IntelliPay™

Lightbox EMV Terminal

Quick Start Guide

8/2/2022

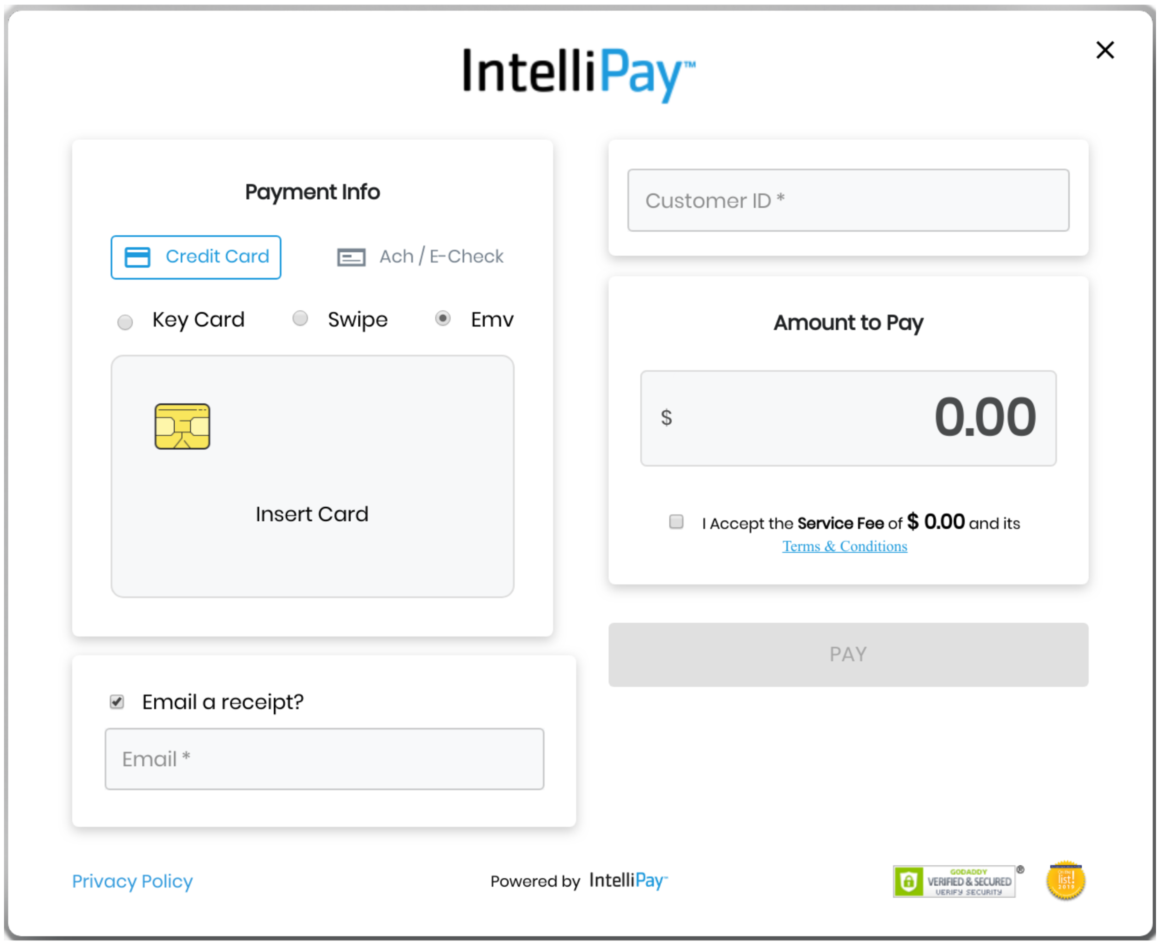
IntelliPay
12884 Frontrunner Blvd, Suite 220
Draper, Utah 84020

Table of Contents

Overview

IntelliPay's Lightbox EMV Terminal displays a payment form that floats over your existing website content. The form makes it easy for your customer to submit a payment without having to leave your billing portal. Lightbox EMV Terminal pulls customer ID numbers and payment amounts from the elements on your website. It then calculates fees and processes transactions using other information you identify on your page.

The image below shows the Lightbox EMV Terminal payment form with prefilled Customer Id and Amount fields:



IntelliPay

Payment Info

Credit Card | Ach / E-Check

Key Card ○ Swipe ○ Emv ●

Insert Card

Customer ID *

Amount to Pay

$ 0.00

☐ I Accept the **Service Fee** of **$ 0.00** and its Terms & Conditions

PAY

☑ Email a receipt?

Email *

Privacy Policy

Powered by IntelliPay

Prefilling the form elements makes the payment process easier for your customer. You can set values for the following form elements:

- Customer Id (“account”) – The application automatically displays the correct customer ID based on the information that you flag on your web page. The customer ID creates a way to track invoices and payments.
- Amount – The amount of the transaction is automatically filled in based on information that you flag your web page. Prefilling the transaction amount makes it easy for your customer to review an invoice and immediately submit a payment for the correct amount

In addition the following fields will help you locate customers and transactions in our interface:

- lastname and firstname – This will help you locate the transaction or customer easier in our interface.
- invoice – This should be filled in with a unique identifier you create for the transaction. This will help you locate the specific transaction easier in our interface.

The service fee amount is calculated based on the amount value and updates if your customer changes the amount.

This document includes information about how to incorporate Lightbox EMV Terminal into your online payment application:

- Implementation – The Implementation section describes how to set up your website to display and customize Lightbox EMV Terminal.
- Customization – The Customization section defines the field values and labels that you can modify on the payment form.
- Response Handling – The Response Handling section defines the information returned by IntelliPay and describes how to display it on your web page.
- Sample Code – The Sample Code section includes code snippets for most common frameworks and languages. Update the authentication variables and add the code to your application to get started using Lightbox EMV Terminal.

Security

IntelliPay will provide you with an API key and a merchant key, which you must use to communicate with Lightbox EMV Terminal. Make sure that you store your API credentials outside of your source code to prevent them from being exposed. Depending on your development environment or language, you may store your credentials in environment variables, secure containers or some other secure storage.

Implementation

Lightbox EMV Terminal displays in front of an existing website. Typically, you will include Lightbox EMV Terminal on an invoice or checkout page. The page will include your customer's ID or account number, the balance due, and it may include information like billing address and phone number.

Lightbox EMV Terminal can automatically pull customer and balance information from your billing page and use it to populate the payment form. To implement Lightbox EMV Terminal on your website, first create a billing site that displays customer and billing details. Next, identify the elements on your site that Lightbox EMV Terminal will use to display and process transaction information by setting the class attributes to "ipayfield". Add code to your website to request the Lightbox EMV Terminal scripts, and create a button that opens the payment form. Finally, handle the transaction response messages from IntelliPay after your customer submits the transaction request.

The steps below provide details about incorporating Lightbox EMV Terminal into your website:

1. On your billing page, identify the elements that will be sent to IntelliPay with the transaction request. Any HTML element that includes class="ipayfield" will be available for the Lightbox EMV Terminal script. For example, <input> elements can be used to display billing details and also make them available to Lightbox EMV Terminal.

   The following HTML tells Lightbox EMV Terminal that the transaction amount should be $1.00:

   ```
   <input class="ipayfield" data-ipayname="amount" type=text
   value="1.00" name='myamt' id='myamt'>
   ```

See the Customization section on page for a complete list of the values you can provide to Lightbox EMV Terminal.

2. Add code that submits a POST request to IntelliPay in the <HEAD> element of the web page that will display the payment form. You must submit this POST request every time your customer reaches the payment page.

   The example below shows PHP that submits the POST request to the autoterminal endpoint:

```
<HEAD>
<?php
$merchantkey = 'YOUR_MERCHANTKEY';
$apikey = 'YOUR_APIKEY';

$result =
file_get_contents('https://secure.cpteller.com/api/custapi.cfc
?method=autoterminal',false,
stream_context_create(array('http' => array('header'  =>
"Content-type: application/x-www-form-urlencoded\r\n",'method'
=> 'POST','content' =>
      http_build_query(array('merchantkey' => $merchantkey,
'apikey' => $apikey, 'operatingenv' =>
'businessattended'))))));

if ($result === FALSE) { /* Handle error */ }
echo $result;
?>
</HEAD>
```

   Update the values for YOUR_MERCHANTKEY and YOUR_APIKEY in the sample code with the values supplied by IntelliPay. The Sample Code section on page includes code for different frameworks and languages.

   IntelliPay returns a style sheet, scripts, and tokens that Lightbox EMV Terminal uses to parse the information on your website, display the payment form, and submit and authenticate your transaction request.

3. Add the response from the POST request in step 2 to the <HEAD> element of your payment page.

4. Add a button to your site that your customer will click to open the payment form.You can launch the payment form using HTML attributes or Javascript, as shown in the samples below:

   If you need your customer to enter payment card details, use the submit methods, which open the payment form with blank fields for entering card data. The example below shows launching the form from HTML:

   ```
   <button data-ipayname=”submit” class=”ipayfield”
   type=”button”>Open Lightbox</button>
   ```

   Call the intellipay.onSubmit() method to open the payment form using Javascript.

5. If required, your customer enters payment information on the form, and clicks the pay button. IntelliPay returns the status of the transaction. See the Response Handling section on page for details about transaction response messages.

Customization

Lightbox EMV Terminal includes functions for setting values on the payment form and customizing some field labels and colors. Depending on the field, you can define customizations using HTML or Javascript. The sections below describe setting field values and customizing the appearance of the form.

Field Values

You can set values for the following fields so that they are already populated when your customer opens the payment form:

- Customer ID (The "account" field)
- Amount

Other fields, which are defined in the Field Definitions section on page , are sent to IntelliPay in the background to support transaction processing. Your customer will not see them on the payment form, but you may display them on your billing page.

To define values that will be sent to IntelliPay using HTML, set the class attribute of any element to "ipayfield". If you use an element of the type HTMLInputElement element, Lightbox EMV Terminal will use the value that you assign to its "value" attribute. If you use any other HTML element, Lightbox EMV Terminal will parse the content of the tag to identify the correct value.

The example below shows how to define an amount value for the transaction using an <input> tag:

```
<input class="ipayfield" data-ipayname="amount" type=text
value="1.00" name='myamt' id='myamt'>
```

The next example shows how to set the same value using a <p> tag:

```
<p class="ipayfield" data-ipayname="amount">1.00</p>
```

Choose which element to use depending on the design and requirements of your billing site.

Form Appearance

You can define label text, item color, and background color for certain form elements using Javascript.

To set the label text, call the intellipay.setItemLabel() method, and include the field name and your customized label text as parameters. You can set label text for the following fields or messages:

- account – The default label for the account field is "Customer Id".
- header – The default text for the header is "Make A Payment".
- button – The default text to display on the submit payment button is "Pay".
- declinemessage - Set the text to display to the user on a decline.
- successmessage - Set the text to display to the user on an approval.

The sample code below shows how to change the default label on the account, header, and `intellipay.setItemLabel("account","Account ID");//Set the label for account`button fields:

```
intellipay.setItemLabel("header","My Payment Form");//Set the form title
intellipay.setItemLabel("button","Make Payment");//Set the pay button text
```

To customize item color, call the intellipay.setItemColor() method. Use the intellipay.setItemBackgroundColor() to define a custom background color for a field. Both methods take a field name and a color value as parameters. You can only set item and background colors for the header field.

The sample code below shows how to set an item color and a background color for the header:

```
intellipay.setItemColor("header","#FF0000");//red text in the
header
intellipay.setItemBackgroundColor("header","#0000FF");//blue
background in the header
```

Payment Methods

You may also wish the customer to preselect whether they are paying with ACH or credit card. We provide the methods setACHAvailable and setCCAvailable which take a boolean that is false if you want to disable that payment method, or true to reenable. They default to enabled if the merchant supports that payment method.

```
<input type="radio" value="CC" name='selectpay' onclick='
intellipay.setACHAvailable(false);intellipay.setCCAvailable(true)
;'>
<input type="radio" value="ACH" name='selectpay' onclick='
intellipay.setACHAvailable(true);intellipay.setCCAvailable(false)
;'>
```

Some fields or elements may be disabled or enabled using the intellipay.enable() or intellipay.disable() functions. Currently only email and account may be disabled. For example:

```
intellipay.disable("email");//Disable the email field
```

Store Only Mode

You may configure the Lightbox EMV Terminal to only store the customers billing information to do a future transaction via stored credentials. We provide the method setStoreOnly which takes a boolean that is true if you want to turn on the store-only mode, or false to return the terminal to payment acceptance mode. A successful stored payment returns the numeric CUSTID parameter as ‘custid’ in the response object passed to the registered runOnApproval() method..

```
<input type="radio" value="StoreOnly" name='selectpay' onclick='
intellipay.setStoreOnly(true);'>
<input type="radio" value="PaymentMode" name='selectpay'
onclick='intellipay.setStoreOnly(false);'>
```

Field Definitions

The table below defines all of the fields that you can use to send data to IntelliPay. Use the Field Name as the value for the data-ipayname attribute:

Field Name	**Format or Example**	**Description**
account	Ex: ABCD1234	Your identifier for this customer. This value will be displayed on the payment form.
amount	N.NN Ex: 123.45 or 0.34. There must be at least one digit to the left of the decimal point.	The transaction amount before the fee. This value will be displayed on the payment form.
firstname	Ex: John	The customer’s first name.
lastname	Ex: Doe	The customer’s last name.
address1	Ex: 1234 Anywhere Street	The first line of the customer’s street address.

address2	Ex: PO Box 1234	The second line of the customer’s street address, if applicable.
city	Ex: Jersey City	The city portion of the customer’s address.
state	Two-character state abbreviation only or empty. Ex: NJ	The state portion of the customer’s address.
zipcode	NNNNN Ex: 07304	The zip code portion of the customer’s address.
country	Three-character ISO identification. Ex: USA	The three-character ISO formatted country code for the country portion of the customer’s address.
phone	NNN NNN NNNN	The customer’s phone number.
email	user@domain.com	The customer’s email address.
fee	N.NN Ex: 123.45 or 0.34. There must be at least one digit to the left of the decimal point.	This field will be updated with the fee amount if that information is available. The fee value cannot be set by the merchant.
invoice	Ex. A1B2C3D4	An invoice number associated with the transaction.
cardnum	Ex. 4111111111111111	The customer’s credit card number.
cardname	Ex. Jane Doe	The cardholder name.
expdate	MMYY ex: 1225	The card expiration date. If no value is set for expdate, Lightbox EMV Terminal sets the value by combining expyear and expmonth.
expyear	YYYY or YY ex: 2025	The year portion of the card’s expiration date.card expiration date. If you set a value for expdate, Lightbox EMV Terminal will not use expyear or expmonth.
expmonth	MM ex: 12	The month portion of the card’s expiration date. If you set a value for expdate, Lightbox EMV Terminal will not use expyear or expmonth.
cvv	NNN or NNNN	The card verification value.
comment	Ex. “Purchase of Large ACME weight by one Wile E. Coyote”	A note about the transaction that will be stored with the payment.
custid	Ex. 111A1B1	The customer ID for a stored wallet transaction. Do not send values for the cardnum and

		cardholder fields if you are performing a stored wallet transaction.
batchid	Ex. 99999998	The batch ID associated with the approval.
routingnum	123456	The routing number on the check
bankacctnum	12345678	The bank account number on the check
bankacctype	C	C for checking, not required

Response Handling

After you submit a transaction, IntelliPay will respond with a result message. The response will contain the following information:

Attribute Name	Format or Example	Description
response	Single character: A for approved, D for declined	Defines the transaction approval status.
status	Numeric: Ex. 12345678, or -6	A positive Payment ID if the transaction is approved, or a negative number representing an error.
authcode	Text: Ex. 384A84	An industry standard authorization code for the transaction.
declinereason	Text: Ex. "Success"	"Success" for an approved transaction, or the reason the transaction was not approved.
amount	Amount: Ex. 1.23	The transaction amount submitted with the approval request.
fee	Amount: Ex. 1.23	The fee amount charged.
call	Text: Ex. card_payment	The API call used to authorize the transaction.
nonce	Text in GUID format	A unique string associated with the response.
hmac	Base64 HMAC	The HMAC for the response. Use your authkey to validate the HMAC value.
receiptelements	JSON	A JSON object with elements useful for rendering a receipt.
custid	Numeric	Returned on a successful Store Only call

To display the response, create elements on your web page with the class set to "ipayresponsefield". The elements that you use to display the response values must be HTML form elements. Add a data-ipayname attribute to the element and assign it a value of one of the Attribute Names in the table above. The example below will display the decline reason in a field named "Decline Reason":

```
<input class=”ipayresponsefield” data-ipayname=”declinereason”
name=”Decline Reason” type=”text”>
```

You can also define a response handler function by calling the intellipay.runOnApproval() and intellipay.runOnNonApproval() functions. The example below shows a function that will run when IntelliPay returns an approval:

```
intellipay.runOnApproval(function(response){
console.log("Demo Got Approval! --> " +
JSON.stringify(response));
});
```

Sample Code

This section includes information about working with Lightbox EMV Terminal in different frameworks and sample code that you can add to your application to quickly begin using Lightbox EMV Terminal. This document includes sample code for the following languages and frameworks:

- PHP
- PERL
- CFML
- RUBY
- C# .NET
- VB.NET
- Node.js (Express.js)
- REACT

The code included in these sections sets your merchantkey and apikey values and sends a POST request to the Lightbox EMV Terminal autoterminal endpoint.

See the Implementation section on page for additional details about how to use the code and display the payment form. See the Customization section on page for information about how to specify field names and values.

PHP

Copy and paste the code below into the <HEAD> element of the web page that will display the payment form:

```
<?php
$merchantkey = 'YOUR_MERCHANTKEY';
$apikey = 'YOUR_APIKEY';

$result =
file_get_contents('https://secure.cpteller.com/api/custapi.cfc?me
thod=autoterminal',false,
      stream_context_create(array('http' => array('header'  =>
"Content-type: application/x-www-form-urlencoded\r\n",'method'
=> 'POST','content' =>
      http_build_query(array('merchantkey' => $merchantkey,
'apikey' => $apikey, 'operatingenv' => 'businessattended'))))));
if ($result === FALSE) { /* Handle error */ }
echo $result;
?>
```

Change the values for YOUR_MERCHANTKEY and YOUR_APIKEY to the values provided by IntelliPay.

See the Implementation section on page for details about displaying the payment form.

PERL

Copy and paste the code below into the <HEAD> element of the web page that will display the payment form:

```
use HTTP::Request::Common qw(POST);
use LWP::UserAgent;

my $res = LWP::UserAgent->new()->request(POST
'https://secure.cpteller.com/api/custapi.cfc?method=autoterminal'
, {'merchantkey' => "YOUR_MERCHANTKEY", 'apikey' =>
"YOUR_APIKEY", 'operatingenv' => "businessattended"});
print $res->content;
```

Change the values for YOUR_MERCHANTKEY and YOUR_APIKEY to the values provided by IntelliPay.

See the Implementation section on page for details about displaying the payment form.

CFML

Copy and paste the code below into the <HEAD> element of the web page that will display the payment form:

```
<cfhttp
url="https://secure.cpteller.com/api/custapi.cfc?method=autotermi
nal" method="post" result="autoterminal" charset="utf-8">
<cfhttpparam type="formfield" name="merchantkey"
value="YOUR_MERCHANTKEY">
<cfhttpparam type="formfield" name="apikey" value="YOUR_APIKEY">
<cfhttpparam type="formfield" name="operatingenv"
value="businessattended">
</cfhttp>
<cfoutput>#autoterminal.filecontent#</cfoutput>
```

Change the values for YOUR_MERCHANTKEY and YOUR_APIKEY to the values provided by IntelliPay.

See the Implementation section on page for details about displaying the payment form.

RUBY

Copy and paste the code below into the <HEAD> element of the web page that will display the payment form:

```ruby
require 'net/http'
require 'net/https'

uri =
URI.parse("https://secure.cpteller.com/api/custapi.cfc?method=aut
oterminal")
https = Net::HTTP.new(uri.host,uri.port)
https.use_ssl = true
req = Net::HTTP::Post.new(uri.path)
req['merchantkey'] = "YOUR_MERCHANTKEY"
req['apikey'] = "YOUR_APIKEY"
req['operatingenv'] = "businessattended"
res = https.request(req)
puts res.body
```

Change the values for “YOUR_MERCHANTKEY” and “YOUR_APIKEY” to the values provided by IntelliPay.

See the Implementation section on page for details about displaying the payment form.

C# .NET

Copy and paste the code below into the <HEAD> element of the ASP web page that will display the payment form:

```
<%@ Import Namespace="System.Net" %>
<%@ Import Namespace="System.IO" %>
<%
HttpWebRequest request = (HttpWebRequest)
WebRequest.Create("https://secure.cpteller.com/api/custapi.cfc?me
thod=autoterminal");
request.KeepAlive = false;
request.Method = "POST";

byte[] postBytes =
Encoding.ASCII.GetBytes("merchantkey=YOUR_MERCHANTKEY&apikey=YOUR
_APIKEY&operatingenv=businessattended");
request.ContentType = "application/x-www-form-urlencoded";
request.ContentLength = postBytes.Length;
Stream requestStream = request.GetRequestStream();
requestStream.Write(postBytes, 0, postBytes.Length);
requestStream.Close();

HttpWebResponse response =
(HttpWebResponse)request.GetResponse();
Response.Write(new
StreamReader(response.GetResponseStream()).ReadToEnd());
%>
```

Change the values for YOUR_MERCHANTKEY and YOUR_APIKEY to the values provided by IntelliPay.

See the Implementation section on page for details about displaying the payment form.

Java

Use this code in your own class. The output of the call() function should go into the <HEAD> of the document.

```
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.URL;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Map;
import java.util.stream.Collectors;
import javax.net.ssl.HttpsURLConnection;

class LightboxSample {
    static String call() {
        try {
            URL url = new
URL("https://secure.cpteller.com/api/custapi.cfc?method=autoterminal");
            HttpsURLConnection con = (HttpsURLConnection) url.openConnection();
            con.setRequestMethod("POST");
            con.setRequestProperty("Content-Type","application/x-www-form-urlencoded");
            Map<String, String> params = new HashMap<>();
            params.put("merchantkey","YOUR_MERCHANT_KEY");
            params.put("apikey","YOUR_API_KEY");
            con.setDoOutput(true);
            con.setDoInput(true);
            con.setConnectTimeout(15000);
            con.setReadTimeout(15000);
            String data = getParamsURLEncoded(params);
            con.setRequestProperty("Content-length", String.valueOf(data.length()));
            DataOutputStream out = new DataOutputStream(con.getOutputStream());
            out.writeBytes(data);
            out.flush();
            out.close();
            String response = new BufferedReader(new
InputStreamReader(con.getInputStream(),
StandardCharsets.UTF_8)).lines().collect(Collectors.joining("\n"));
            return response;
        } catch(IOException e){}
        return null;
    }

    public static String getParamsURLEncoded(Map<String, String> params) {
        StringBuilder sb = new StringBuilder();
        int count = 0;
        try {
            for (Map.Entry<String, String> entry : params.entrySet()) {
                if(count > 0)
                    sb.append("&");
                sb.append(URLEncoder.encode(entry.getKey(), "UTF-8"));
                sb.append("=");
                sb.append(URLEncoder.encode(entry.getValue(), "UTF-8"));
                count++;
            }
        } catch(UnsupportedEncodingException e) { //Can't happen
        }
        return sb.toString();
    }
    public static void main(String[] args){
        System.out.println(call());
    }
}
```

Change the values for YOUR_MERCHANTKEY and YOUR_APIKEY to the values provided by IntelliPay.

See the Implementation section on page for details about displaying the payment form.

VB.NET

Copy and paste the code below into the <HEAD> element of the ASP web page that will display the payment form:

```
<%@ Import Namespace="System.Net" %>
<%@ Import Namespace="System.IO" %>
<%
```

```
Dim request as HttpWebRequest =
WebRequest.Create("https://secure.cpteller.com/api/custapi.cfc?me
thod=autoterminal")
request.KeepAlive = False
request.Method = "POST"

Dim postdata As String =
"merchantkey=YOUR_MERCHANTKEY&apikey=YOUR_APIKEY&operatingenv=bus
inessattended";
request.ContentType = "application/x-www-form-urlencoded";
request.ContentLength = postdata.Length;

Dim writer As New StreamWriter(request.GetRequestStream(),
System.Text.Encoding.UTF8)
writer.Write(postdata)
writer.Close();

HttpWebResponse response =
(HttpWebResponse)request.GetResponse();
Response.Write(new
StreamReader(response.GetResponseStream()).ReadToEnd());
%>
```

Change the values for YOUR_MERCHANTKEY and YOUR_APIKEY to the values provided by IntelliPay.

See the Implementation section on page for details about displaying the payment form.

Node.js (Express.js)

Follow the steps below to incorporate Lightbox EMV Terminal into your Node.js project:

1. Add an endpoint to your project that you will call to retrieve your authentication credentials. You will use the credentials to set values for your merchant key and API key.

6. Add the code below to your client-side project. This code retrieves your credentials from the endpoint your created in step 1, uses them to set values for YOUR_MERCHANT_KEY and YOUR_API_KEY, and sends a POST request to the Lightbox EMV Terminal autoterminal endpoint:

```
const express = require('express');
const axios = require('axios');
const qs = require('qs');
const app = express();

// You can name this endpoint whatever you would like
app.get('/api/lightbox_credentials', authMiddleware, (req,
res, next) => {
   const options = {
     method: 'POST',
     headers: { 'content-type':
'application/x-www-form-urlencoded' },
     data: qs.stringify({
       merchantkey: '<YOUR_MERCHANT_KEY>',
       apikey: '<YOUR_API_KEY>',
       operatingenv: 'businessattended'
     }),
     url:
'https://secure.cpteller.com/api/custapi.cfc?method=autotermin
al',
   };

   axios(options).then((response) => {
     res.send(response.data);
   });
});
```

7. Add the code below to the endpoint you created in step 1. Send a GET request to your endpoint. The code below returns the code you need to add to your web page:

```
axios.get(''http://<YOUR_SERVER_URL>/api/lightbox_credentials'
).then(response => {
     // This block of code will inject the lightbox code into
your page
    var rg=document.createRange();

document.documentElement.appendChild(rg.createContextualFragme
nt(response.data));
    intellipay.initialize();
});
```

8. Add a button to your web page that displays the Lightbox EMV Terminal as shown in the sample code below:

```
<button data-ipayname="submit" class="ipayfield">Open
Lightbox</button
```

See the Customization section on page for details about customizing the appearance of the payment form. See the Implementation section on page for details about displaying the payment form and submitting the transaction.

REACT

Copy and paste the code below into your REACT project to display the payment form and process payments using the Lightbox EMV Terminal:

```
import React, { Component } from 'react';
import axios from 'axios';

class PaymentPage extends Component {
   componentDidMount() {

axios.get(''http://<YOUR_SERVER_URL>/api/lightbox_credentials').t
hen(response => {
           var rg=document.createRange();

document.documentElement.appendChild(rg.createContextualFragment(
response.data));
           intellipay.initialize();
       });
   }

   render () {
       return (
           <div>
               <button data-ipayname="submit"
class="ipayfield">Open Lightbox</button>
               <input className="ipayfield"
data-ipayname="amount" type="hidden" value="1.00" />
           </div>
       );
   }
}

export default PaymentPage;
```

See the Implementation section on page for details about displaying the payment form.

Angular

Copy and paste the code below into your Angular project to display the payment form and process payments using the Lightbox EMV Terminal:

```
import { Component, OnInit, HostListener } from '@angular/core';
import { HttpClient } from '@angular/common/http';
@Component({
 selector: 'post-request-error-handling',
 templateUrl: 'post-request-error-handling.component.html',
})
export class PostRequestErrorHandlingComponent implements OnInit {
 errorMessage;
 constructor(private http: HttpClient) {}
 ngOnInit() {
   var url = 'https://test.cpteller.com/api/custapi.cfc?method=autoterminal';
   const formData = new FormData();

   //Put in your credentials below.
   formData.append('merchantkey', 'YOUR_MERCHANTKEY');
   formData.append('apikey', 'YOUR_APIKEY');

   this.http.post<any>(url, formData, { responseType: 'text' }).subscribe(
     (response) => {
       var rg = document.createRange();
       document.documentElement.appendChild(
         rg.createContextualFragment(response)
       );
       intellipay.initialize();
     },
     (err) => {
       console.log(err.message);
     },
     () => {
       console.log('completed');
     }
   );
 }
 openIntelliPayLightBox() {
   document.getElementById('intelliPayButton').click();
 }
 isIntelliPayEvent(e) {
   var domain = e.origin.substring(
     e.origin.indexOf('.') + 1,
     e.origin.lastIndexOf('.')
   );
   return domain === 'cpteller';
 }
 @HostListener('window:message', ['$event'])
 postMessage(event) {
   console.log('operation', event.data.operation);
   if (this.isIntelliPayEvent(event) && typeof event.data.operation !==
'undefined' ) {
     switch (event.data.operation) {
       case 'isready':
         //This will load the lightbox onload.
         document.getElementById('intelliPayButton').click();
         break;
       case 'approval':
         // event.data.response this will give you addational details.
         console.log('payment was approved');
         break;
       case 'decline':
         // event.data.response this will give you addational details.
         console.log('payment was decline');
         break;
       case 'closemodal':
         this.errorMessage =
           'Payment Window closed. Reopen it by clicking here or  by clicking
the button Pay Via IntelliPay below.';
         break;
       default:
         console.log('default event not created.');
     }
   }
   console.log('any message', event.data);
 }
}
```

See the Implementation section on page for details about displaying the payment form.

Caveats

1. Behind the scenes we use message passing to communicate between the Lightbox EMV Terminal and your web app.

   If you use JavaScript code that listens to the window "message" event you must still allow non-target events (those of a different event.origin) to propagate.

   To do this we recommend you check event.origin to exclude events from unexpected sources. If the X does not close the frame it's likely the event is being captured and not propagated.

   See the postMessage API references for more details:
   https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage

2. If you have a Content Security Policy on the webpage with the Lightbox embedded, you'll need to add https://*.cpteller.com/ to your script-src and frame-src policies. If don't have both a script-src and frame-src policy, but you have a restrictive child-src then you'll have to add https://*.cpteller.com/ to your child-src policy. See https://developers.google.com/web/fundamentals/security/csp for information about Content Security Policies.