no CreditToken is currently selected in the Inline Script system at that moment, you can still execute this command without encountering an error, and the result will remain successful.

Calling functions within the Inline Script system requires waiting for the Inline Script to complete its initialization before they can be used.

Example: HTML page for payment using the selected CreditToken forcibly

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<title>Payment Page</title>
	<script>
		// sample data for UseCreditToken
		var useCreditToken = { RequestExpireDate: "20240515133045", CreditToken: "XXX-CreditToken-1-XXX",
MerchantSecurityCheck: "1234567890abcdef1234567890abcdef" };

		function CCDCallbackEvent(eventId, subEventId, data) {
			if (eventId === "Initialized") {
				console.log("CCDCallbackEvent: Initialized:");
				ccdinline.UseCreditToken(useCreditToken.RequestExpireDate, useCreditToken.CreditToken,
useCreditToken.MerchantSecurityCheck);
			}
			else if (eventId === "UseCreditTokenSucceed") {
				console.log("CCDCallbackEvent: UseCreditTokenSucceed: CreditToken=" +
data.CreditToken + ", CardType=" + data.CardType);
			}
			else if (eventId === "UseCreditTokenFailed") {
				console.log("CCDCallbackEvent: UseCreditTokenFailed: status=" + subEventId + ", code=" +
data.Code + ", message=" + data.Message);
			}
			else if (eventId === "UseCreditTokenError") {
				console.log("CCDCallbackEvent: UseCreditTokenError: status=" + subEventId + ", message="
+ data);
```

```
                }
                else if (eventId === "UnuseCreditTokenSucceed") {
                        console.log("CCDCallbackEvent: UnuseCreditTokenSucceed:");
                }
        }
        </script>
</head>

<body>
        <div id="CreditCardData">
                <br>
                <div style="font-size:30px;vertical-align:center;">Payment Page</div>
                <br>
                <div>
                        <label>Cardholder Name</label>
                        <div id="ccdinline-card-name"></div>
                        <label>Card Number</label>
                        <div id="ccdinline-card-number"></div>
                        <label>Expiry Date</label>
                        <div id="ccdinline-card-expiry"></div>
                        <label>Security Code</label>
                        <div id="ccdinline-card-cvv" ></div>
                        <label>Remember Card</label>
                        <div id="ccdinline-card-remember"></div>
                        <br>
                        <br>
                        <form id="payment-form" method="POST" action="payment_post.php">
                                <script
                                        src="https://sandbox-bankdemo3.chillpay.co/js/ccdpayment.js"
                                        data-merchant-code="XXX-MERCHANTCODE-XXX"
                                        data-api-key="XXXXXXX-API-KEY-XXXXXXX"
                                        data-callback-event-receiver="CCDCallbackEvent" >
                                </script>

                                <label>Payment Details</label>
                                <div><span> Order No : </span><input id="OrderNo" name="OrderNo" type="text"
value="my order" maxlength="20" ></div>
                                <div><span> Customer Id : </span><input id="CustomerId" name="CustomerId"
```

```
type="text" value="customer id" maxlength="100" ></div>
                        <div><span> Amount : </span><input id="Amount" name="Amount" type="number"
step="0.01" value="1500.00"  ></div>
                    <button>Submit</button>
                    </form>
                </div>
        </div>
</body>
</html>
```

1.2.6 Using the CheckCreditToken and DeleteCreditToken functions

These two functions are utility functions provided by the Inline Script system for merchants to manage CreditToken. They are as follows:

1. Function for checking if the CreditToken is ready for use or not.

```
ccdinline.CheckCreditToken(requestExpireDate, creditToken, securityCheck);
```

Details regarding the parameters will be similar to the function "ccdinline.AddCreditTokenToCardSelectUI" as described in section 1.2.4

When called, the result of the data retrieval will be sent as an event, with 3 event IDs to handle as follows:

- **CheckCreditTokenSucceed** : This event occurs when ChillPay successfully verifies that the CreditToken is ready for use. Details can be viewed in table 1.5
- **CheckCreditTokenFailed** : This event occurs when ChillPay checks the CreditToken and either cannot find the token or it is in a state not ready for use. Details can be viewed in table 1.5
- **CheckCreditTokenError** : This event occurs when there is an unsuccessful attempt to check the desired CreditToken, due to an error either in the Inline script or during the connection with the system's API. Details can be viewed in table 1.5

2. Function for deleting CreditToken from the system

```
ccdinline.DeleteCreditToken(requestExpireDate, creditToken, securityCheck);
```

By details regarding parameters, it should be similar to the function "ccdinline.AddCreditTokenToCardSelectUI" in section 1.2.4

- **DeleteCreditTokenSucceed** : This event occurs when ChillPay successfully deletes a CreditToken from the system. Details about the data received from this event can be found in Table 1.5
- **DeleteCreditTokenFailed** : This event occurs when ChillPay attempts to delete a CreditToken from the system but fails due to an error on ChillPay's server. Details about the data received from this event can be found in Table 1.5
- **DeleteCreditTokenError** : This event occurs when ChillPay attempts to delete a CreditToken from the system but encounters an error either in the Inline script or during the connection to the system's API. Details about the data received from this event can be found in Table 1.5
-

Calling functions available in the Inline Script system requires waiting for the Inline Script to complete initialization before they can be used.

An example is an HTML payment page that includes checking the selected CreditToken.

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="utf-8">
	<meta name="viewport" content="width=device-width, initial-scale=1.0">
	<title>Payment Page</title>
	<script>
		// sample data for CheckCreditToken
		var checkCreditToken = { RequestExpireDate: "20240515133045", CreditToken: "XXX-CreditToken-1-XXX", MerchantSecurityCheck: "1234567890abcdef1234567890abcdef" };

		function CCDCallbackEvent(eventId, subEventId, data) {
			if (eventId === "Initialized") {
				console.log("CCDCallbackEvent: Initialized:");

				ccdinline.CheckCreditToken(checkCreditToken.RequestExpireDate, checkCreditToken.CreditToken, checkCreditToken.MerchantSecurityCheck);
			}
			else if (eventId === "CheckCreditTokenSucceed") {
				console.log("CCDCallbackEvent: CheckCreditTokenSucceed: CreditToken=" + data.CreditToken + ", CardType=" + data.CardType);
			}
			else if (eventId === "CheckCreditTokenFailed") {
				console.log("CCDCallbackEvent: CheckCreditTokenFailed: status=" + subEventId + ", code=" + data.Code + ", message=" + data.Message);
			}
			else if (eventId === "CheckCreditTokenError") {
				console.log("CCDCallbackEvent: CheckCreditTokenError: status=" + subEventId + ", message=" + data);
			}
			else if (eventId === "DeleteCreditTokenSucceed") {
				console.log("CCDCallbackEvent: DeleteCreditTokenSucceed: CreditToken=" + data.CreditToken);
			}
			else if (eventId === "DeleteCreditTokenFailed") {
```

```
                                console.log("CCDCallbackEvent: DeleteCreditTokenFailed: status=" + subEventId
+ ", code=" + data.Code + ", message=" + data.Message);
                        }
                        else if (eventId === "DeleteCreditTokenError") {
                                console.log("CCDCallbackEvent: DeleteCreditTokenError: status=" + subEventId
+ ", message=" + data);
                        }
        }
        </script>
</head>

<body>
        <div id="CreditCardData">
                <br>
                <div style=""font-size:30px;vertical-align:center;"">Payment Page</div>
                <br>
                <div>
                        <label>Cardholder Name</label>
                        <div id="ccdinline-card-name"></div>
                        <label>Card Number</label>
                        <div id="ccdinline-card-number"></div>
                        <label>Expiry Date</label>
                        <div id="ccdinline-card-expiry"></div>
                        <label>Security Code</label>
                        <div id="ccdinline-card-cvv" ></div>
                        <label>Remember Card</label>
                        <div id="ccdinline-card-remember"></div>
                        <br>
                        <br>
                        <form id="payment-form" method="POST" action="payment_post.php">
                                <script
                                        src="https://sandbox-bankdemo3.chillpay.co/js/ccdpayment.js"
                                        data-merchant-code="XXX-MERCHANTCODE-XXX"
                                        data-api-key="XXXXXXX-API-KEY-XXXXXXX"
                                        data-callback-event-receiver="CCDCallbackEvent" >
                                </script>

                                <label>Payment Details</label>
```

```
                                    <div><span> Order No : </span><input id="OrderNo" name="OrderNo"
type="text" value="my order" maxlength="20" ></div>
                                    <div><span> Customer Id : </span><input id="CustomerId" name="CustomerId"
type="text" value="customer id" maxlength="100" ></div>
                                    <div><span> Amount : </span><input id="Amount" name="Amount"
type="number" step="0.01" value="1500.00"  ></div>
                            <button>Submit</button>
                            </form>
                    </div>
            </div>
</body>
</html>
```

2. Connecting ChillCredit Public Api

2.1 API for CreatePaymentCreditToken

The API CreatePaymentCreditToken call will include headers as specified in Table 2.1, parameters according to the details in Table 2.2, and the response will follow the details in Table 2.3. Request parameters should be sent in JSON format, and the response received will also be in JSON format. When making the API call, if the operation is successful, the response will return with HTTP Code 200 (OK). However, if an error occurs, the response will return with HTTP Code 400 (Bad Request).

- URL Sandbox (POST Method) : https://sandbox-bankdemo3.chillpay.co/ChillCredit/public/CreatePaymentCreditToken
- URL Production (POST Method) : https://api.chill.credit/api/v1/public/PaymentCreditToken/Create

Table 2.1 : Headers for calling the CreatePaymentCreditToken API

No.	Header Name	Data Type	Mandatory/Optional	Default
1	Content-Type	String	M	application/json

Table 2.2 : Request Parameters for calling the CreatePaymentCreditToken Api

No.	Request Parameter	Data Type	Length	Mandatory /Optional	Description	Remark
1	Portal	String	50	M	The data provided by ChillPay.	Ex. ChillPay
2	Merchant	String	50	M	The data provided by ChillPay.	Ex. M000001
3	MerchantAPIKey	String	1000	M	API keycode for system connection	ChillPay provided the data.
4	CardNumber	String	30	M	Credit card number	
5	ExpireMonth	String	2	M	Expire Month of credit card	Ex. 01
6	ExpireYear	String	4	M	Expire Year of credit card	Ex. 2026
7	NameOnCard	String	255	M	Name of credit card	
8	CVV	String	3	M	card verification value	Ex. 123
9	PortalCreditToken	String	500	M	Token for remember card function	It is a token issued by the ChillPay system when a customer chooses to remember card. If you want to use a previously memorized card, you must have this value

						as well.
10	MerchantCheckSum	String	32	M	The result of bringing all the values to encrypt MD5.	Used to confirm the validity of the list.

The method for calculating the CheckSum.

1. Concatenate the values of parameters from 1 to 10 in Table 2.2 in sequence. If a parameter does not have a value, it is considered as an empty string.
2. Append the SecretKey obtained from ChillPay to the end of the concatenated string. (The SecretKey is a confidential value that should not be disclosed and is not publicly available).
3. Hash the concatenated data using the MD5 algorithm. Details can be found at https://en.wikipedia.org/wiki/MD5. After hashing, you will obtain an MD5 Hash value, which will be used as the CheckSum parameter (as described in point 10 in Table 1.3).

Example of CheckSum Calculation

```
{
        "Portal": "ChillPay",
        "Merchant": "M000001",
          "MerchantAPIKey": "xxx",
        "CardNumber": "123456789",
        "ExpireMonth": "01",
         "ExpireYear": "2026",
        "NameOnCard": "testname",
        "CVV": "123",
        "PortalCreditToken": "123456789"
}
```

Example of Other Necessary Data for the Merchant

```
Merchant SecretKey: "yyy"
```

From the calculation method in step 1, find the concatenated data.

```
ConcatData = Portal + Merchant + MerchantAPIKey + CardNumber + ExpireMonth + ExpireYear + NameOnCard + CVV + PortalCreditToken + Merchant SecretKey
```

From the calculation method in step 2, appending with the Merchant SecretKey will yield the following result:

```
ConcatData = " ChillPayM000001xxx123456789012026testname123123456789yyy"
```

From the calculation method in step 3, hashing with MD5 will yield the following result:

```
MerchantCheckSum = "2b332382ff893b8c4c18960d87499e4b"
```

Table 2.3 : Response Parameters Returned from Calling the CreatePaymentCreditToken API

No.	Response Parameter	Data Type	Length	Mandatory /Optional	Description	Remark
1	status	int		M	Transaction result code (Order No. 1)	See detail in Appendix A.
2	code	int		M	Transaction result code (Order No. 2)	See detail in Appendix B.
3	message	string	255	M	Explanation for transaction result (Order No. 2)	
4	totalRecord	int		O	Total record	
5	data	object		O		
5.1	data.portal	string	50	M	The data provided by ChillPay.	Ex. ChillPay
5.2	data.merchant	string	50	M	The data provided by ChillPay.	Ex. M000001
5.3	data. paymentCreditToken	string	32	M	Token for payment transactions	It is a token issued by the ChillCredit system for use in making payment transactions in the next step.

Example of calling CreatePaymentCreditToken API (PHP language code)

```
<?php
$curl = curl_init();
curl_setopt_array($curl, array(
  CURLOPT_URL => 'https://sandbox-bankdemo3.chillpay.co/ChillCredit/public/CreatePaymentCreditToken',
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => 'POST',
  CURLOPT_POSTFIELDS =>'{
    "Portal": " portal01",
    "Merchant": "M000001":,
    "MerchantAPIKey": "xxx",
    "CardNumber": "123456789",
    "ExpireMonth": "01",
```

```
    "ExpireYear": "2026",
    "NameOnCard": "testname",
    "CVV": "123",
    "PortalCreditToken": "123456789",
    "MerchantCheckSum": "2b332382ff893b8c4c18960d87499e4b"
}',
CURLOPT_HTTPHEADER => array(
    'Content-Type: application/json'
  ),
));
$response = curl_exec($curl);
curl_close($curl);
?>
```

Example of Response Data (JSON) received from calling the API in the case of a successful transaction.

```
{
    "status": 0,
    "code": 200,
    "message": "Success",
    "totalRecord": 1,
    "data": {
        "portal": "ChillPay",
        "merchant": "M000001",
        "paymentCreditToken": "-pUkAKDF3EhbN6IQ3m5rTafnrXbGDCfe"
    }
}
```

Example of Response Data (JSON) received from calling the API in the case of an unsuccessful transaction.

```
{
    "status": 1,
    "code": 1001,
    "message": "Invalid CVV",
    "totalRecord": 0,
    "data": null
}
```

B. Information and Testing Procedure for Sandbox test

Credit card number information for Sandbox test

Credit card number - For Test Credit Card, Pay with points, Installment (Except Krungsri Consumer and Krungsri First Choice payment channels)				
Card Number	Card Scheme	Expiry (MM/YY)	CVV	OTP Pass
4141 0000 0000 1414	VISA	Any	Any	123456
5353 0000 0000 3535	MASTER Card	Any	Any	123456
3535 0000 0000 5353	JCB	Any	Any	123456
6262 0000 0000 2626	UnionPay	Any	Any	123456

C. Appendix

Appendix A.

Result Code of Transaction No. 1

Response Code	Description	Remark
0	Successful transaction	Success
1	Unsuccessful transaction	Fail
2	List of errors	Error
3	System error, unable to work	System Error

Appendix B.

Result Code of Transaction No. 2

Response Code	Description	Remarks
200	Successful transaction.	Success
1001	Invalid parameter data.	Invalid Parameter
1002	Invalid merchant code.	Invalid Merchant Code
1003	Invalid API Key.	Invalid API Key
1004	Bank Channel Code is incorrect.	Invalid Channel ID
1005	Invalid Route No..	Invalid Route No
1006	Invalid Order Number.	Invalid Order No
1007	Invalid amount.	Invalid Amount
1008	Customer ID / Customer Name is incorrect.	Invalid Customer ID
1009	Invalid mobile phone number.	Invalid Mobile Phone Number
1010	Invalid language code.	Invalid Language Code
1011	Invalid Checksum code.	Invalid Checksum
1012	Invalid reference number (Transaction ID).	Invalid Transaction ID
1013	Invalid payment details.	Invalid Description
1014	Invalid Token Flag information.	Invalid Token Flag
1015	Invalid currency code information.	Invalid Currency Code
1016	Invalid Credit Card Token Information.	Invalid Credit Card Token
1017	Incorrect installment month information.	Invalid Installment Month
1018	Invalid IP Address information	Invalid IP Address
1019	Invalid Shop ID information	Invalid Shop ID
1020	Invalid URL information of product image	Invalid Product Image URL

1021	Invalid card type data.	Invalid Card Type
2001	Merchant account has not been approved.	Account Unauthorized
2002	Merchant API Key has not been approved.	Invalid Merchant API Key
2003	Checksum data is incorrect.	Invalid Checksum Data
2004	No data found for the payment route of the merchant.	Invalid Merchant Routing
2005	The amount paid is less than the minimum amount specified.	Payment minimum price {0} less than {1} E.g., Payment minimum price 15.00 less than 20.00
2006	Amount paid more than the maximum amount specified.	Payment maximum price {0} is over than {1} E.g., Payment maximum price 55000.00 is over than 50000.00
2007	Unable to create a payment receipt item.	Request Transaction Error
2008	Unable to create a payment Token list.	Request Transaction Token Error
2009	Unable to find the desired item.	Transaction Not Found
2010	No credit card Token information for payment.	Credit Card Token Not Found
2011	No currency code found for payment.	Currency Code Not Supported
2012	No monthly installments were found.	Installment Month Not Found
2013	Mobile Phone No. not registered with K PLUS Mobile Application.	Mobile Phone No. not registered with K PLUS Mobile Application.
2014	Invalid / Incorrect email	Invalid email address.
3001	System error, unable to complete the transaction.	System Error
3002	System is unavailable at xx:xx:xx - xx:xx:xx o'clock. Please try again later.	System is unavailable at xx:xx:xx - xx:xx:xx o'clock. Please try again later.
9001	KPLUS System Maintenance Time	KPLUS System Maintenance Time

Appendix C.

The result code of the customer payment (Payment information from the bank)

Response Code	Description	Remark
0	Complete payment transaction.	Success
1	Failed to make payment transaction.	Fail
2	The customer cancelled payment transaction.	Cancel
3	An error occurred during the payment transaction.	Error
9	Transaction waiting to payment.	Transaction Pending
20	Successful Void transaction.	Void Success
21	Complete Refund transaction.	Refund Success
22	Make a transaction to request a Refund.	Request to Refund
23	Complete the transaction to transfer money to the merchant.	Settlement Success
24	Void transaction failed.	Void Fail
25	Refund transaction is unsuccessful.	Refund Fail

Appendix D.

Currency Code

No.	Code (ISO Code)	Currency
1	764	THB - Thai Baht
2	840	USD - US dollar
3	978	EUR - Euro
4	392	JPY - Japanese Yen
5	826	GBP - Pound Sterling (UK)
6	036	AUD - Australian Dollar
7	554	NZD - New Zealand Dollar
8	344	HKD - Hong Kong Dollar
9	702	SGD - Singapore Dollar
10	756	CHF - Swiss Franc
11	458	MYR - Malaysian ringgit
12	156	CNY - Chinese yuan

Appendix E.

The payment status of the Credit Card Inline system.

Payment Status	Description	Remark
Paid	Payment successful.	Payment successful. You can now display the payment result page.
WaitAuthorize	Invalid parameter.	Payment has not been completed yet. It is awaiting confirmation of payment (Confirm OTP). The merchant needs to redirect the customer to the bank's OTP confirmation page to confirm the OTP before proceeding.