Table of Contents

Introduction

Trust My Travel work differently to a normal payment company and ARE NOT a payment gateway. Our value add is the fact that we build in financial protection to all transactions meaning that our acquiring partners and the cardholders are protected against the insolvency of our providers.

Due to our unique proposition we capture a lot more information than a traditional payment provider, and we also do not just sit one MID behind a provider. As such we ask that you do not treat this implementation as a payment gateway integration or make assumptions about this integration based on previous integrations of payment providers.

Overview

To begin using the Trust My Travel Payment Modal, you will need:

- A TMTProtects account
- A channel on your TMTProtects account that is ready for processing (nb this can be a channel in test mode).
- A secret key for that channel.
- The base currency of the channel
- Your TMTProtects site path

See the Channel Data page and Site Path page for further detail.

Authorisation

To obtain a valid token for transacting, and to prevent tampering with transaction data, an authorisation string must be created for each transaction and hashed and salted with your channel secret. A GMT timestamp in the required format must be included in this string and appended to it. Authorisation strings are valid for 15 minutes. Please see the Authorisation page page for further detail and code examples.

Scripts

Having obtained these details, you will need to include the following script on your checkout page. This script must always be loaded from tmtprotects.com

<script src="https://payment.tmtprotects.com/tmt-payment-modal.3.5.0.js"></script>

Beneath this script, you will need a script that inits the TMT Payment Modal and passes in your path and either formId or data values depending on the implementation that you are using.

Form implementation:

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path: 'test-site',
            formId: 'tmt-payment-form'
        })
    }
</script>
```

Object implementation:

```
<script>
    window.tmtPaymentModalReady = function () {
```

```
            const data = {
                ...
            }

            const button = document.getElementById('trigger-modal');

            button.addEventListener('click', function () {
                const tmtPaymentModal = new window.tmtPaymentModalSdk({
                    path: 'test-site',
                    data: data
                })
            })
        }
</script>
```

For examples of further options that can be passed to the TMT Payment Modal, please See the options page

NB: If you are serving up the payment modal from within an iframe, you will need to define the iframe origin as well as all ancestors via the origin option

Alternative Payment Methods

It is now possible to process non-credit-card payments. By default, the modal will only serve up the credit card interface, but you can configure it to offer any, or all, of our alternative payment methods as well as, or instead of, taking credit card payments. See the options page for configuration details, and the Alternative Payment Methods page for the individual requirements of each method.

Implementation

In order to pass booking and transaction data to the modal, you will need to choose from one of our Payment Modal implementations. Select the method that suits your workflow best. The basic callbacks for this process are covered below. See the callbacks page for details on all available callbacks.

End to End Process

From the point at which the user clicks to pay, and triggers the modal, the following events occur:

- User clicks pay
- Modal is triggered
- Modal validates that all required data is present
- Modal attempts to obtain a token using the authstring
- Modal attempts to create a booking using the booking data provided
- Modal executes booking_logged callback
- Modal renders payment form
- User adds credit card details and clicks pay
- Transaction attempted
- If transaction is subject to 3DS2, the relevant authentication is served to the user
- If transaction is not a credit card payment, the relevant third party form is served up to the user

If the transaction is successful, the transaction_logged callback is triggered with the full API response to the `POST /transactions` call performed by the modal.

If the card issuing bank declines the card for some reason the transaction_failed callback is triggered with the full API response to the `POST /transactions` call performed by the modal.

After the completion of any of the scenarios detailed above, the user experience is now back in your hands, and it is up to you to close the modal, and validate the response.

Closing the Modal

The Payment Modal comes with a `closeModal` method that allows you to programtically close the modal from the same page that it was triggered from.

```
window.tmtPaymentModalReady = function () {
    var tmtPaymentModal = new window.tmtPaymentModalSdk({
        path: 'test-site',
        formId: 'tmt-payment-form'
    })
    tmtPaymentModal.on('transaction_logged', function (data) {
        // Call AJAX functions to update database
        ...
        tmtPaymentModal.closeModal();

        // Redirect to success or fail page.
    })
})
```

Troubleshooting

Should you encounter any issues in getting through the End to End process, please do the following in the order shown:

- Consult the Troubleshooting section of this documentation
- Refer to the TMT Status Page to ensure that your issue is not an open bug
- Contact TMT Member Support with as much detail as possible on the issue

Release Notes

- Prevent AMEX transactions on non-supported currency
- Rename currency and language picker IDs to prevent duplicate declarations
- Fixed bug where `transaction_logged` callback is triggered twice
- Improvements to 3DS validation outcome look-up
- Enhancements for use with new payment gateway

Browser Support

The current version of the Payment Modal has been tested in latest versions of Chrome, Firefox, Safari, Edge and IE11.

Previous Versions

Previous versions of the Payment Modal and the related documentation can be found on our demos site

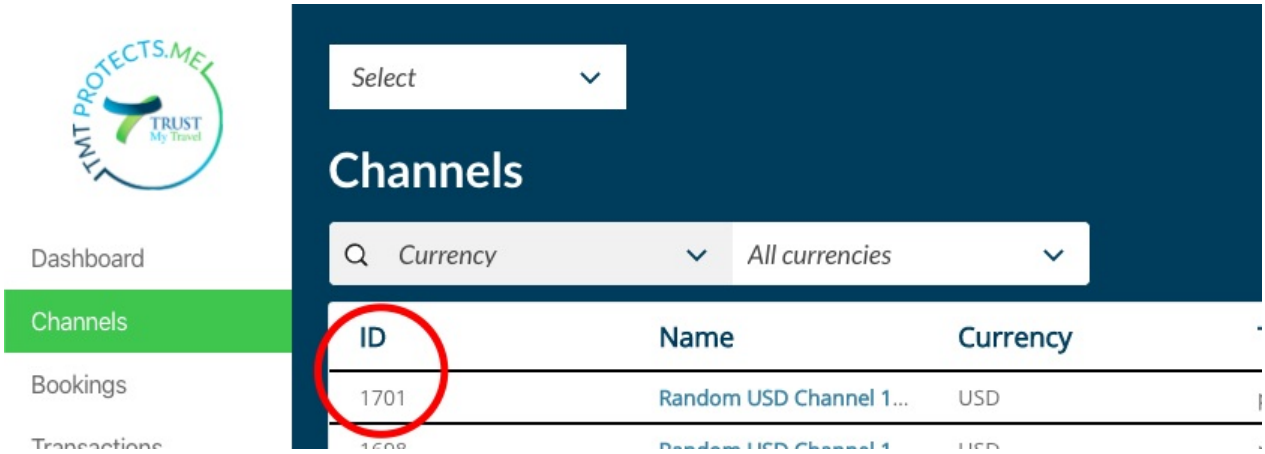
Support

Please subscribe to https://status.tmtprotects.com/ for updates on new versions of the modal. Please direct all issues and questions to membersupport@trustmytravel.com supplying as much detail as possible such as including links to pages where the modal is being implemented or code examples.

Channel Data

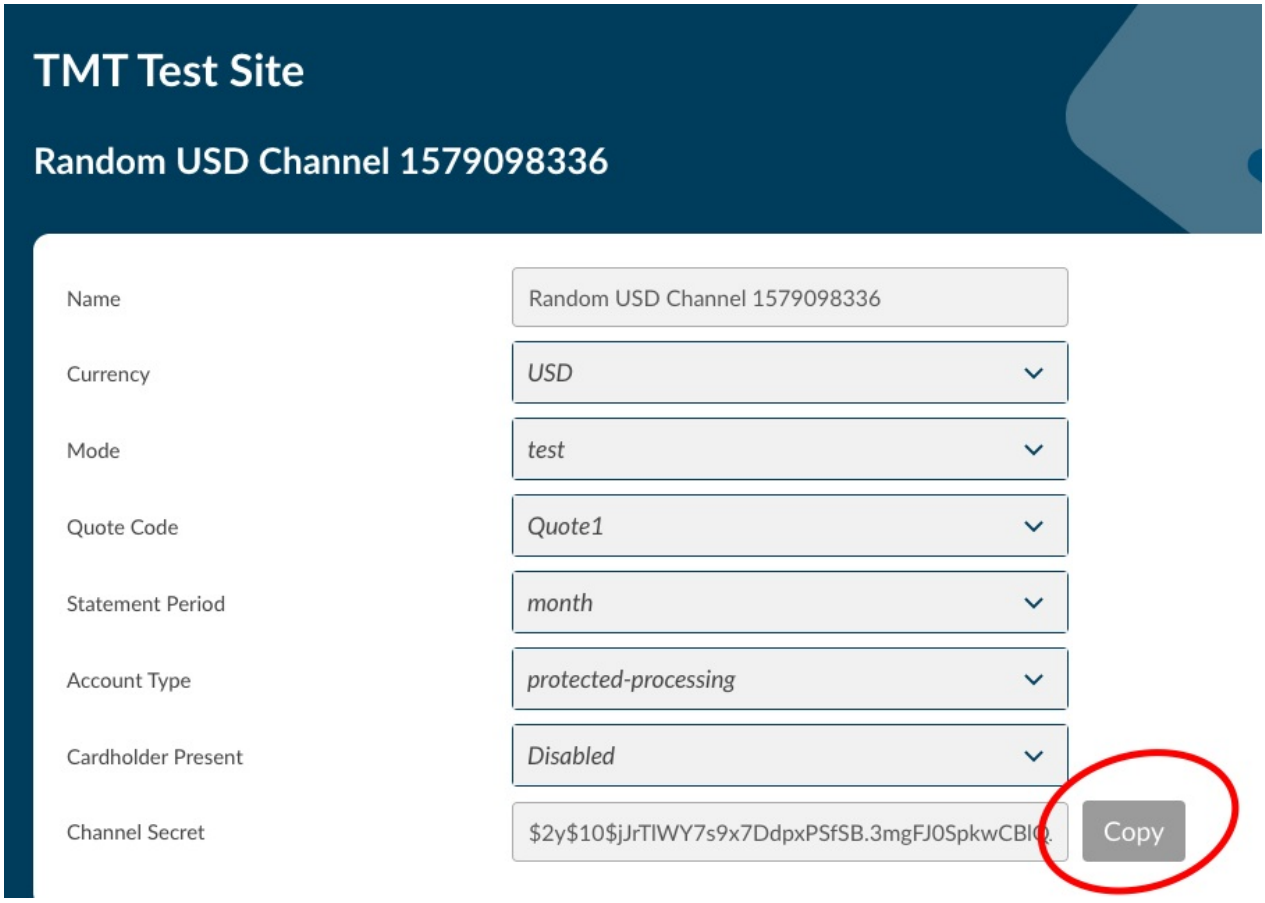
Channel ID and Secret

You can obtain the ID and Channel Secret for the channel you wish to integrate by logging into the TMT dashboard and going to the channels page. Here the IDs of all available channels are shown



In this example, the channel ID is 1701

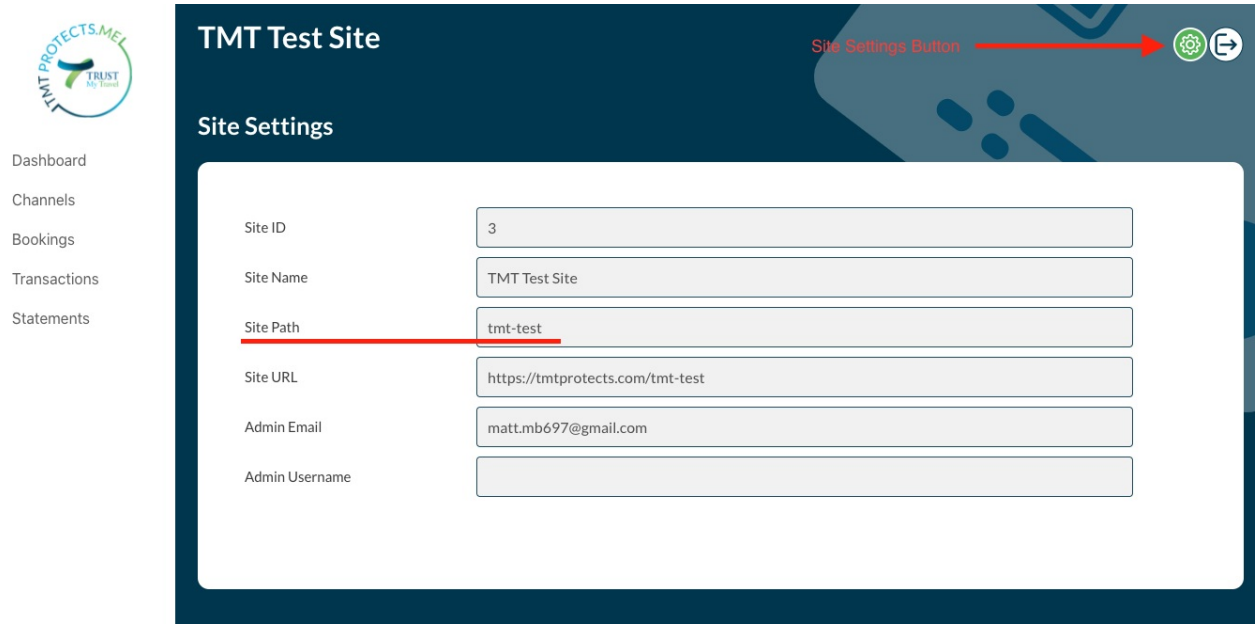
Click on the channel to open it and then click on the blue "View" button in the "Channel Secret" row. This will reveal the channel secret and a button to copy it clipboard.



Site Path

To find your site path, login to the TMTProtects Dashboard and click on the site settings button as shown in the screenshot below. This will reveal your site settings including the "Site Path".

This value should be passed to the Payment modal via the path option.



Authorisation

The authorisation string should be generated by taking the steps listed below. Code examples are provided further down the page along with PHP helper classes.

To test the authorisation string your app generates matches the expected authorisation string, or to assist with debugging why an authorisation string is not working you can use the Auth Test Demo

The steps required to generate the authorisation string are as follows:

- Generate a GMT datetime stamp with format: YmdHis
- Concatenate the following values, in the order shown, into a query string:
  - Channel ID
  - Channel base currency
  - Transaction total in base currency
  - The GMT datetime stamp
- Hash the string using sha256
- Append the channel secret to the hashed string and hash again, again with sha256
- Append the datetime stamp to this hashed and salted string

Extending Authorisation

If there are other items included in the transaction that you wish to insure against tampering, these values can also be included in the authstring.

The following fields can be included in any implementation:

- country
- date
- email
- firstname
- reference
- surname

You can also include the following in the Data Object Implementaion

- allocations
- charge_channel

IMPORTANT

- If you include additional values in the authstring, you must declare them via the Verify Option.
- Values must be concatenated in alphabetical order of the field they relate to with the timestamp appended afterwards
- If you are including allocations in your authstring, the order of the fields in the allocation objects must match the order of the fields passed to the init method.
- Arrays must be json encoded

Examples

If the language you use does not have examples shown, please send a request to techsupport@trustmytravel.com indicating the coding language you are using.

- PHP
- Node.js

PHP

A `TmtAuthstring\Create` class is available on the TMT Github Page along with instructions on implementation, or you can write your own using the examples below:

Basic Implementation

```
// Get current time in GMT.
$time_now = new DateTime('now', new DateTimeZone('GMT'));

// Create timestamp in 'YmdHis' format. E.g. 20190812055213
$timestamp = $time_now->format('YmdHis');

// Concatenate the values for channels, currencies, total and your timestamp in that order.
$booking_vars = [
    'channels'   => 2,
    'currencies' => 'USD',
    'total'      => 9999,
    'timestamp'  => $timestamp,
];

$string = implode('&', $booking_vars);

// SHA256 the string.
$auth_string = hash( 'sha256', $string );

// Fetch your channel secret and concatenate to string.
$secret = 'MYCHANNELSECRET123';
$salted_auth_string = hash( 'sha256', $auth_string . $secret );

// Concatenate with timestamp.
$final_auth_string = $salted_auth_string . $timestamp;
```

Extended

```
// Get current time in GMT.
$time_now = new DateTime('now', new DateTimeZone('GMT'));

// Create timestamp in 'YmdHis' format. E.g. 20190812055213
$timestamp = $time_now->format('YmdHis');

// Concatenate the values in alphabetical order then append timestamp.
$booking_vars = [
    'allocations'    => json_encode([
        [
            'channels'      => 23,
            'currencies'    => 'GBP',
            'operator'      => 'flat',
            'total'         => 1000,
        ],
    ]),
    'channels'      => 2,
    'currencies'    => 'USD',
    'reference'     => 'SOMEREFERENCE',
    'total'         => 9999,
];

$booking_vars['timestamp'] = $timestamp;

$string = implode('&', $booking_vars);

// SHA256 the string.
$auth_string = hash( 'sha256', $string );

// Fetch your channel secret and concatenate to string.
$secret = 'MYCHANNELSECRET123';
$salted_auth_string = hash( 'sha256', $auth_string . $secret );
```

```
// Concatenate with timestamp.
$final_auth_string = $salted_auth_string . $timestamp;
```

Node JS

```
// Get current time in GMT.
const date = new Date();
const utcDate = new Date(date.getUTCFullYear(), date.getUTCMonth(), date.getUTCDate(), date.getUTCHours(), date.getUTCMinutes(
), date.getUTCSeconds());

// Create timestamp in 'YYYYMMDDHHmmss' format. E.g. 20190812055213
const timestamp = format(utcDate, 'YYYYMMDDHHmmss')

// Concatenate the values for channels, currencies, total and your timestamp in that order.
const bookingVars = {
    channels: 2,
    currencies: 'USD',
    total: 9999,
    timestamp: timestamp
}

let string = []
for (const key in bookingVars) {
    string.push(bookingVars[key])
}
string = string.join('&')

// SHA256 the string.
const encode = crypto.createHash('sha256').update(string).digest('hex')

// Fetch your channel secret and concatenate to string.
const { CHANNEL_SECRET } = 'MYCHANNELSECRET123'

const authString = crypto.createHash('sha256').update(
    Buffer.concat([
        new Buffer(encode),
        new Buffer(CHANNEL_SECRET)
    ])
).digest('hex')

// Concatenate with timestamp.
const appAuthString = authString + timestamp;
```

Options

Mandatory

The following option is mandatory and must be included for the payment modal to function correctly:

- path

You must also include one of these options:

- formId
- data

Test Environments

If you are using a channel that is in test mode, you will need to add the environment option.

Iframes

If you are serving up the payment modal from within an iframe, all ancestors in the chain must be defined using the origin option.

Optional

The following options can be used where required

- paymentMethods
- paymentCurrency
- lang
- disableLang
- disableCloseWindowPrompt
- debug
- verify
- transactionType
- installments

Examples

path

The path of your TMTProtects site.

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path: 'test-site',
            ...
        })
    }
</script>
```

formId

The ID of the form containing the required booking and transaction data. See Payment Form Implementation for more detail.

```
<form id="myPaymentForm" action="complete.php" method="post">
    // Form inputs etc...
</form>

<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path: 'myPaymentForm',
            ...
        })
    }
</script>
```

data

An object containing all required booking and transaction data. See Data Object Implementation for more detail.

```
<button id="trigger-modal" class='btn btn-primary'>Pay Now</button>

<script>
    window.tmtPaymentModalReady = function () {

        const data = {
            booking_id: '0',
            channels: '2',
            country: 'GB',
            // Authentication
            booking_auth: authentication_string,
            // Lead Traveller
            firstname: 'John',
            surname: 'Smith',
            email: 'john.smith@example.org',
            date: '2020-05-15',
            // Payment details
            payee_name: 'Jane Smith',
            payee_email: 'jane.smith@example.org',
            payee_address: '123 test address',
            payee_city: 'Test city',
            payee_country: 'GB',
            payee_postcode: '0000',
            currencies: 'GBP',
            total: '9999'
        }

        const button = document.getElementById('trigger-modal')

        button.addEventListener('click', function () {
            var tmtPaymentModal = new window.tmtPaymentModalSdk({
                path: 'test-site',
                data: data
            })
        })
    }
</script>
```

Environment

Different versions of the tokeniser tool are used according to whether a channel is in test mode or not. If the channel you are implementing the modal for is in test mode, you will need to set the environment option accordingly. For other modes, the environment option will default to live.

NB: While you are permitted to use non-secure URLs in test mode, you will not be permitted to do so in live mode

```
<script>
```

```
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path: 'test-site',
            formId: 'my-payment-form',
            environment: 'test'
        })
    }
</script>
```

Origin

If you are serving up the payment modal from within an iframe, all ancestors in the chain must be defined in a comma separated list with the parent listed first, followed by the origin that will render it and so on up the chain.

As an example, foo.com is an iframe that is serving up the modal within a file on bar.com:

```
<body>
    <!--content served up by bar.com-->

    <iframe src=foo.com>
        <button id="trigger-modal">Pay</button>

        <script>
            window.tmtPaymentModalReady = function () {

                const button = document.getElementById("trigger-modal");

                button.addEventListener("click", function () {
                    const tmtPaymentModal = new window.tmtPaymentModalSdk({
                        path: "some-site",
                        origin: "foo.com,bar.com",
                        data: {
                            ...
                        }
                    });
                });
            }
        </script>
    </iframe>
</body>
```

paymentMethods

By default, the payment modal offers payment via credit card. However, we also offer the following payment methods that can be used in addition to, or instead of, credit card payments:

- Alipay
- Giropay
- iDEAL
- DLocal (Installments)
- Rapipago
- Sofort

To indicate the methods you want to use, pass them in as an array as per the examples below. Payment methods will appear in the order you define them, with the exception of "credit-card", which will always be the default interface if included. Please review the Alternative Payment Methods page page for further details on these methods.

Using all available payment methods

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
```

```
            paymentMethods: [
                'credit-card',
                'alipay',
                'dlocal',
                'giropay',
                'ideal',
                'sofort',
                'rapipago'
            ]
        })
    }
</script>
```

Using Alipay and Credit Card

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            paymentMethods: [
                'credit-card',
                'alipay'
            ]
        })
    }
</script>
```

Using Giropay and Alipay with Giropay as default

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            paymentMethods: [
                'giropay',
                'alipay'
            ]
        })
    }
</script>
```

paymentCurrency

The default behaviour of the payment modal is to offer payment in the base currency of your channel, and allow the customer to change the payment currency as required. Should you know that the customer making payment is based in a country that does not use your channel's base currency, you can improve the user experience by defining their currency to default the payment modal to.

For example, if the base currency of your channel is USD and your customer is based in Germany, you would define the paymentCurrency as EUR as shown below.

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            paymentCurrency:'EUR'
        })
    }
</script>
```

Should you wish to display your prices in currencies other than your base currency, you will need to utilise your channel's forex feed

lang

The modal is rendered in English by default. Should you know that the user prefers an alternate language, the modal can be set to load in that language should a translation be available. Once loaded, the user is still free to switch languages should they wish. The language which the modal is in at the point of transaction determines what language the user's payment receipt shall be in.

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            lang: 'ptBR'
        })
    }
</script>
```

disableLang

If the translations you require for your customer base are not available, you can disable the translation picker.

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            disableLang: true
        })
    }
</script>
```

disableCloseWindowPrompt

If you have your own means of handling user attempts to close the browser or refresh during transaction you may wish to disable the in-built `onbeforeounload` close window prompt.

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            disableCloseWindowPrompt: true
        })
    }
</script>
```

debug

Set `debug = true` to enable validation and error logs in console.

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            debug: true
        })
    }
</script>
```

Verify

If you have extended the authstring to include other booking and transaction values, you will need to include the `verify` option in order to pass in an array of the fields that you have included in the authstring.

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            verify: ["reference"]
        })
    }
</script>
```

The following fields can be used in the `verify` array in any implementation:

- country
- date
- email
- firstname
- reference
- surname

You can also include the following in the Data Object Implmentaion

- allocations
- charge_channel

Transaction Type

We now allow for pre-authorizing a card via the modal leaving you to make a Capture request via an API call in order to capture the payment.

NB: If you intend to use this option, please note the following:

- An authorize transaction will not result in funds being removed from the customer's account. You must complete a capture request in order to complete the transaction
- Authorize transactions are subject to a per transasction fee as are capture transactions
- Authorize transactions can only be captured within a short time frame. This is generally up to 5 days but can differ according to the bank processing the payment. We would advise that you remain well inside 5 days for this to avoid losing transactions
- Allocations are not permitted on authorize transactions and must be included with the Capture request

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            transactionType: 'authorize'
        })
    }
</script>
```

Installments

If you are offering installments as an alternative payment option, you can control which of the available installment options are offered by passing the required value or values in an array. This can be used to present the user with a pre-defined installment value, or to hide the installments interface altogether (this can be useful if you know that your customer has a Brasilian bank card but want to run a normal credit card payment).

Example: No Installments

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            paymentMethods: ['dlocal']
            installments: [1]
        })
    }
</script>
```

Example: Set Installment Plan to 3 Installments

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            paymentMethods: ['dlocal']
            installments: [3]
        })
    }
</script>
```

Example: All Installment Options (default)

```
<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
            path:'test-site',
            formId: 'my-payment-form',
            paymentMethods: ['dlocal']
            installments: [1, 3, 6, 9, 12]
        })
    }
</script>
```

Alternative Payment Methods

Credit card payments are available in all the currencies Trust My Travel support and made using the data required by the modal on initialisation. The alternative payment methods that we offer are restricted by currency and further considerations are needed regarding the data supplied when triggering the modal.

Payment Currency

Alternative payment methods are only available in the payment currencies defined below. If any of these methods are made available to the customer and they select one, the payment currency will automatically switch to the currency linked to the payment method.

APM	Payment Currency
Alipay	USD
Giropay	EUR
iDEAL	EUR
DLocal	BRL
Rapipago	ARS
Sofort	EUR

The same applies if the paymentCurrency option is defined.

For example, if the modal is launched with USD as payment currency and the user has the option to select Giropay, then the payment currency will switch to EUR.

Authorize Transaction Type is Ignored

Alternative payment methods only allow for purchase transactions. It is not possible to pre-authorize an alternative payment method transaction. As such, if the modal is triggered with the transactionType option set to "authorize" and the user selects an alternative payment, the transactionType option will be ignored.

Testing

If you are using Google Chrome, you can right click and select 'Translate to English' (or the language of your browser).

Alipay Success

In the third party payment window, login with:

- Username: `alipaytest20091@gmail.com`
- Password: `111111`

Add `111111` as the Alipay password and click confirm.

Alipay Fail

Not yet available

NB *Certain companies cannot use Alipay. Please check if your industry is featured on their restricted list - if it is here TMT will be unable to support Alipay on your account.*

Giropay Success

In the third party payment window, login with:

- Username: `chiptanscatest2`
- Password: `12345`
- Click Pay Now
- Click Continue
- Enter `123456` as TAN and click Log in

Giropay Fail

In the third party payment window click on the Abort button.

iDEAL Success

Run a transaction for any value other than EUR 2 and click the Confirm Transaction button.

iDEAL Fail

Run a transaction for EUR 2 and click the Confirm Transaction button.

Installments Success:

Use `4242 4242 4242 4242` as credit card number, a valid expiry date and `100` as PIN. Pick any number of installments and enter an ID number.

Installments Fail:

Use `4242 4242 4242 4242` as credit card number, a valid expiry date and `101` as PIN. Pick any number of installments and enter an ID number.

Rapipago Success

There is no third party sandbox payment window available for this so success is triggered behind the scenes. For a success, run a transaction for any value greater than or equal to ARS 50.

Rapipago Fail

There is no third party sandbox payment window available for this so success is triggered behind the scenes. For a fail, run a transaction for any value less than ARS 50.

Sofort Success

In the third party payment window, login with:

- Account number: `88888888`
- the PIN field: `123456`

Select any account on the next page and then click Next.

- Enter 12345 in the TAN field and click Next.

Sofort Fail

Unavailable.

Individual Requirements

Rapipago

Rapipago transactions are only permitted where the payer is in Argentina. This is indicated by passing the ISO country code for Argentina, `AR`, to the modal using either the value of an element with class `tmt_payee_country` or the value of the field `payee_country` passed in via an object.

Giropay

Giropay transactions require the inclusion of a description of what the transaction is for. If a booking description has been passed to the modal using either the value of an element with class `tmt_description` or the value of the field `description` passed in via an object then this will be used with the transaction request.

If no information is supplied via these fields, then the description passed to Giropay defaults to:

- "COMPANY - sale"

Where COMPANY is the value of the channel receipt label if one exists, and if not, the name of the site.

Implementations

There are two ways in which you can pass booking and transaction data to the Payment Modal.

Payment Form

Create a form with the required fields defined and pass the ID of the form to the Payment Modal SDK. On submitting the payment form, the modal is triggered, a booking is placed using the data present on the payment page and the user is prompted for their credit card details in order to complete the transaction.

- Payment Form Implementation
- Payment Form Demo, minumum options
- Payment Form Demo, full options

Data Object

An object containing all required fields can be passed to the Payment Modal SDK. An event listener is also added, the modal is triggered when the nominated event is triggered, a booking is placed using the data present on the payment page and the user is prompted for their credit card details in order to complete the transaction.

- Data Object Implementation
- Data Object Demo, minumum options
- Data Object Demo, full options

Payment Form

For this implementation, you will need a form with the fields defined below present on the page with the css class properties shown (either hidden from or displayed to the user).

On triggering the modal, a booking is placed using the data present in the form and the user is prompted for their credit card details in order to complete the transaction.

If any of the required data is not present, an error is output detaling the data that is not present.

If transaction fails for some reason, the user is given a further two attempts to make payment before the transaction is failed permanently.

On successful transaction, the user is shown the success dialog.

Required Data and Relevant CSS Class

All transactions made using the payment form must have the following:

CSS Class	Description
tmt_booking_auth	The hashed and salted authorisation string for the transaction
tmt_booking_id	Set this to 0 to create a new booking, or an existing booking ID if you preloaded a booking
tmt_channels	Set this to the ID of the channel you wish to use for the transaction
tmt_payee_name	The name of the person making payment as it appears on their credit/debit card
tmt_payee_email	The email of the person making payment
tmt_payee_country	The ISO 3166-1 alpha-2 value of the country of the person making payment
tmt_currencies	The ISO 4217 value for the currency the travel item is being sold in (must match the currency of the channel in use)
tmt_total	The total being billed in the currency of the channel in use as a cent value (e.g. $10.00 = 1000)

New Bookings

Unless you have preloaded a booking, you will also have to supply the following booking specific fields, which will be used to create a new booking prior to the transaction:

CSS Class	Description
tmt_country	The ISO 3166-1 alpha-2 value of the country the booking takes place in
tmt_firstname	The firstname of the lead traveller
tmt_surname	The surname of the lead traveller
tmt_email	The email address of the lead traveller
tmt_date	The end date of travel in YYYY-MM-DD format

Address Data and Relevant Class Name

If your account is NOT enabled for Cardholder Present (all accounts are disabled for Cardholder Present by default) then you will need to supply the fields shown in the address data table. It is up to you to validate that the end user has completed the address fields prior to triggering the modal. Failing to validate these fields will result in a developer error being output in the event of an end user not completing them.

CSS Class	Description
tmt_payee_address	The adress of the person making payment
tmt_payee_city	The city of the person making payment
tmt_payee_postcode	The postcode/zip of the person making payment

Optional Data and Relevant Class Name

Class	Description
tmt_reference	Your own reference
tmt_description	A description of the product being sold
tmt_pax	The amount of people the product is for

Payment Form Example

```
<form id="myPaymentForm" action="complete.php" method="post">

    <div class="row">

        <div class="col-sm-8">

            <div class="form-group">
                <div class="col-sm-12">
                    <h2>Billing</h2>
                </div>
            </div>

            <div class="form-group">

                <div class="col-sm-6">
                    <label for="payee_name">Payee Name</label>
                    <input name="payee_name" type="text" class="form-control tmt_payee_name">
                </div>

                <div class="col-sm-6">
                    <label for="email">Email</label>
                    <input type="email" class="form-control tmt_payee_email">
                </div>

            </div>

            <div class="form-group">
                <div class="col-sm-12">
                    <label for="address">Address</label>
                    <input name="address" type="text" class="form-control tmt_payee_address" value="" />
                </div>
            </div>

            <div class="form-group">
                <div class="col-sm-12">
                    <label for="city">City</label>
                    <input name="city" type="text" class="form-control tmt_payee_city" value="" />
                </div>
            </div>

            <div class="form-group">
                <div class="col-sm-8">
                    <label for="country">Country: </label>
                    <select name="country" class="form-control tmt_payee_country">
                        <option value="US">United States of America</option>
                        <option value="GB">United Kingdom</option>
                        <option value="AU">Australia</option>
                    </select>
```

```
                </div>
                <div class="col-sm-4">
                    <label for="zip">Zip / Postcode: </label>
                    <input name="zip" type="text" class="form-control tmt_payee_postcode" value="" />
                </div>
            </div>

            <div class="form-group">

                <div class="col-sm-12">
                    <input id="tmt-pay" type="submit" value="Pay" name="pay" class="btn btn-primary btn-block" />
                </div>
            </div>

        </div>

        <div class="col-sm-4">

            <div class="form-group">
                <div class="col-sm-12">
                    <h2>Your Cart</h2>
                </div>
            </div>

            <div class="form-group">
                <div class="col-sm-12">

                    <ul class="list-group">
                        <li class="list-group-item">
                            <div>
                            <h6 class="my-0">Guided Tour of Big Ben</h6>
                            <small class="text-muted">3 hour guided tour of Britain's most famous timepiece.</small>
                            </div>
                            <span class="text-muted">£22</span>
                        </li>
                        <li class="list-group-item">
                            <div>
                            <h6 class="my-0">Stand-up Paddleboard the Thames</h6>
                            <small class="text-muted">Who needs the ocean when you can paddle down the charming River Thames?<
/small>
                            </div>
                            <span class="text-muted">£9</span>
                        </li>
                        <li class="list-group-item">
                            <span>Total</span>
                            <input type="hidden" class="tmt_currencies" value="GBP" />
                            <strong>GBP 31</strong>
                            <input type="hidden" class="tmt_total" value="3100" />
                        </li>
                    </ul>
                </div>
            </div>
        </div>

        <!-- HIDDEN VALUES -->
        <input type="hidden" class="tmt_booking_id" value="0" />
        <input type="hidden" class="tmt_channels" value="4" />

        <!-- BOOKING DETAILS -->
        <input name="firstname" type="hidden" class="form-control tmt_firstname" value="John">
        <input name="surname" type="hidden" class="form-control tmt_surname" value="Smith">
        <input name="email" type="hidden" class="form-control tmt_email" value="john.smith@example.org">
        <input name="country" type="hidden" class="tmt_country" value="GB" />
        <input type="hidden" class="tmt_date" value="2019-05-12" />
        <input type="hidden" class="tmt_booking_auth" value="<?php echo $final_auth_string; ?>" />
    </div>

</form>

<script>
    window.tmtPaymentModalReady = function () {
        var tmtPaymentModal = new window.tmtPaymentModalSdk({
```

```
            path:'test-site',
            formId:'myPaymentForm'
        })
    }
</script>
```

Data Object

For this implementation, an object containing required properties as defined below is passed to the modal.

On triggering the modal, a booking is placed using data from the object and the user is prompted for their credit card details in order to complete the transaction.

If any of the required data is not present, an error is output detaling the data that is not present.

If transaction fails for some reason, the user is given a further two attempts to make payment before the transaction is failed permanently.

On successful transaction, the user is shown the success dialog.

Required Data

All transactions made using the data object must have the following:

Key	Description
booking_auth	The hashed and salted authorisation string for the transaction
booking_id	Set this to 0 to create a new booking, or an existing booking ID if you preloaded a booking
channels	Set this to the ID of the channel you wish to use for the transaction
payee_name	The name of the person making payment as it appears on their credit/debit card
payee_email	The email of the person making payment
payee_country	The ISO 3166-1 alpha-2 value of the country of the person making payment
currencies	The ISO 4217 value for the currency the travel item is being sold in (must match the currency of the channel in use)
total	The total being billed in the currency of the channel in use as a cent value (e.g. $10.00 = 1000)

New Bookings

Unless you have preloaded a booking, you will also have to supply the following booking specific fields, which will be used to create a new booking prior to the transaction:

Key	Description
country	The ISO 3166-1 alpha-2 value of the country the booking takes place in
firstname	The firstname of the lead traveller
surname	The surname of the lead traveller
email	The email address of the lead traveller
date	The end date of travel in YYYY-MM-DD format

Address Data

If your account is NOT enabled for Cardholder Present (all accounts are disabled for Cardholder Present by default) then you will need to supply the fields shown in the address data table.

Key	Description
payee_address	The adress of the person making payment

payee_city	The city of the person making payment
payee_postcode	The postcode/zip of the person making payment

Optional Data

Key	Description
reference	Your own reference
description	A description of the product being sold
pax	The amount of people the product is for
allocations	See Allocations objects for more details
charge_channel	See Allocations objects for more details

Data Object Example

```
<button id="trigger-modal" class='btn btn-primary'>Trigger Payment Modal</button>

<script>
    window.tmtPaymentModalReady = function () {

        const data = {
            // Booking Data
            booking_id: '0',
            channels: '2',
            country: 'GB',
            date: '2020-05-12',
            currencies: 'GBP',
            total: '9999',
            reference: 'test reference', // optional
            description: 'Some holiday', // optional
            pax: '3', // optional,
            // Authentication
            booking_auth: hashed_salted_auth_string,
            // Lead Traveller
            firstname: 'John',
            surname: 'Smith',
            email: 'john.smith@example.org',
            // Payment details
            payee_name: 'Jane Smith',
            payee_email: 'jane.smith@example.org',
            payee_address: '123 test addres',
            payee_city: 'Test city',
            payee_country: 'GB',
            payee_postcode: '1234',
        }

        const button = document.getElementById('trigger-modal')
        button.addEventListener('click', function () {

            var tmtPaymentModal = new window.tmtPaymentModalSdk({
                path: 'test-site',
                data: data
            })
        })
    }
</script>
```

Allocations

The Data Object implementation also allows for allocating funds to alternative channels. These allocations can be flat amounts or percentages of the transaction total. You can also nominate which channel incurs our charges.

Allocation Object Fields

Key	Type	Description
channels	integer	The ID of the allocation channel
currencies	string	The currency of the allocation channel
total	integer	The total in cents or as a percentage to be allocated
operator	string	Either "flat" or "percent"

Additional Request Fields

Key	Type	Description
charge_channel	integer	The ID of the channel to deduct TMT's charges from. If not included, this will default to the main transaction channel

Examples

£10.00 of a total of $220.00 is being allocated to a channel with the ID: 23.

TMT's charges for the transaction will be deducted from this channel and not the master channel

```
{
    booking_id: '0',
    channels: 2,
    currencies: 'USD',
    total: '22000',
    ...
    allocations: [{
        channels: 23,
        currencies: 'GBP',
        operator: 'flat',
        total: 1000
    }],
    charge_channel: 23
}
```

5% of a total of $220.00 is being allocated to a channel with the ID: 23.

TMT's charges for the transaction will be deducted from this channel and not the master channel

```
{
    booking_id: '0',
    channels: 2,
    currencies: 'USD',
    total: '22000',
    ...
    allocations: [{
        channels: 23,
        currencies: 'GBP',
```

```
        operator: 'percent',
        total: 5
    }],
    charge_channel: 23
}
```

£10.00 of a total of $220.00 is being allocated to a channel with the ID: 23.

TMT's charges for the transaction will be deducted from the master channel with id = 2. There is no need to indicate this via the request as TMT payments are deducted from the master channel by default.

```
{
    booking_id: '0',
    channels: 2,
    currencies: 'USD',
    total: '22000',
    ...
    allocations: [{
        channels: 23,
        currencies: 'GBP',
        operator: 'flat',
        total: 1000
    }]
}
```

Notes

- Allocations are only permitted via the Data Object implementation
- Allocation data can be protected from tampering via the verify option
- If you are using the verify option, ensure you order allocation objects the same in the verification as the instantiation
- The channel that incurs TMT's charges must be left with sufficient funds to cover the cost of the charges.
- The total of all allocations + TMT's charges must not be greater than the transaction total.
- If you are setting the transactionType option to "authorize", you cannot include allocations.

Pre-Loading Bookings

Bookings can be created in advance of prompting the user for payment via the TMT API.

User Tokens

All API requests must include a valid JWT token. To obtain a token, perform an API request as follows, where `{username}` and `{password}` are the user credentials supplied to you by Trust My Travel:

Request

```
POST /wp-json/jwt-auth/v1/token HTTP/1.1
Host: https://tmtprotects.com/wp
Content-Type: application/json

{
  "username": "{username}",
  "password": "{password}"
}
```

Response

```
{
    "id": 2,
    "name": "testsiteadmin",
    "username": "testsiteadmin",
    "user_email": "testsiteadmin@example.org",
    "user_nicename": "testsiteadmin",
    "user_display_name": "testsiteadmin",
    "usertype": "member_admin",
    "type": "member_admin",
    "sites": [
        {
            "name": "TMT Test Site",
            "url": "http://tmtprotects.com/tmt-test",
            "path": "/tmt-test/"
        }
    ],
    "token": "eyJ0e...PiUmyY",
    "refresh_token": "eyJ0e...p37TI"
}
```

Add Booking

- `{path}` corresponds with your site path
- `{token}` corresponds with a User Token
- `{channel_id}` corresponds with the ID of the channel you wish to add the booking to
- `{channel_currency}` corresponds with the currency of the channel you wish to add the booking to

Request

```
POST /wp-json/tmt/v2/bookings HTTP/1.1
Host: https://tmtprotects.com/{path}
Content-Type: application/json
Authorization: Bearer {token}

{
    "firstname": "John",
    "surname": "Smith",
```

```
    "email": "john.smith@example.org",
    "date": "2028-08-12",
    "total": 1000,
    "currencies": "{channel_currency}",
    "channels": {channel_id},
    "countries": "GB"
}
```

Notes

- The `date` field is for the date of travel.
- The `countries` field pertains to the country the booking takes place in and must be a valid ISO 3166-1 alpha-2 value
- The `total` for the booking is in cents

Response

```
{
    "id": 3097,
    "trust_id": "3-3097",
    "author": null,
    "created": "2020-03-31 12:28:03",
    "modified": "2020-03-31 12:28:03",
    "status": "draft",
    "internal_id": 3097,
    "title": "Smith | john.smith@example.org",
    "content": null,
    "firstname": "John",
    "surname": "Smith",
    "email": "john.smith@example.org",
    "date": "2028-08-12",
    "pax": null,
    "reference": null,
    "total": 1000,
    "total_unpaid": 1000,
    "currencies": null,
    "countries": "GB",
    "country": "GB",
    "transaction_ids": [],
    "channels": null,
    "language": "enGB"
}
```

Notes For full detail on the bookings endpoint you can request the schema using the example below where `{token}` corresponds with a User Token and `{path}` corresponds with your site path.

```
OPTIONS /wp-json/tmt/v2/bookings HTTP/1.1
Host: https://tmtprotects.com/{path}
Authorization: Bearer {token}
```

Callbacks

In order to allow you to capture relevant API data as the modal process occurs, we provide the following callbacks:

- token_error
- booking_logged
- booking_exists
- booking_error
- transaction_logged
- transaction_failed
- ~~transaction_rejected~~
- transaction_timeout
- transaction_error
- modal_closed
- close_window_attempted

token_error

If you have incorrectly hashed and salted the Payment Modal auth string, or if the auth string has expired, an error message is output to the modal, and the token_error callback is triggered. The error response from the token endpoint is passed to the token_error callback.

```
// token_eror
{
    code: "jwt_auth_invalid_request"
    message: "Session has expired"
    data: {
        status: 403
    }
}
```

booking_logged

If you do not create a booking prior to the user arriving at your payment page, and therefore set the value of your `.tmt_booking_id` input to 0, a booking will be created prior to the transaction being attempted using the POST /bookings endpoint. The response to this request will be passed to the booking_logged callback. It is advisable to log the ID of the booking as this can be used to establish whether a transaction was successful or not if timeouts occur.

```
// booking_logged
{
    author: "24"
    channels: 84
    content: ""
    countries: "GB"
    created: "2019-08-12 10:29:15"
    currencies: "USD"
    date: "2020-05-12"
    email: "john.smith@example.org"
    firstname: "John"
    id: 2600
    modified: "2019-08-12 10:29:15"
    pax: 0
    reference: ""
    status: "draft"
    surname: "Smith"
    title: "Smith | john.smith@example.org"
    total: 999
    total_unpaid: 999
```

```
    transaction_ids: []
    trust_id: "21-2600"
    ...
}
```

booking_exists

This callback is triggered when passing a booking ID for an already existing booking to the modal.

You may wish to create bookings prior to the user arriving at the payment page using the POST /bookings endpoint and then include the ID for that booking in the input with class `.tmt_booking_id` or keyed with `booking_id` in a data object. The booking is looked up via the GET /bookings/ID endpoint and the response is passed to the booking_exists callback.

```
// booking_exists
{
    author: "24"
    channels: 84
    content: ""
    countries: "GB"
    created: "2019-08-12 10:29:15"
    currencies: "USD"
    date: "2020-05-12"
    email: "john.smith@example.org"
    firstname: "John"
    id: 2600
    modified: "2019-08-12 10:29:15"
    pax: 0
    reference: ""
    status: "draft"
    surname: "Smith"
    title: "Smith | john.smith@example.org"
    total: 999
    total_unpaid: 999
    transaction_ids: []
    trust_id: "21-2600"
}
```

booking_error

This callback is triggered instead of the `booking_logged` or `booking_exists` callbacks in the event of an error in the booking data provided. The error message is output to the modal and the booking_error callback is triggered with the error message passed as the single argument that the booking_error callback receives.

For example, the channel ID supplied corresponds with a channel that has GBP as its base currency, but the booking currency is supplied as USD.

```
{
    response: code: "rest_invalid_param"
    data: {
        status: 400,
        params: {
            channels: "Channel ID currency does not match nominated currency."
        }
    }
    message: "Invalid parameter(s): channels"
}
```

transaction_logged

This callback is triggered when the user has successfully completed a transaction. It includes the response from the POST /transactions endpoint. The response to this request will be passed as the single argument that the transaction_logged callback receives.


```
{
    3ds_response: {}
    adjustments: []
    api_urls: []
    author: "24"
    bin_number: "411111"
    bookings: [
        {
            id: 2605,
            total: 999,
            currencies: "USD",
            reference: ""}
    ]
    card_types: "visa"
    channels: 84
    content: "Succeeded!"
    countries: "US"
    created: "2019-08-12 10:39:26"
    currencies: "GBP"
    forex: []
    forex_rate: ""
    hash: "44a256f2e5150dc1d0341feb6346cef685e0c0a05d179757ab282298f31a8bb8"
    id: 2606
    ip_address: ""
    last_four_digits: "1111"
    linked_id: 0
    modified: "2019-08-12 10:39:29"
    payee_email: "matt.mb697@gmail.com"
    payee_name: "Matthew Bush"
    payee_surname: "Bush"
    payment_ids: [2607, 2608, 2609]
    payment_methods: "credit-card"
    psp: "spreedly"
    statement_batches: "WEEK-33-1-2019-test"
    status: "complete"
    title: "John Smith | john.smith@example.org | purchase"
    token: "VXOcZHZR2wSeOxAvPEGoPeB9Avp"
    total: 830
    total_remaining: 830
    transaction_types: "purchase"
    trust_id: "21-2606"
}
```


transaction_failed

When the user has attempted a transaction, but it has been rejected by the card issuing bank, the response from the POST /transactions endpoint will be passed as the single argument that the transaction_failed callback receives.

Example


```
{
    3ds_response: {}
    adjustments: []
    api_urls: []
    author: "24"
    bin_number: "510510"
    bookings: [
        {
            id: 2610,
            total: 999,
            currencies: "USD",
            reference: ""}
    ]
```

```
    card_types: "master"
    channels: 84
    content: "Unable to process the purchase transaction."
    countries: "GB"
    created: "2019-08-12 10:43:29"
    currencies: "GBP"
    forex: []
    forex_rate: ""
    hash: "cf6a7a4504568672f16101a342c67982ed42d9d3042a6c9a87bab93e0d29fcaa"
    id: 2611
    ip_address: ""
    last_four_digits: "5100"
    linked_id: 0
    modified: "2019-08-12 10:43:31"
    payee_email: "john.smith@example.org"
    payee_name: "John Smith"
    payee_surname: "Smith"
    payment_ids: [2612]
    payment_methods: "credit-card"
    psp: "spreedly"
    statement_batches: "WEEK-33-1-2019-test"
    status: "failed"
    title: "John Smith | john.smith@example.org | purchase"
    token: "VdOZCqLvFowUIfi2cZVQxfqLDqF"
    total: 830
    total_remaining: 0
    transaction_types: "purchase"
    trust_id: "21-2611"
}
```


transaction_rejected

- Deprecated

transaction_timeout

At the point at which the user has completed their credit card details and submitted the transaction, the transaction process is in motion. Unless there is an issue at the TMTProtects side, the transaction request will be relayed to the bank. Should a timeout occur between the bank responding to TMTProtects or TMTProtects honouring the Payment Modal API request, an error message is displayed on the modal informing the user that there was a timeout but payment may have been successful and informing them of the booking ID for the transaction. The transaction_timeout callback is also triggered with details of the timeout and the booking ID pertaining to the transaction supplied.

The booking can be looked up using the GET /bookings/ID endpoint. The response will include an array of linked transactions under the field `transaction_ids` the last value in this array will pertain to the most recent transaction.

The transaction can then be looked up using the GET /transactions/ID endpoint to ascertain if it was successful or not.


```
{
    booking_id: 2622
    message: "Request timed out"
    name: "TimeoutError"
}
```


transaction_error

If the transaction attempt failed due to connectivity issues with the card issuing bank or for any reason other than being rejected for anything other than the card issuing bank's criteria, the transaction_error callback is triggered with the error response being passed as the single argument

```
{
    name: TypeError
    message: Failed to fetch
}
```

modal_closed

Should the user close the modal at any stage in the process, the modal_closed callback is triggered and an object is passed as the single argument

```
{
    message: modal closed
}
```

close_window_attempted

Should the user close attempt to close their browser window while the transaction is being processed, the close_window_attempted callback is triggered and an object is passed as the single argument. The user is prompted to confirm the close to try and prevent a disconnection.

```
{
    message: 'User attempted to close browser window while transaction is being processed!'
}
```

Validating Modal Callback Data

Hash Verification

During the End to End Process, the transaction_logged or transaction_failed callbacks would have called with the transaction response passed to them. Should you wish to validate a response, you will need to obtain the values for `id`, `status` and `total` as well as the channel secret for the channel you are using.

From there, you can use the `TmtAuthstring\Validate` class on the TMT Github Page following the example shown.

Alternatively, you can write your own verification method based on the example below:

Example

```
$values = [
    'id'        => $id,
    'status'    => $status,
    'total'     => $total
];

$varString    = implode('&', $values);
$authString   = hash('sha256', $varString);
$validHash    = hash('sha256', $authString . $channel_secret);

if (hash_equals($hash, $validHash)) {
    // Valid hash.
};
```

API Verification

The "id" can be used to verify the transaction via a GET /transactions/id request to the TMTProtects API.

Forex

In order to display prices in currencies other than your channel's base currency, you can perform a GET request for your channel as shown in our API Documentation

As shown in the documentation, and below, the response object includes a field named `forex_feed`. This contains a `rates` object that contains all currencies available against your channel's base currency and the rate to apply to your base amount to obtain an amount in that currency.

Example

Your channel has a base currency of EUR, you are selling a product for EUR 99.99 and you wish to display a price in GBP.

Perform a GET Request for the channel `GET {{url}}/wp-json/tmt/v2/channels/{{channel_id}}`

```
var settings = {
    "url": "{{url}}/wp-json/tmt/v2/channels/{{channel_id}}",
    "method": "GET",
    "timeout": 0,
```

```
};
```

Obtain `response.forex_feed.rates.GBP.rate` using the documentation as an example, this would be `0.93881`

```
$.ajax(settings).done(function (response) {
    var rate = response.forex_feed.rates.GBP.rate;
});
```

Multiply your base cost of EUR 99.99 by the rate: `99.99 * 0.93881 = 93.8716119`

```
var paymentAmount = rate * baseAmount;
```

Round to the nearest cent value (rounding down from .5 where applicable) to get GBP 93.87.

```
var displayAmount = paymentAmount.toFixed(2);
```

Test Credit Cards

- Use the values below to test the various payment flows. Use any valid Year / Month combination.
- For the Challenge and 3DS1 Fallback flows, you will be shown a 3DS authentication simulator. The password for this simulator is `Checkout1!`

Credit Card Number	CVV	Flow	Outcome
4485 0403 7153 6584	100	Frictionless Flow	Success
4485 0403 7153 6584	101	Frictionless Flow	Fail
4573 8231 6871 0907	100	Challenge Flow	Success
4573 8231 6871 0907	101	Challenge Flow	Fail
4484 0700 0003 5519^	257	3DS1 fallback	Success
4484 0700 0003 5519^	258	3DS1 fallback	Fail
5352151570003404^^	100	No 3DS2	Success
5352151570003404^^	101	No 3DS2	Fail

- ^Transaction total should not be 5000c or this will not trigger.
- ^^Transaction total must be 5000c or this will not trigger

Translations

The TMTProtects Payment Modal currently supports the following translations:

Language	"lang" option
Chinese (Traditional)	zhZH
English (default)	enGB
French	frFR
German	deDE
Italian	itIT
Japanese	jaJA
Kazakh	kkKK
Korean	koKO
Latvian	lvLV
Portuguese	ptBR
Romanian	roRO
Russian	ruRU
Spanish	esES
Ukranian	ukUK
Uzbek (Tajik)	uzUZ

Should you wish to contribute a translation, please supply translations for the fields below to techsupport@trustmytravel.com.

Payment Modal

```
{
    "form": {
        "title": "Payment Details",
        "invoice": "Invoice",
        "cc_no": "Credit Card Number",
        "cvv": "CVV",
        "expiry": "Expiry Date",
        "pay": "Pay",
        "success": "Payment successful",
        "retry": "Retry",
        "terms": "This site uses Trust My Travel t/a TMTProtects to facilitate and protect your payment as merchant of record.
 By clicking Pay, you agree to Trust My Travel’s terms",
        "waitingForTransactionResult": "Waiting for transaction result...",
        "closeWarning": "Don't close this window until you have completed the transaction",
        "securityCheck": "Loading security check",
        "failed": "Payment failed",
        "close": "Close",
        "redirecting": "Redirecting to payment processor..."
    },
    "paymentStatus": {
        "submitting": "Submitting Payment...",
        "contactingBank": "Contacting Bank...",
        "apologies": "Apologies, this is taking longer than usual...",
        "stillWaiting": "Still waiting for a response..."
    },
    "errors": {
```

```
        "connecting": "Payment Service Provider unavailable",
        "timeout": "Your payment attempt has timed out, but may have been successful. Please contact the site admin and quote
booking ID {{id}}"
    },
    "paymentMethods": {
        "alipay": "Alipay",
        "credit-card": "Credit Card",
        "dlocal": "Installments",
        "giropay": "Giropay",
        "ideal": "iDEAL",
        "rapipago": "Rapipago",
        "sofort": "Sofort"
    },
    "dlocal": {
        "noInstallments": "No installments",
        "monthlyInstallments": "monthly installments",
        "single": "Single installment",
        "singleFees": "Single installment fees",
        "singleTotal": "Single installment total",
        "TotalCost": "Total cost of plan"
    },
    "validation": {
        "creditCard": "Credit card number entered is not valid",
        "cvv": "CVV number entered is not valid",
        "expiry": "Please enter a valid expiry date",
        "futureExpiry": "Please enter a valid future dated expiry date",
        "document": "Documento Nacional de Identidad (DNI) or Clave Única de Identificación Tributaria (CUIT) is required"
    }
}
```

Email Receipts

```
'subject'                   => 'Trust My ID: :trust_id',
'subject_refund'            => 'Trust My ID: :trust_id Refund',
'subject_chargeback'        => 'Trust My ID: :trust_id Chargeback',
'greeting'                  => 'Dear :Name',
'intro'                     => 'This email confirms your booking with :member is protected by TrustProtects.Me',
'refund_intro'              => 'A refund has been issued on your booking with :member.',
'chargeback_intro'          => 'We have received a chargeback on your booking with :member.',
'booked_header'             => 'Item Booked with:',
'item_header'               => 'Item(s) Ordered:',
'amount_header'             => 'Amount Paid:',
'amount_refund_header'      => 'Amount of Refund:',
'amount_chargeback_header'  => 'Amount of Chargeback:',
'important_header'          => 'Important:',
'p1'                        => 'Your payment is only protected if the details above are correct and it is important that you c
heck them for any inaccuracies. Any inaccuracies not declared within 7 days of receipt of this email will invalidate any prote
ction.',
'p2'                        => 'Please quote the above Trust My ID in any correspondence with us and email customer@trustprote
cts.me',
'p3'                        => 'For full details of the financial protection please visit https://www.trustprotects.me and vie
w our terms at https://www.trustprotects.me/terms',
'p4'                        => 'TrustProtects.Me is a division of Qubotic Limited which includes Trust My Travel and Trust My
Buy',
```

Troubleshooting

If you are having difficultly integrating the Payment Modal, please read through the troubleshooting guides below.

Form Implementation

Nothing Happens

You are confident you have completed the integration, you visit your test payment page, click to pay, and nothing happens!

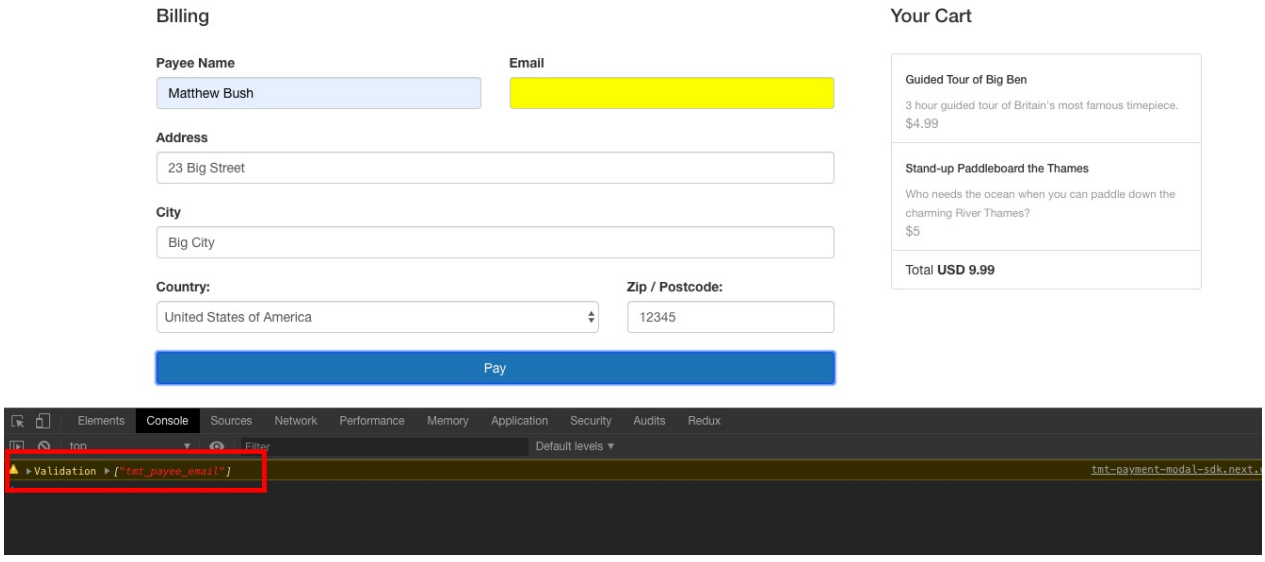
Please check the following:

- That you have an input with the class `tmt_payee_name` and a value
- That you have an input with the class `tmt_payee_email` and a value

Note that if either or both of these inputs are visible, then a style attribute would be attached to them in the event of no value being supplied. E.g.

```
<input name="payee_name" type="text" class="form-control tmt_payee_name" style="background: yellow;">
```

If you have set either or both of these inputs to hidden, then its not immediately obvious if no values are present. It is recommended that you enable Debug mode if this is the case. You should then see output to this effect.



Form Submits

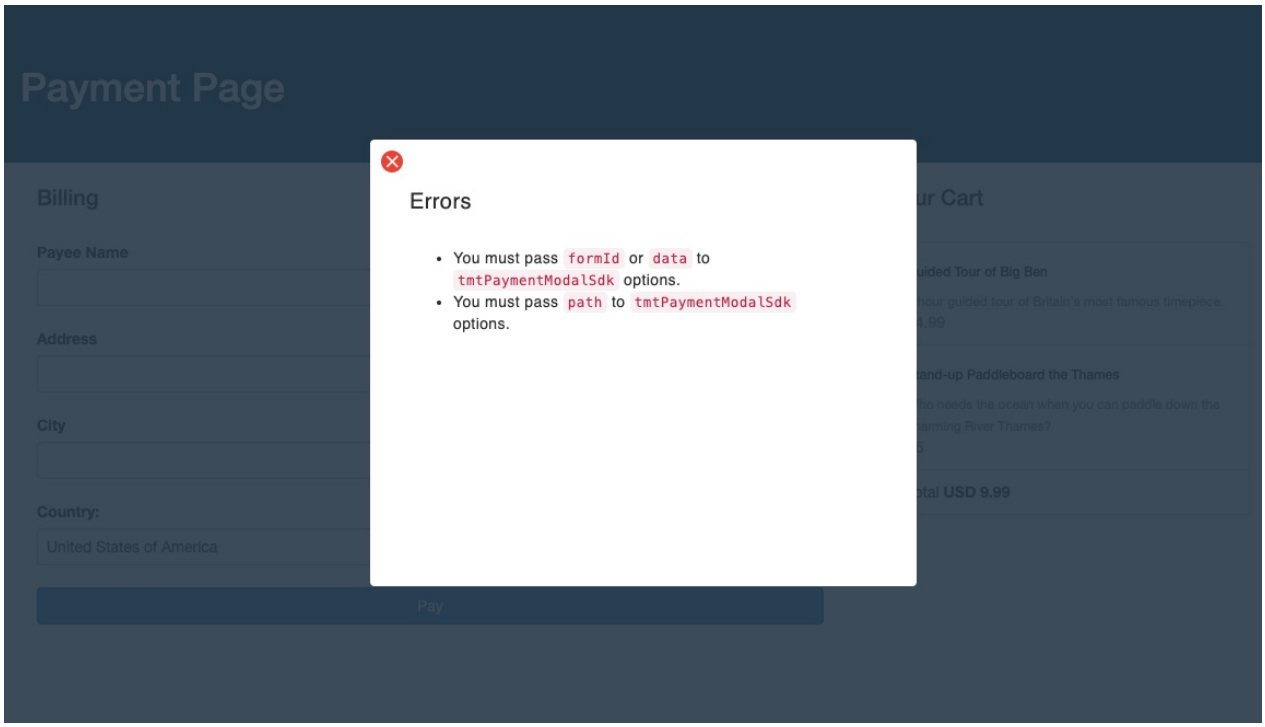
You are confident you have completed the integration, you visit your test payment page, click to pay, and the payment form submits without triggering the modal

Please check the following

- That you have correctly included the Payment Modal scripts
- That no other javascript included on your payment page is triggering errors in console

Init Errors

If you do not init the modal with the mandatory options for the implementation you require, then the modal will be triggered as per the screenshot below informing you which mandatory fields are missing.



This error would be resolved by passing a formId and path to the modal init call, for example:

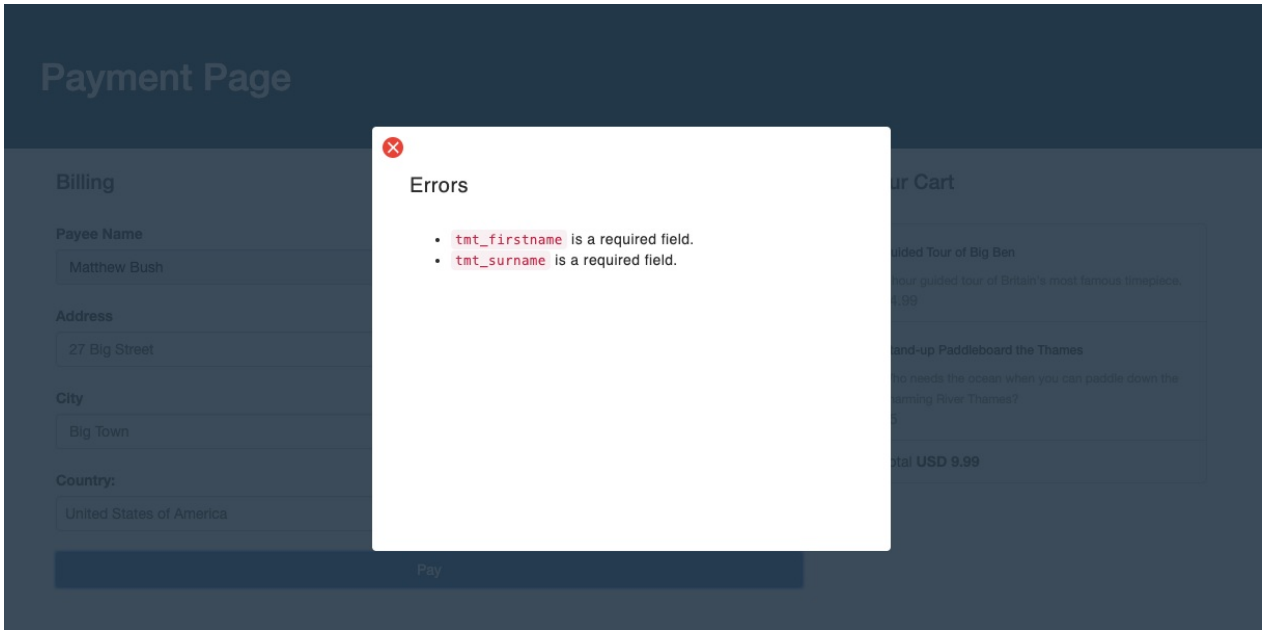
```
var tmtPaymentModal = new window.tmtPaymentModalSdk({
    path: "tmt-test",
    formId: "tmt-payment-form"
})
```

Required Field Errors

To successfully trigger the Payment Modal, required data must be present and correctly referenced depending on the implementation you are using:

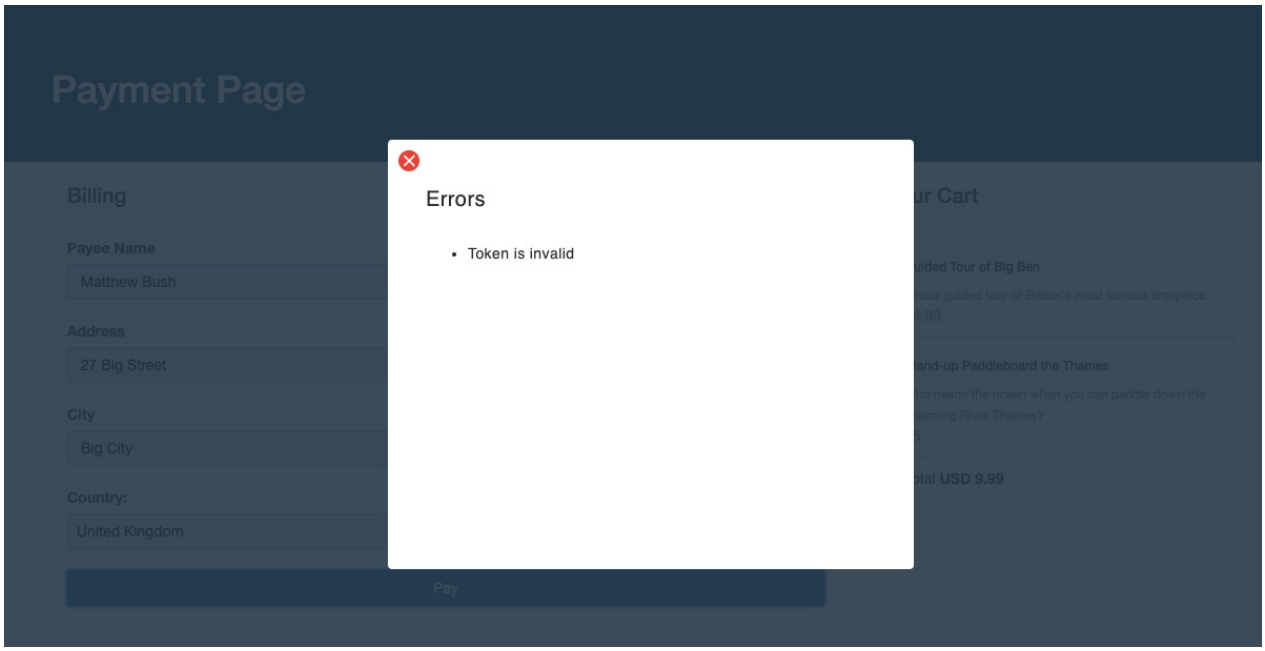
- Form implementation required fields
- Data Object implementation required fields

If you do not include all the required fields, the modal will trigger with a error output to indicate the missing fields similar to the below. If you have debug mode enabled, the missing fields will also be output to console.



Invalid Token

To identify yourself to the modal, you need to pass it a valid authstring. Failure to do this will result in output as per the screenshot below.

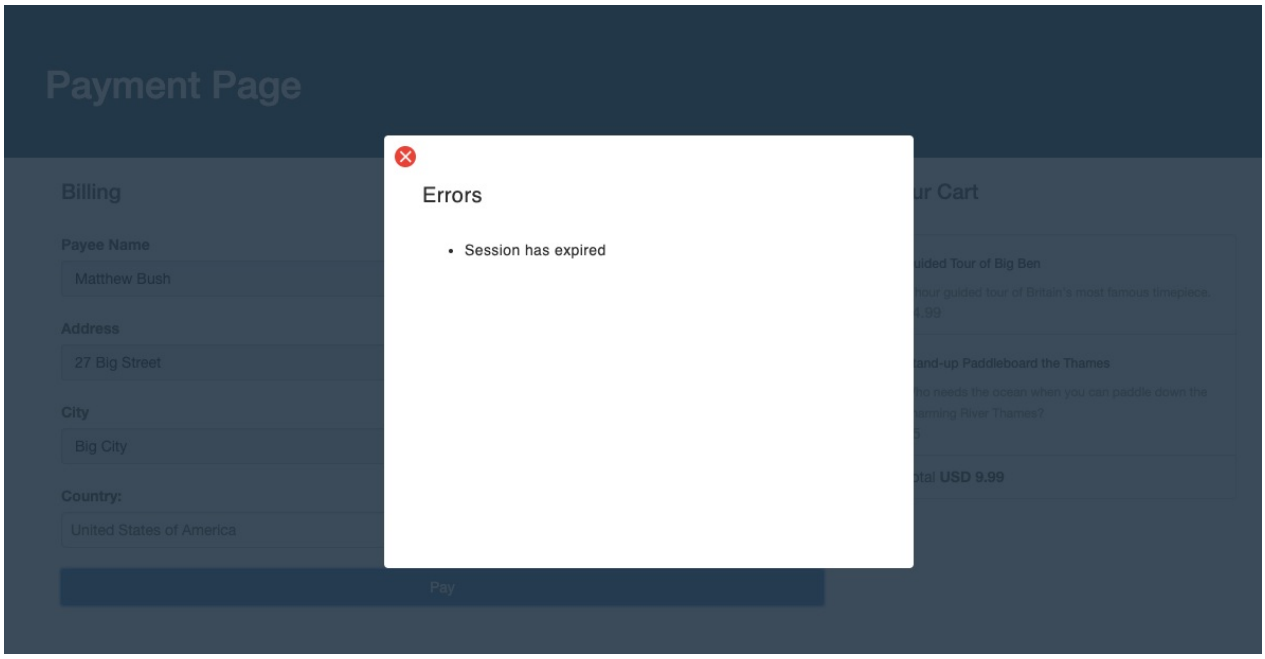


Should you receive this error, please check the following:

- Are you concatenating the fields in alphabetical order as shown in the examples?
- Are you using the same channel ID as that passed in the form or data object?
- Are you using the base currency for the channel with the ID passed in the form or data object?
- Are you salting the authstring with the channel secret for the channel with the ID passed in the form or data object?
- Have you used the same timestamp in the authstring as the timestamp which is appended to it?
- If you are using additional fields in the authstring, have you declared them in the Verify Option?

Expired Token

To prevent reuse of tokens, they are only valid for 15 minutes. In order to prevent reuse of expired tokens, a timestamp is added to the authstring and then appended to it so that a duplicate authstring can be built API side for comparison. If you fail to append the timestamp, or if it is older than 15 minutes, you will receive output as per the screenshot below:

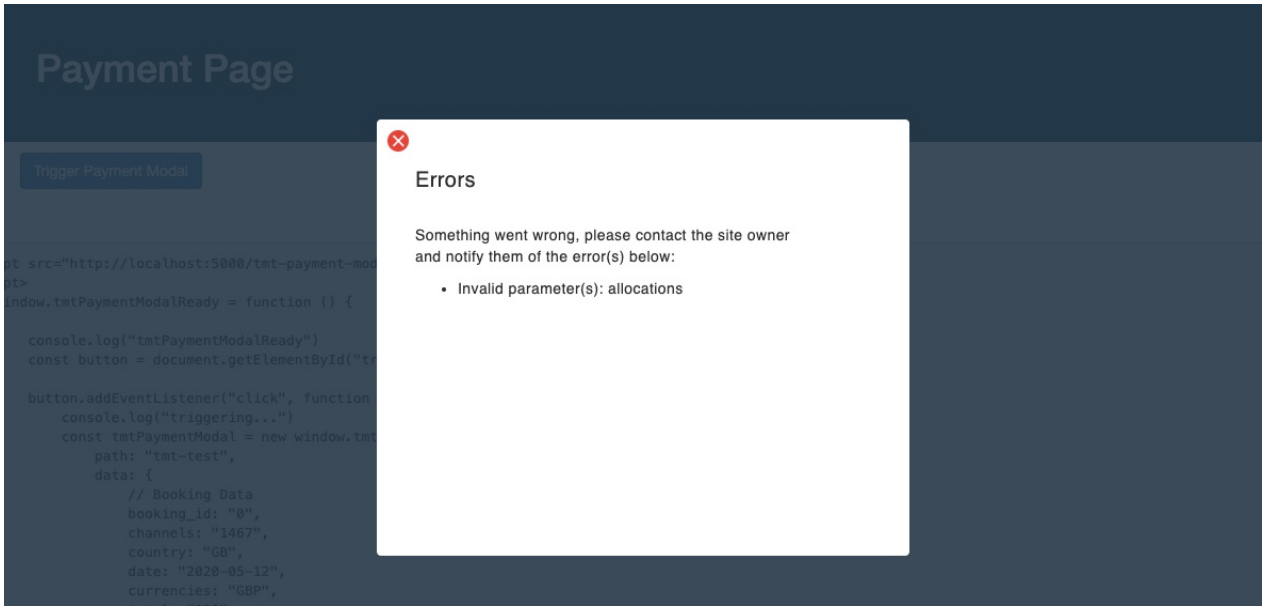


Should you receive this error, please check the followung:

- Are you using and appending the same timestamp?
- Are you generating a timestamp in GMT?
- Are you ouputting your timestamp in the format YYYYDDMMHIS?

Allocation Errors

If you are using the Data Object Implementation and including Allocations, you may receive output as per the screenshot below after having successfully triggered the modal and entered credit card details:



Should you receive this error, please ensure that the channel that is incurring charges has sufficient funds to meet those charges.

For example, consider a channel with ID = 23, which has a per transaction fee of USD 0.50 and has a credit card percentage of 3.5% applied. The two examples below would result in too little being available to meet those charges:

Example One: Channel 23 receives allocation and incurs charges

- USD 2 is allocated to Channel 23
- Channel 23 is nominated as `charge_channel`
- Charges levied against Channel 23 would be USD 4 (3.5% of USD 100 = USD 3.50 + USD 0.50 per transaction fee)
- USD 2 is not sufficient to cover charges of USD 4, error is returned.

```
{
    booking_id: '0',
    channels: 2,
    currencies: 'USD',
    total: '10000',
    ...
    allocations: [{
        channels: 23,
        currencies: 'USD',
        total: 200,
        operator: 'flat'
    }],
    charge_channel: 23
}
```

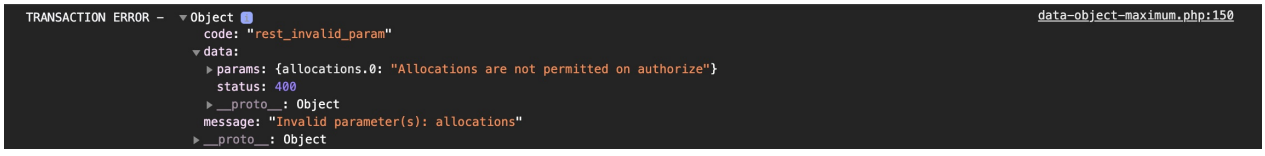
Example Two: Channel 23 is master channel and incurs charges

- USD 98 is allocated to Channel 2
- No `charge_channel` defined, so defaults to main channel, which is 23.
- Charges levied against Channel 23 would be USD 4 (3.5% of USD 100 = USD 3.50 + USD 0.50 per transaction fee)
- USD 2 remaining after allocating USD 98 to channel 2 is not sufficient to cover charges of USD 4, error is returned.

```
{
    booking_id: '0',
    channels: 23,
    currencies: 'USD',
    total: '10000',
    ...
    allocations: [{
        channels: 2,
        currencies: 'USD',
        total: 9800,
        operator: 'flat'
    }]
}
```

Payment Failure

If payments are failing unexpectedly, for example when testing with credit cards that should be passing, please listen on the transaction_error callback as this should give you feedback on where you are going wrong. The example below shows a transaction attempt that has failed as allocations were included on an authorize transactions.



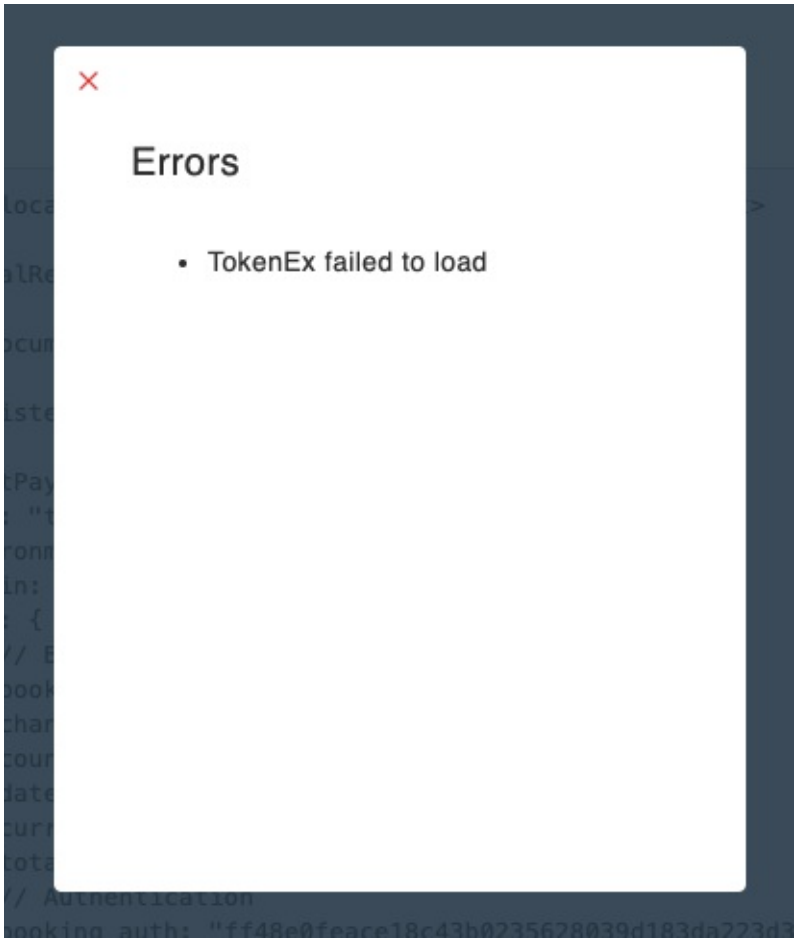
Invalid Data Token

If you successfully run a transaction from end to end, but the transaction fails with `"content": "Invalid data/token."`, then you have triggered the modal in an environment that does not match that of the channel you are running the transaction in. Cards are tokenised in the environment specified in modal instantiation. API requests made via the tokeniser to the payment gateway are made in the

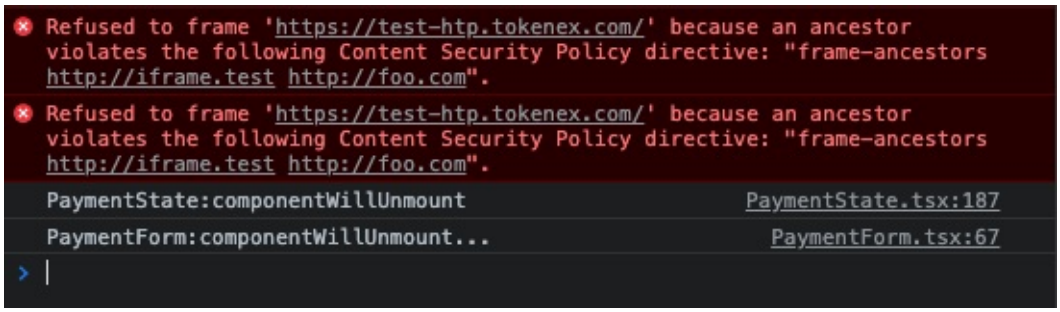
environment of the channel (live or test). If a channel has an account_mode of "live" and the modal is instantiated with environment "test", the card will be tokenised in the test token environment where-as the request will be routed via the live environment. No token will exist in the live environment so the response of "Invalid data/token" will be returned, and the transaction will fail.

TokenEx Failed to Load

TokenEx are our third party provider for tokenising cards. When the modal is triggered, we initialise their scripts. If we can't initialise their scripts, we can't process payments and this error is displayed.



If you are serving the modal up within an iframe, this error has likely been caused by an incorrect, or no value for origin. This can be verified by checking console logs for errors like the below:



Appendix

ISO 3166-1 alpha-2 Country Codes

CODE	Country
AD	Andorra
AE	United Arab Emirates
AF	Afghanistan
AG	Antigua and Barbuda
AI	Anguilla
AL	Albania
AM	Armenia
AO	Angola
AQ	Antarctica
AR	Argentina
AS	American Samoa
AT	Austria
AU	Australia
AW	Aruba
AX	Åland Islands
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BD	Bangladesh
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BH	Bahrain
BI	Burundi
BJ	Benin
BL	Saint Barthélemy
BM	Bermuda
BN	Brunei Darussalam
BO	Bolivia (Plurinational State of)
BQ	Bonaire, Sint Eustatius and Saba
BR	Brazil
BS	Bahamas

BT	Bhutan
BV	Bouvet Island
BW	Botswana
BY	Belarus
BZ	Belize
CA	Canada
CC	Cocos (Keeling) Islands
CD	Congo, Democratic Republic of the
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CK	Cook Islands
CL	Chile
CM	Cameroon
CN	China
CO	Colombia
CR	Costa Rica
CU	Cuba
CV	Cabo Verde
CW	Curaçao
CX	Christmas Island
CY	Cyprus
CZ	Czechia
DE	Germany
DJ	Djibouti
DK	Denmark
DM	Dominica
DO	Dominican Republic
DZ	Algeria
EC	Ecuador
EE	Estonia
EG	Egypt
EH	Western Sahara
ER	Eritrea
ES	Spain
ET	Ethiopia

FI	Finland
FJ	Fiji
FK	Falkland Islands (Malvinas)
FM	Micronesia (Federated States of)
FO	Faroe Islands
FR	France
GA	Gabon
GB	United Kingdom of Great Britain and Northern Ireland
GD	Grenada
GE	Georgia
GF	French Guiana
GG	Guernsey
GH	Ghana
GI	Gibraltar
GL	Greenland
GM	Gambia
GN	Guinea
GP	Guadeloupe
GQ	Equatorial Guinea
GR	Greece
GS	South Georgia and the South Sandwich Islands
GT	Guatemala
GU	Guam
GW	Guinea-Bissau
GY	Guyana
HK	Hong Kong
HM	Heard Island and McDonald Islands
HN	Honduras
HR	Croatia
HT	Haiti
HU	Hungary
ID	Indonesia
IE	Ireland
IL	Israel
IM	Isle of Man
IN	India
IO	British Indian Ocean Territory
IQ	Iraq

IR	Iran (Islamic Republic of)
IS	Iceland
IT	Italy
JE	Jersey
JM	Jamaica
JO	Jordan
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KH	Cambodia
KI	Kiribati
KM	Comoros
KN	Saint Kitts and Nevis
KP	Korea (Democratic People's Republic of)
KR	Korea, Republic of
KW	Kuwait
KY	Cayman Islands
KZ	Kazakhstan
LA	Lao People's Democratic Republic
LB	Lebanon
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
LY	Libya
MA	Morocco
MC	Monaco
MD	Moldova, Republic of
ME	Montenegro
MF	Saint Martin (French part)
MG	Madagascar
MH	Marshall Islands
MK	North Macedonia

ML	Mali
MM	Myanmar
MN	Mongolia
MO	Macao
MP	Northern Mariana Islands
MQ	Martinique
MR	Mauritania
MS	Montserrat
MT	Malta
MU	Mauritius
MV	Maldives
MW	Malawi
MX	Mexico
MY	Malaysia
MZ	Mozambique
NA	Namibia
NC	New Caledonia
NE	Niger
NF	Norfolk Island
NG	Nigeria
NI	Nicaragua
NL	Netherlands
NO	Norway
NP	Nepal
NR	Nauru
NU	Niue
NZ	New Zealand
OM	Oman
PA	Panama
PE	Peru
PF	French Polynesia
PG	Papua New Guinea
PH	Philippines
PK	Pakistan
PL	Poland
PM	Saint Pierre and Miquelon
PN	Pitcairn
PR	Puerto Rico

PS	Palestine, State of
PT	Portugal
PW	Palau
PY	Paraguay
QA	Qatar
RE	Réunion
RO	Romania
RS	Serbia
RU	Russian Federation
RW	Rwanda
SA	Saudi Arabia
SB	Solomon Islands
SC	Seychelles
SD	Sudan
SE	Sweden
SG	Singapore
SH	Saint Helena, Ascension and Tristan da Cunha
SI	Slovenia
SJ	Svalbard and Jan Mayen
SK	Slovakia
SL	Sierra Leone
SM	San Marino
SN	Senegal
SO	Somalia
SR	Suriname
SS	South Sudan
ST	Sao Tome and Principe
SV	El Salvador
SX	Sint Maarten (Dutch part)
SY	Syrian Arab Republic
SZ	Eswatini
TC	Turks and Caicos Islands
TD	Chad
TF	French Southern Territories
TG	Togo
TH	Thailand
TJ	Tajikistan

TK	Tokelau
TL	Timor-Leste
TM	Turkmenistan
TN	Tunisia
TO	Tonga
TR	Turkey
TT	Trinidad and Tobago
TV	Tuvalu
TW	Taiwan, Province of China
TZ	Tanzania, United Republic of
UA	Ukraine
UG	Uganda
UM	United States Minor Outlying Islands
US	United States of America
UY	Uruguay
UZ	Uzbekistan
VA	Holy See
VC	Saint Vincent and the Grenadines
VE	Venezuela (Bolivarian Republic of)
VG	Virgin Islands (British)
VI	Virgin Islands (U.S.)
VN	Viet Nam
VU	Vanuatu
WF	Wallis and Futuna
WS	Samoa
YE	Yemen
YT	Mayotte
ZA	South Africa
ZM	Zambia
ZW	Zimbabwe