Table of Contents

Overview

Before You Begin

To begin using the modal, you will need:

- A TMTProtects account
- A channel on your TMTProtects account that is ready for processing. You can do this via the TMTProtects Dashboard or via the TMTProtects API
- A secret key for that channel. You can obtain this via the TMTProtects Dashboard or by a direct API call using the TMTProtects API

Scripts

To get set-up with the TMT Payment Modal 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.1.24.0.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` depending on the implementation that you are using.

Example

If the API URL provided to you by Trust My Travel was `https://tmtprotects.com/test-site` then your path variable will be `test-site` as shown below.

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

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 timestamp must be included in this string and appended to it. Authorisation strings are valid for 15 minutes.

The process for generating the authorisation string is as follows:

```
// 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 = getenv('CHANNEL_SECRET');
$salted_auth_string = hash( 'sha256', $auth_string . $secret );

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

To test the authorisation string your app generates matches the expected authorisation string, you can use the Auth Test Demo

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 of Transaction

The modal workflow is considered complete either when a user has successfully completed a transaction, or if a user has had three consecutive unsuccessful attempts to complete a transaction.

Successful Transaction

Once a transaction is successfully completed, the transaction_logged callback is triggered. This will include the full API response to the `POST /transactions` call performed by the modal. An example of a `POST /transactions` response can be found in the TMTProtects API Docs

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

The user experience is now back in your hands, and it is up to you to close the modal and redirect the user to your payment success page.

Unsuccessful Transaction

An unsuccessful transaction occurs when the user has supplied incorrect credit card data, or if the card issuing bank declines the card for some reason. The Payment Modal allows for three failed attempts and will call the transaction_failed callback on each failure.

Rejected Transaction

On the third and final unsuccessful transaction, the user is prevented from making any further attempts to pay, and the transaction_rejected callback is triggered. At this point you can safely close the modal.

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.
    })
})
```

Release Notes

- Fixed bug where alerts to users about closing or refreshing the page blocked page redirects on successful transaction
- Added option for disabling page redirect warnings

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 techsupport@trustmytravel.com supplying as much detail as possible such as including links to pages where the modal is being implemented or code examples.

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

Optional

The following options can be used where required

- paymentCurrency
- lang
- disableLang
- disableCloseWindowPrompt

Examples

path

The path of your TMTProtects site. If the API URL provided to you by Trust My Travel was `https://tmtprotects.com/test-site` then your path will be `test-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',9
        }

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

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

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

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

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

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
tmt_channels	Set this to the ID of the channel you created
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_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_date	The end date of travel in YYYY-MM-DD format
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 shown as a cent value (e.g. $10.00 = 1000)

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="UK" />
        <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

Key	Description
booking_auth	The hashed and salted authorisation string for the transaction
booking_id	Set this to 0 to create a new booking
channels	Set this to the ID of the channel you created
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
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
date	The end date of travel in YYYY-MM-DD format
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 shown as a cent value (e.g. $10.00 = 1000)

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',
        total: 1000,
        operator: 'flat'
    }],
    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',
```

```
        total: 5,
        operator: 'percent'
    }],
    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',
        total: 1000,
        operator: 'flat'
    }]
}
```

Notes

- 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.

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 eastablish 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: ""
    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. This callback can be called up to three times before a transaction is rejected.

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: ""
    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

On the third and final unsuccessful transaction, the user is prevented from making any further attempts to pay, and the transaction_rejected callback is triggered with an object containing an error message returned. At this point you can safely close the modal.

```
{
    message: "Three failed attempts"
}
```

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!'
}
```

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

Credit Card Number	CVV	Expiry	Outcome
4111 1111 1111 1111	123	Any valid Year / Month combination	Success
5105 1051 0510 5100	123	Any valid Year / Month combination	Fail

Introduction

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 Code Examples

Authorisation String

```
// 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 = getenv('CHANNEL_SECRET');
$salted_auth_string = hash( 'sha256', $auth_string . $secret );

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

Node.JS Code Examples

Authorisation String

```
// 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 } = process.env

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

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

Translations

The TMTProtects Payment Modal currently supports the following translations:

Language	"lang" option
Chinese	zhZH
English (default)	enGB
German	deDE
Italian	itIT
Japanese	jaJA
Kazakh	kkKK
Korean	koKO
Portuguese	ptBR
Romanian	roRO
Russian	ruRU
Spanish	esES
Ukranian	ukUK
Uzbek	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 sites 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"
    },
    "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}}"
    }
}
```

Payment Receipt

```
'subject' => 'Payment Reference: :trust_id',
'greeting' => 'Dear :Name',
'p1' => 'This is a receipt for payment and is not your booking confirmation or voucher',
'p2' => 'Thank you for your payment for travel services to :member for :currency :amount. Trust My Travel Limited have process
ed your payment on the behalf of :member, meaning your travel booking is financially protected by TMTProtects.Me',
'p3' => 'TMTProtects.Me by Trust My Travel protects Travellers in the event of non-delivery of service and insolvency by their
 Travel Provider.',
'item_header' => 'Item(s) Ordered:',
'enquiries_header' => 'ENQUIRIES',
'p4' => 'Booking queries regarding your order can be addressed to your Travel Provider directly by visiting their website and
contacting them via their "contact us" page. Your order details listed above may be required in order to identify your booking
 and answer your questions.',
'p5' => 'Payment or Financial Protection queries can be addressed to trust@trustmytravel.com where you can receive support and
 information relating to the processing and protection of your credit card payment. Please include your full name and the abov
e Order Reference in the subject of your email.'
```

Refund Receipt

```
'subject' => 'Refund Receipt: :trust_id',
'heading' => 'Payment Reference: :trust_id',
'greeting' => 'Dear :Name',
'p1' => 'A refund of :currency :amount has been issued on your recent purchase with :member. ',
'p2' => 'Your transaction will appear on your card statement as \'trustmytravel.com+441780408121\'. Please note it can take 3/
4 working days to show up on your statement.',
'item_header' => 'Item(s) Ordered:',
'p4' => 'Queries regarding your order can be addressed to :member directly by visiting their website and contacting them via t
heir "contact us" page. Your order details listed above may be required in order to identify your booking and deal with your q
uestions.',
'p5' => 'In addition you can receive customer support relating to the processing of your credit card payment by emailing us at
 customerservices@trustmytravel.com with the above payment reference in the subject of your email.'
```

Chargeback Receipt

```
'subject' => 'Chargeback Notification:  :trust_id',
'heading' => 'Trust ID:  :trust_id',
'greeting' => 'Dear :Name',
'p1' => 'We have received a chargeback of :currency :amount on the above booking. ',
'p2' => 'We believe this to be a legitimate transaction. Can you please reply to us as soon as possible and let us know if you
 did indeed make this booking and if so can you please reverse this with your issuing bank?',
'sign_off' => 'Yours sincerely,',
'signed' => 'The Team',
'item_header' => 'Item(s) Ordered:',
```