Version 6.18.6

PROTECTPAY® PAYER MANAGEMENT INTERFACE: HOSTED PAYMENT PAGE (HPP)

Instructions to Interface with ProPay ProtectPay Payer Management Interface

PROPAY
A TSYS® Company

Contents

The ProtectPay Payer Management Interface: Hosted Payment Page (HPP) is a Payer Management Interface (PMI) that allows merchants to maintain a payment page that mirrors the look and feel of their website without storing, transmitting or processing the data that their payment pages collect. The Hosted Payment Page is based on an HTTP redirect to enable cross origin browser processes for a client system.

How to use this manual

This manual is designed to facilitate developers in building software solutions to consume the Hosted Payment Page. It is not written for a single development platform. It provides basic information required to properly interact with the Hosted Payment Page.

A developer should have an understanding of Hyper Text Transfer Protocol (HTTP) communication, the consuming of external Web services, Web Form POST methodology, AJAX request and Advanced Encryption Standard (AES) encryption using the Cipher Block Chaining (CBC) mode of operation, Cross Origin Resource Sharing (CORS) security standards and creating a Secure Sockets Layer (SSL) connection on the intended development platform.

While ProPay offers resources and materials that assist in creating and developing software solutions it is the responsibility of the integrating developer to design and develop his or her own software solution on the intended development platform to make use of and consume the services offered by ProPay.

Updated manuals can always be found at www.propay.com/Resources.

Additional Resources

- See ProtectPay API Manual for ProtectPay API Methods that are referenced in this manual.
- See ProtectPay API Manual Appendix A for a list of response values returned by ProtectPay.
- See ProtectPay API Manual Appendix B for a list of supported Processors, Gateways and Service Providers.
- See ProtectPay API Manual Appendix C for a list of supported Swipe Devices.

Disclaimer

ProPay provides the following documentation on an "AS IS" basis without warranty of any kind. ProPay does not represent or warrant that ProPay's website or the API will operate securely or without interruption. ProPay further disclaims any representation or warranty as to the performance or any results that may be obtained through use of the API.

Regardless of its cause, ProPay will not be liable to client for any direct, indirect, special, incidental, or consequential damages or lost profits arising out of or in connection with client's use of this documentation, even if ProPay is advised of the possibility of such damages. Please be advised that this limitation applies whether the damage is caused by the system client uses to connect to the ProPay services or by the ProPay services themselves.

1.0 Processing with the Hosted Payment Page

The Hosted Payment Page (HPP) is a Payer Management Interface (PMI) of the ProtectPay Application Programming Interface (API). ProtectPay ensures the payers' payment information is collected, updated, and stored in accordance with PCI standards. The Hosted Payment Page is an HTML5 page hosted by a secure ProtectPay server. The merchant's checkout page will be redirected to this page for secure payment. The Hosted Payment Page enables a merchant to collect sensitive payment method information without having it traverse the merchant's system. This minimizes the merchants PCI compliance requirements and limits the risk and exposure of the merchant by not handling sensitive payment information.

Why the Hosted Payment Page

The integration option that best alleviates the burden of PCI compliance is this. Because cardholder data is collected on a page completely hosted by Propay, a merchant can avoid many costly requirements associated with demonstrating compliance. At the same time, the HPP has several features that create a customized checkout process for a merchant's website.

Hosted Payment Page Processing Options

The Hosted Payment Page can be configured to perform various payment method storage and/or processing requests. These include:

- Create a PaymentMethodId
- Create and Authorize a PaymentMethodId for a specified amount
- Create and Process a PaymentMethodId for a specified amount
- Authorize a payment method for a specified amount
- Process a payment method for a specified amount
- Authorize a payment method for a specified amount and create a PaymentMethodId only if successful

1.1 Summary of Processing

1. A customer finishes shopping and clicks on a link to check out.
2. Before the customer is able to see the checkout page, the merchant should make a call to the ProtectPay API to Create a Hosted Transaction Instance. (This is where much of the customization of cardholder experience takes place)
3. The merchant redirects the cardholder's browser to the HPP, including the Hosted Transaction ID in the redirect.
4. The cardholder provides payment details and hits submit.
5. ProtectPay processes the request and redirects the cardholder browser to a return URL specified when the Hosted Transaction Instance was configured.
   Http://www.yourreturnurl.com/?result=success
   Http://www.yourreturnurl.com/?result=failure&message=errormessage)
6. The merchants uses the ProtectPay API to Get Hosted Transaction Results

1.2 Best Practices

- The HPP is only required when creating a PaymentMethodId, or processing payment method information without wanting to store it. Once a PaymentMethodId is created it should be processed against directly using the ProtectPay API.
- The Hosted Payment Page will perform card number validation against a Mod 10 using the LUHN algorithm, CVV length validation, card expiration date validation and card number for card type validation upon submission. If an error is detected the form will raise a submission error for the merchants checkout page to handle and display an error on the Hosted Payment Page.
- Credit card transactions can take several seconds to process. This is caused by several variables with the gateway, the processor, and the issuer. There will be a wait during which a cardholder may become impatient. ProPay has provided a warning against clicking the back button, or refresh while a payment method is processing.
- Do not create a HostedTransactionIdentifier with the Name or AVS fields set to required and then use CSS to hide them. This will create an 'uncool' experience.
- Do not create a HostedTransactionIdentifier without a Return URL. It's probably a good thing that this raises an error and doesn't allow the transaction to process.

2.0 Testing and Certification

To improve the customer experience, ProPay requires that developers test their software solutions before receiving credentials to process live transactions. Doing so ultimately improves the end-user experience so please plan accordingly and develop a timeline that provides for testing and certification against the ProPay Integration environment. Integrating a developed software solution to the ProPay web integration requires the following steps:

1. Request API credentials from your sales representative or account manager. By involving him or her in the process, Propay can provide you with guidance about the methods required for your project's scope.
2. Design, build, and test your solution using the ProtectPay integration environment.
3. Contact your Project Manager when you believe you are ready to certify. Your PM will go over your integration with you. (This is a relatively informal process, but one that ensures you've covered all your bases.)
4. Request Production (Live) Credentials from your Project Manager.

Production URLs

The ProtectPay Production REST base URI: https://api.propay.com/protectpay
The ProtectPay Production SOAP URI: https://api.propay.com/protectpay/sps.svc
The ProtectPay Production WSDL URI: https://api.propay.com/protectpay/sps.svc?wsdl
The ProtectPay Production WSDL single file URI: https://api.propay.com/protectpay/sps.svc?singlewsdl
The HPP URL: https://protectpay.propay.com/hpp/v2

Test URLs

The ProtectPay Integration REST base URI: https://xmltestapi.propay.com/protectpay
The ProtectPay Integration SOAP URI: https://xmltestapi.propay.com/protectpay/sps.svc
The ProtectPay Integration WSDL URI: https://xmltestapi.propay.com/protectpay/sps.svc?wsdl
The ProtectPay Integration WSDL single file URI: https://xmltestapi.propay.com/protectpay/sps.svc?singlewsdl
The HPP URL: https://protectpaytest.propay.com/hpp/v2

Live Credentials MUST be kept confidential

2.1 Troubleshooting and Technical Support

Your Project Manager acts as a technical resource during integration and will assist you with trouble shooting problems encountered while you work on your solution. In an effort to make this possible, you should be prepared to provide the following information when you encounter a problem during integration:

1. Timestamp of the incident (specify time zone)
2. URI Requests are being made to
3. HTTP Method being used
4. XML/SOAP/JSON data passed to the URI
5. XML/SOAP/REST/HTTP Response received.

Despite all the best preparations, planning and testing there are occasions where errors can occur when transitioning from the testing systems to the live environment. Providing less information may result in a delay to any technical support questions regarding the Application Programming Interface. The ProPay Technical Support team can only assist in the troubleshooting of the API and not a client's software solution when undesired effects occur in a client's software solution when consuming the ProtectPay API.

Limitations based on a supported gateway

ProtectPay is works with multiple gateways over which ProPay has no control. As such there are instances where a gateway may return an error with a transaction passed to it from ProtectPay. These errors are indicated by the 200 series in Appendix.2. If a transaction request returns a 200 series error ProPay technical support can only troubleshoot that the MerchantProfileId is setup properly according to the specifications found in Appendix B, and upon request, provide the raw request to and response from the gateway.

Should a client require additional troubleshooting they should contact the Processor Gateway for an explanation of their specific failure. ProPay Technical Support cannot troubleshoot non ProPay merchant account issues.

3.0 Technical Implementation

Required ProtectPay API Methods:

The following ProtectPay API methods are needed to implement a HPP solution:

- Create a Payer
- Create Hosted Transaction Instance
- Get Hosted Transaction Results

Web Browser support:

Desktop Browser	Version
Internet Explorer	10+
Microsoft Edge	13+
Fire Fox	44+
Safari	9+
Google Chrome	47+
Opera	36+
Chromium	47+

Mobile Browser	Version
Safari	8.4+
Android Browser	4.4+
Blackberry Browser	10+
Opera Mobile	12+
Chrome	49+
Fire Fox	45+
IE	11+
UC	9.9+

This list is not exhaustive but covers the most popular browsers

Transport Layer Security (TLS):

ProPay recognizes the importance of handling financial transactions in a secure manner and ensures that ProtectPay offers the best transmission security available. ProPay ensures that ProtectPay API request information is transmitted using the latest Transport Layer Security (TLS) encryption practices. TLS creates a secure connection between client and server over which encrypted information is sent. ProPay hosts the SSL certificate for this connection type. Each ProtectPay API method request, regardless of the interface, will negotiate an SSL connection automatically over port 443.

3.1 Possible Load Errors

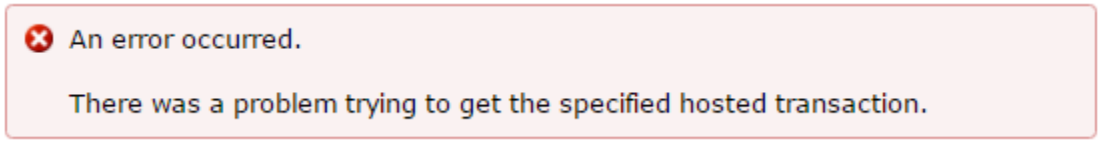
There are multiple conditions where your redirect to the HPP might fail. This section describes those conditions and what the user will see if they occur.

Passing an Invalid Hosted Transaction ID

If an invalid HostedTransactionIdentifier is used to retrieve the Hosted Payment Page the following error will be displayed by the Hosted Payment Page:

Passing an Already-Consumed Hosted Transaction ID

A HostedTransactionIdentifier is a onetime use token. If in the event a client attempts to use a HostedTransactionIdentifier that has already been used the following error will be displayed by the Hosted Payment Page:



Sending to the HPP when your Instance lacks a Return URL

If a Return URL is not provided when you created a Hosted Transaction Instance, the following error message will be displayed by the Hosted Payment Page:

3.2 Configuring the HPP with API parameters

The Hosted Payment Page configuration is set when creating the HostedTransactionIdentifier. There are several options available to define the functionality of the page your cardholder will see.

In order to use the Hosted Payment Page to store payment methods only please set the following attributes when creating the HostedTransactionIdentifier. The Hosted Payment Page will store the payment method information and the results will return the PaymentMethodId under the PayerId that is used to create the HostedTransactionIdentifier and once stored the PaymentMethodId can be processed against directly using the ProtectPay API.

Desired Behavior	StoreCard	AuthOnly	ProcessCard	OnlyStoreCardOnSuccessfulProcess
Store Payment Method Only	True	False	False	False
Validating Card Data prior to storage	False	True	False	True
Authorize Payment Method Only	False	True	False	False
Process Payment Method Only	False	False	True	False
Process and Store Payment Method	True	False	True	False
Process and Store if Payment Succeeds	False	False	True	True

The minimum authorization amount is 1.00 [Currency]. If a merchant wishes to use an authorization request to 'validate' a card but does not wish to process, ProPay recommends a minimum authorization and then an immediate void

3.3 Preloading Information with API Parameters

In order to provide an enhanced customer user experience, the Hosted Payment Page can load with non-sensitive payment information prefilled. The following fields can be pre-populated into the Hosted Payment Page. (See ProtectPay API documentation for details.)

In order to preload the Payer Name field the **CardHolderNameRequirementType** must also be set to 1 or 2.

- Payer Name

In order to preload address information, the **AvsRequirementType** must also be set to 1 or 2.

- Address 1
- Address 2
- City
- State
- Postal Code
- Country

3.4 Response Handling

Return Response:

Once the payment done successfully Hosted Payment Page redirect to the ReturnURL passed when the HPP was configured by Create Hosted Transaction Instance.

- On Success: ReturnURL+ ?result=success (e.g. www.abc.com?result=success) This includes declined transactions
- On Cancel: ReturnURL+ ?result=cancel (e.g. www.abc.com?result=cancel)
- On Failure: ReturnURL+ ?result=failure&message=error message (e.g. www.abc.com?result=failure&message=errormessage)

Optional – Use results from a failure

When a transaction is declined, the Get Hosted Transaction Results returns several non-sensitive data points from the attempt. Using this information in a subsequent Create Hosted Transaction Instance call can create a better user experience because the cardholder will not need to re-enter data.

Hosted Transaction Result Attribute	CreateHostedTransactionIdentifier Attribute
PaymentMethodInfo.AccountName	Name
PaymentMethodInfo.Description	Description
PaymentMethodInfo.BillingInformation.Address1	Address1
PaymentMethodInfo.BillingInformation.Address2	Address2
PaymentMethodInfo.BillingInformation.City	City
PaymentMethodInfo.BillingInformation.State	State
PaymentMethodInfo.BillingInformation.ZipCode	ZipCode
PaymentMethodInfo.BillingInformation.Country	Country

4.0 Customizing the Look and Feel

CSS3:

The Hosted Payment Page can be customized to match the look at feel of a merchant's checkout page. One of the parameters passed when creating a Hosted Transaction Instance is 'CSSUrl'. If passed the Hosted Payment Page will reference the CSS file hosted there. The file must be accessible to the HPP or it will load with its default style.

Default CSS File

The ProtectPay Integration environment default CSS file: https://protectpaytest.propay.com/hpp/css/pmi.css You can use this to help build your own.

Notes on elements' styling

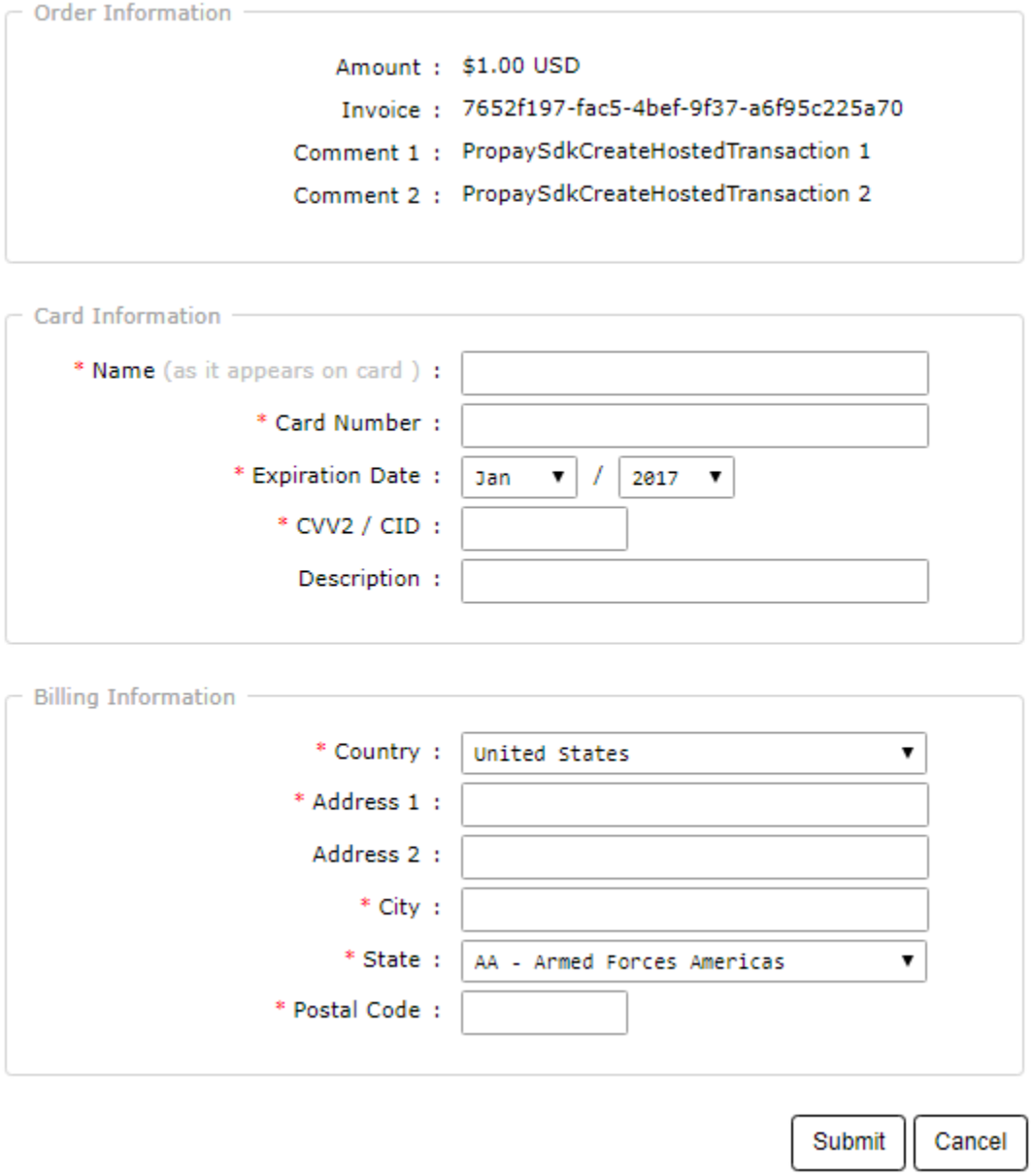
The following elements may not be immediately obvious and warrant extra explanation in how they impact styling.

The HPP-CSS sample application is designed to be used by developers integrating the Hosted Payment Page into their software solutions by providing the raw HTML for both Hosted Payment Page configurations as well as the default CSS file and a blank css file containing all the classes contained in the HTML. This will allow a developer to more quickly create a custom css, if desired, before creating a Hosted Transaction Identifier.

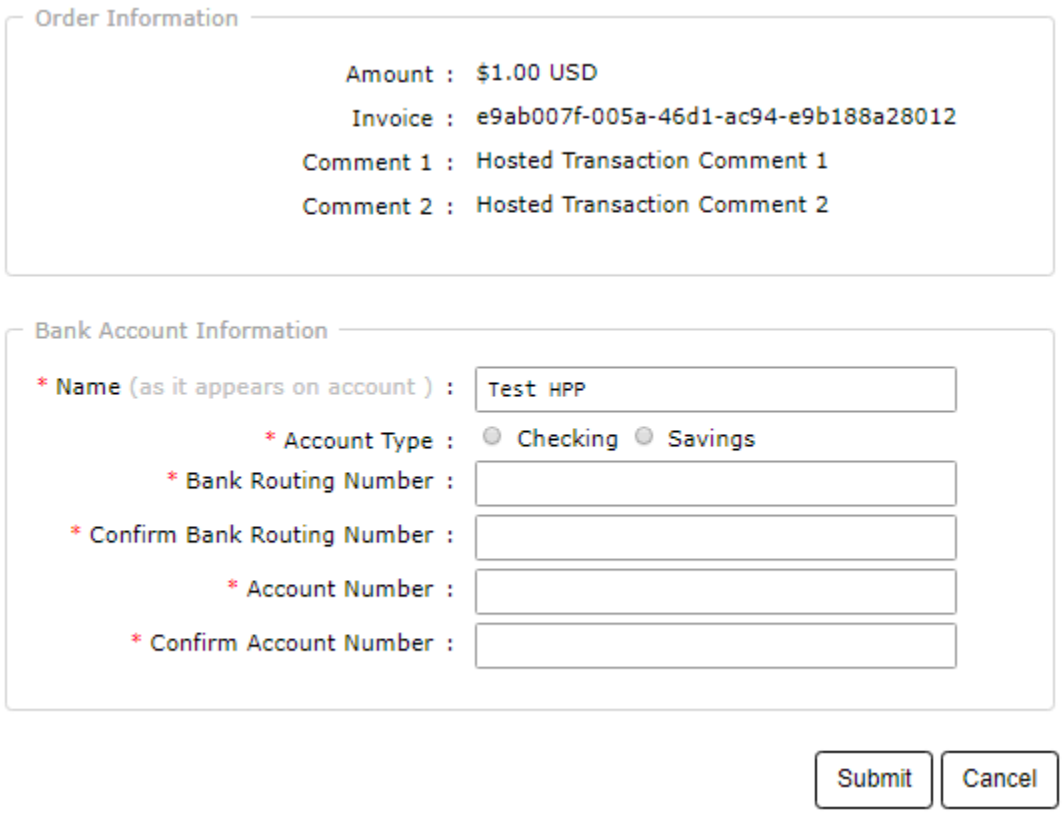
'Gotchas' (Things that may not be immediately obvious

.ErrorBackground - Used to provide highlighting to one of the ProPay error messages WHEN ErrorMessage is Displayed
.AcceptOnlyNumbers - Class applicable to multiple elements. Enables a check that data entered allows only for numeric characters.
.field-validation-error { display: table } - Text included when a validation error occurs underneath the Text Entry
.validation-summary-errors { display: none } -
.validation-summary-valid { display: none } -

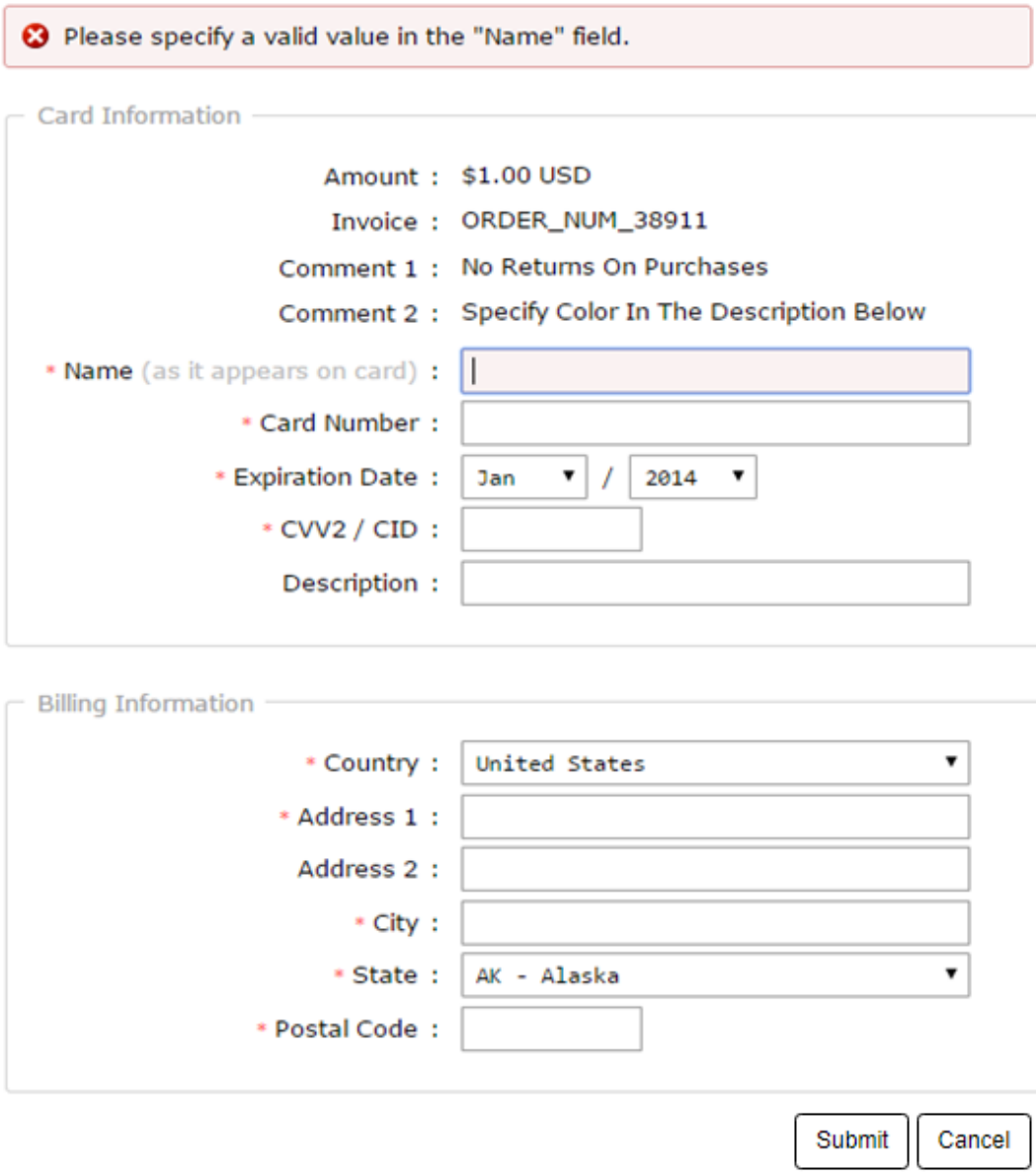
Default Appearance of the Hosted Payment Page (Credit Card Transactions)



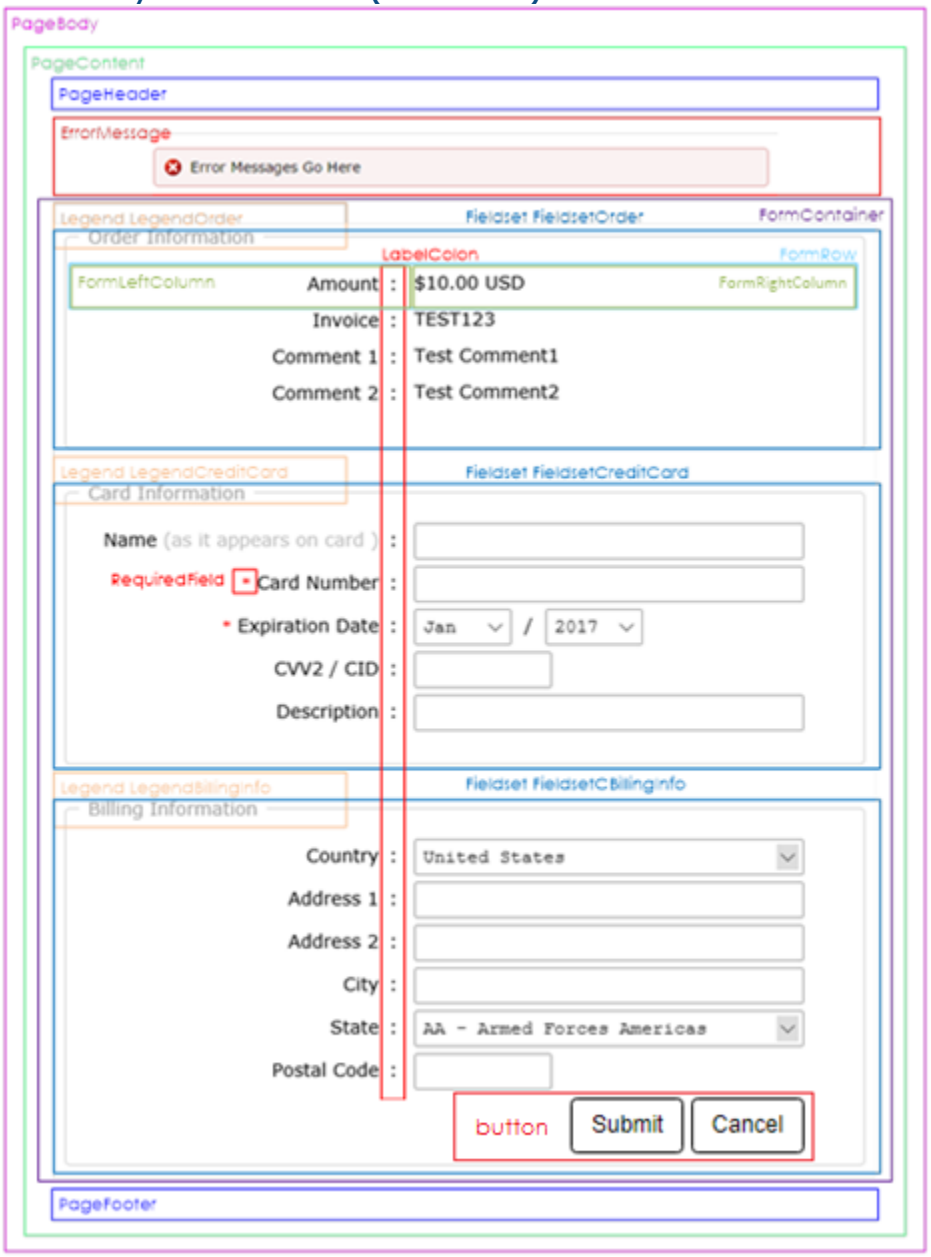
Default Appearance of the Hosted Payment Page (ACH Transactions)



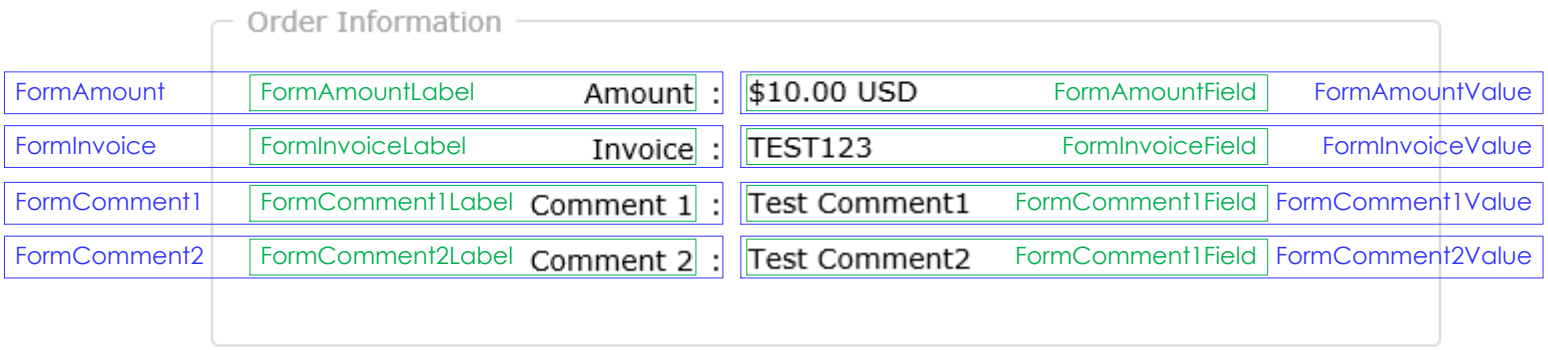
Default appearance of HPP when failing form validation (Error Style is Identical for both Credit Card and ACH)



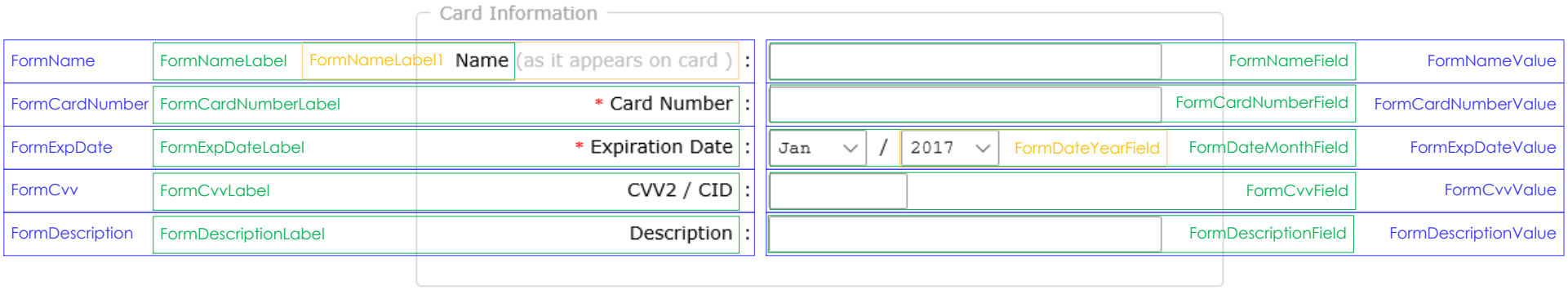
HPP CSS Style Section Guide (Credit Card)



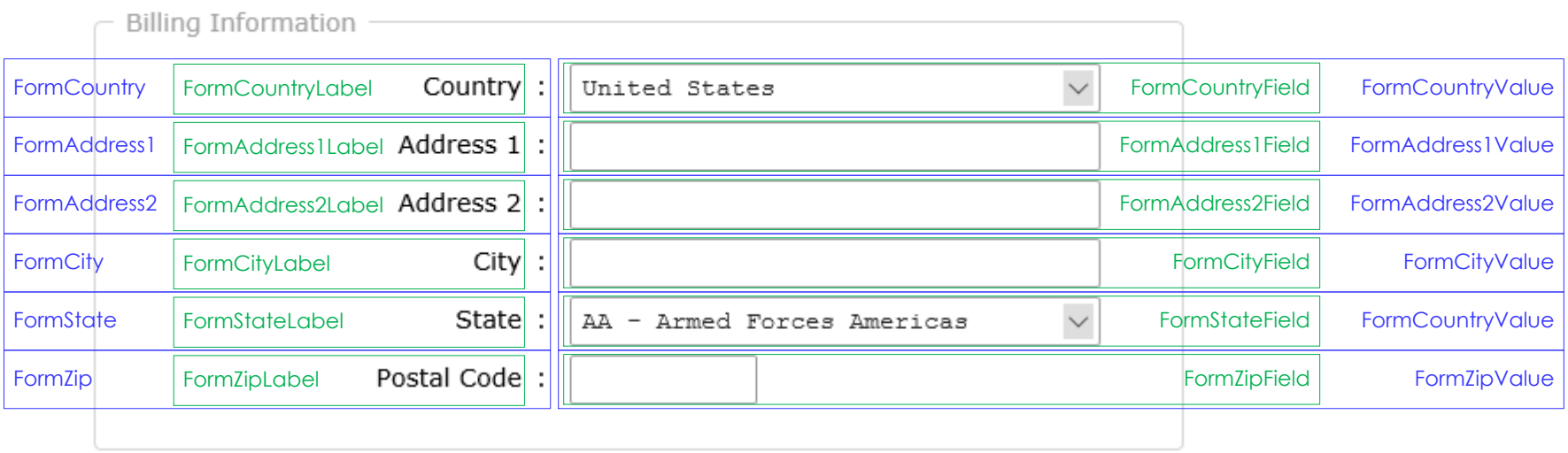
- Bank Account Information inherits LegendCreditCard and FieldsetCreditCard for simplicity of CSS for both payment types



- All Labels inherit the "Label" class

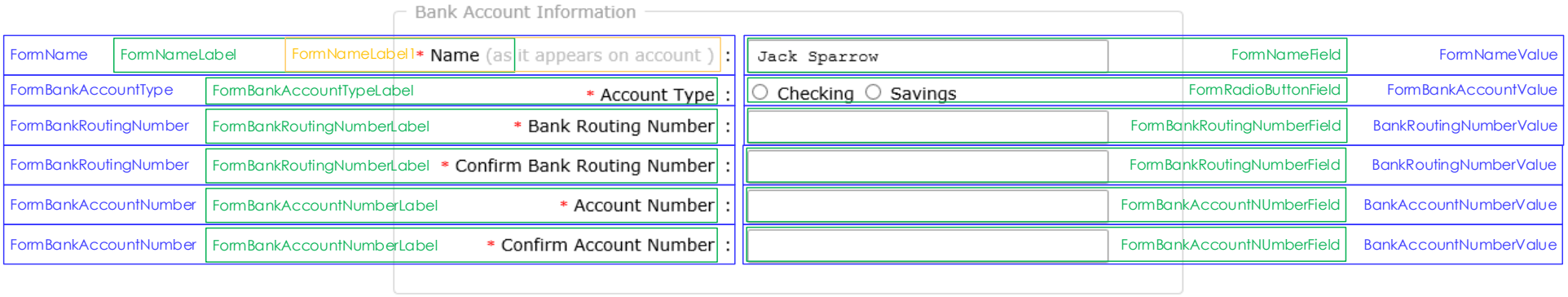


- All Labels inherit the “Label” class
- All TextFields inherit the “TextBox” Class
- All Drop Down Lists inherit the “DropDownClass”



- All Labels inherit the "Label" class
- All TextFields inherit the "TextBox" Class
- All Drop Down Lists inherit the "DropDownClass"

HPP CSS Style Element Guide (ACH Information)



- All Labels inherit the "Label" class
- All TextFields inherit the "TextBox" Class
- All Radio Buttons Down Lists inherit the "RadioButton" Class

- All Buttons inherit the “button” class

Mobile Viewport CSS Customization Example

The following CSS snippet will help a developer to be able to change the default sizes of elements of the Hosted Payment Page for mobile view ports.

```
/*ProPay provides the following code "AS IS."
*ProPay makes no warranties and ProPay disclaims all warranties and conditions, express, implied or statutory, including without limitation the implied warranties of title, non-infringement, merchantability, and fitness for a particular purpose.
*ProPay does not warrant that the code will be uninterrupted or error free, nor does ProPay make any warranty as to the performance or any results that may be obtained by use of the code.*/

.FormContainer {
    width: 570px;
    box-sizing: border-box;
    background-color: #ffffff;
    padding: 20px 20px;
    margin-top: 0px;
    margin-bottom: 35px;
    max-width: 1000px;
    width: 100%;
    margin: 0 auto;
    text-align: center;
    font: bold 14px sans-serif;
}

.FormContainer .FormRow {
    text-align: left;
    margin-bottom: 10px;
}

.FormContainer FormLeftColumn {
    text-align: center;
    margin-bottom: 10px;
}

.FormContainer .FormNameLabel {
    display: inline-block;
    text-align: left;
    padding: 0 0 10px;
}

.FormContainer input {
    color:  #5f5f5f;
    box-sizing: border-box;
    width: 200px;
    box-shadow: 1px 2px 4px 0 rgba(0, 0, 0, 0.08);
    padding: 5px 5px;
    border: 1px solid #dbdbdb;
}
```

```
.FormContainer select {
    background-color: #ffffff;
    color:  #5f5f5f;
    box-sizing: border-box;
    max-width: 240px;
    box-shadow: 1px 2px 4px 0 rgba(0, 0, 0, 0.08);
    padding: 12px 8px;
    border: 1px solid #dbdbdb;
}

.FormLeftColumn {
                font-weight: bold;
    display: inline-block;
                width: 49%;
}

.FormRightColumn {
    display: inline-block;
                width: 49%;
                text-align: right;
}

.FieldSet {
                margin-top: 20px;
                padding-top: 20px;
}

.TextBox {
                -webkit-appearance: none;
                -webkit-border-radius: 0;
                box-sizing: border-box;
                border: 1px solid #dfe3e9;
                padding: 0 .7em;
                border-radius: 4px;
                font-size: 13px;
                height: 33px;
                line-height: 33px;
                box-shadow: none;
                outline: none;
                color: #354052;
                width: 100%;
}

.FormLeftColumn {
                font-weight: bold;
                font: 400 16px/1.2 "proxima-nova", sans-serif;
                color: #7f8fa4;
                font-size: 14px;
                line-height: 1.2;
                font-weight: regular;
```

```
}

.FieldSet {

		border: 0px;
		margin-top: 20px;
		padding-top: 20px;
}

.Legend{
		font-weight: bold;
		font: 600 18px/1.4 "proxima-nova", sans-serif;
		color: #7f8fa4;
}
```

Language Customization CSS Example

The following CSS snippet will help a developer to be able to change the default language of the Labels in the Hosted Payment Page to various languages.

```
/*ProPay provides the following code "AS IS."
*ProPay makes no warranties and ProPay disclaims all warranties and conditions, express, implied or statutory, including without limitation the implied warranties of title, non-infringement, merchantability, and fitness for a particular purpose.
*ProPay does not warrant that the code will be uninterrupted or error free, nor does ProPay make any warranty as to the performance or any results that may be obtained by use of the code.*/

/*The following CSS will changed the language for "Order Information" in LegendOrder from English to Spanish*/

.LegendOrder{
    color: transparent;
}
.LegendOrder::after{
    content: "Información del Pedido";
    color: #A3A3A3;
}
```

Drop Down Localization CSS Customization

The following CSS snippet will help a developer to be able to hide non-applicable countries in the drop down list in the Hosted Payment Page.

```
/*ProPay provides the following code "AS IS."
*ProPay makes no warranties and ProPay disclaims all warranties and conditions, express, implied or statutory, including without limitation the implied warranties of title, non-infringement, merchantability, and fitness for a particular purpose.
*ProPay does not warrant that the code will be uninterrupted or error free, nor does ProPay make any warranty as to the performance or any results that may be obtained by use of the code.*/

/*The following CSS will hide all the country options and only display the United States and Canada*/


.FormCountryField{}
.FormCountryField option{display:none}
.FormCountryField option:nth-child(1){display:inline}
.FormCountryField option:nth-child(2){display:inline}


/*The following CSS will hide the non-Continental States, excluding HI and AK providing a list comprised of only the 50 States of the United States of America*/

.FormStateDropDown{}
.FormStateDropDown option:nth-child(1){display:none}
.FormStateDropDown option:nth-child(2){display:none}
.FormStateDropDown option:nth-child(5){display:none}
.FormStateDropDown option:nth-child(7){display:none}
.FormStateDropDown option:nth-child(16){display:none}
.FormStateDropDown option:nth-child(45){display:none}
.FormStateDropDown option:nth-child(58){display:none}
```

MasterPass CSS Classes

/*ProPay provides the following code "AS IS."
*ProPay makes no warranties and ProPay disclaims all warranties and conditions, express, implied or statutory, including without limitation the implied warranties of title, non-infringement, merchantability, and fitness for a particular purpose.
ProPay does not warrant that the code will be uninterrupted or error free, nor does ProPay make any warranty as to the performance or any results that may be obtained by use of the code./

.masterpass-error-message { padding: 10px;font-size: 15px;font-family:Arial,sans-serif; } - class use to display error message in case masterpasss transaction cannot be initiated.

.masterpass-result-message{} – class use to display result message after getting result from MasterPass i.e. click master pass button-> Web page will navigate to master pass page for user to enter details and either of following 3 will be the result.

.masterpass-result-success {}- Successful authentication from master pass -class use to display result success message
.masterpass-result-cancel {} – Transaction Cancelled by the user - class use to display result cancel message
.masterpass-result-failure {} – Transaction failed - class use to display result failure message

/*The following will display a MasterPass success message*/

```
<div class="masterpass-result-message">
<label class="masterpass-result-success">you have selected to pay with MasterPass</label></div>
```