Version 4.0.1

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

Instructions to Interface with ProPay ProtectPay Payer Management Interfaces.

PROPAY®

A TSYS® Company

Revision History:

Date	Version	Description	Author(s)
2/22/2016	4.0.0	Updated and synchronized with version 4.X.X of the ProtectPay API manual. Additional descriptions added, new diagram added, additional information included Integration Section reworked tables adjusted and samples updated **This manual succeeds the following API Manuals:** **ProtectPay Seamless Payment Interface Manual 3.0.6** ***Previous Manuals should be discarded**	Jared James Steven Barnett Tanner Olsen Elizabeth Thompson
9/22/2016	4.0.1	Updated Style to reflect required changes Updated information to include ACH transaction Corrected minor information Added additional error information Re-arranged structure Added plain text versions of required filed	Jared James Steven Barnett Tanner Olsen
11/09/2016	4.0.2	Updates, Enhancements and Synchronization with the ProtectPay API Manual	Jared James Steven Barnett Tanner Olsen

Contents

1.0 ProPay® ProtectPay® Application Programming Interface

The ProtectPay Payer Management Interface: Hosted Payment Page 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 enables a merchant to collect and process sensitive payment method information on a ProtectPay hosted web page loaded inside an iframe on the merchant's checkout page.

How to use this manual

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

A developer should have an understanding of Hyper Text Transfer Protocol (HTTP) communication, the consuming of external Web services, Web Form POST methodology, Cross Origin Resource Sharing (CORS) security standards, Microsoft® SignalR, JavaScript, jQuery, 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.

For additional development resources please visit: https://developer.propay.com

Additional Resources

See ProtectPay API Manual for API Methods referenced in this manual

Important Concepts

- ProtectPay is not a Processor or Gateway; it is a secure collection of sensitive payment data.
- ProtectPay stores data securely for both single and recurring or subsequent payments using industry best practices.
- ProtectPay utilizes a proprietary interface to process transactions through several major gateways, processors and services providers.
- The Hosted Payment Page will not return the results of a transaction; an additional API call is required to get the results of the transaction.

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.

To receive updates to this API documentation please send a request to requestapiupdates@propay.com.

1.1 Description of 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 displayed in an iframe on the merchant's checkout page. 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 to most fully remove merchants from PCI scope is the Hosted Payment Page. Current web browser security standards prevent a web page from requesting resources from a domain other than the domain or origin of the current page being served (CORS standard). This restriction makes it necessary to create a real time two way communication path between the Hosted Payment Page and the merchant's checkout page. This is accomplished by use of Microsoft SignalR. This creates a proxy environment where a merchant's checkout page can send messages to the Hosted Payment Page and it in return can send messages back to the merchant's checkout page. In this way there is no possibility of a merchant being able to pass any sensitive payment information back to their systems. The Hosted Payment Page is only needed when new payment method information must be collected. The Hosted Payment Page can be configured to create a PaymentMethodId from the payer entered data. Once a PaymentMethodId has been created for the specified PayerId it can be processed against using the ProtectPay API directly while maintaining minimal risk, exposure and PCI compliance scope.

Hosted Payment Page Processing Configurations

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

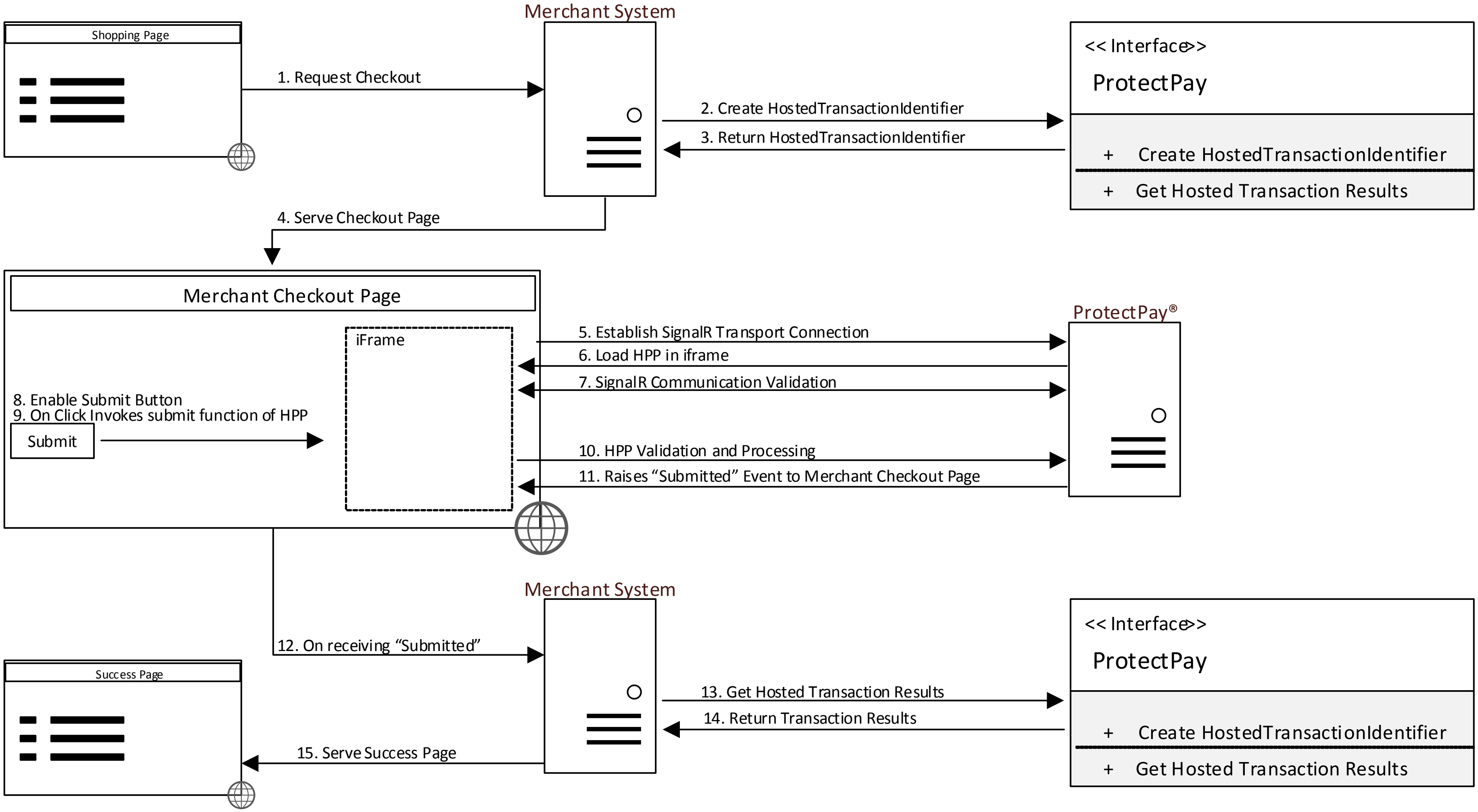
The values required for each configuration are determined by the type of transaction requested.

1.2 Processing with the Hosted Payment Page

Summary of Processing

1. Customer requests checkout from the merchant's system.
2. The merchant's system creates a HostedTransactionIdentifier using ProtectPay API Method 4.7.2 'Create HostedTransactionIdentifier'.
3. The ProtectPay API returns the HostedTransactionIdentifier.
4. The merchants system serves the checkout page.
5. The checkout page establishes a real time SignalR connection to the ProtectPay Hosted Payment Page server.
   - ❖ See section 3.2.0 for a diagram of communication.
6. Using the HostedTransactionIdentifier the merchant's page requests from the ProtectPay server the Hosted Payment Page identified by the HostedTransactionIdentifier which is loaded into an iframe on the merchant's checkout page.
7. The Hosted Payment Page validates communication between itself and the merchant's checkout page.
   - ❖ See section 3.2.0 for a diagram of communication validation.
8. Upon validation the merchant checkout page enables the "submit" button.
   a. This should be tied to the formIsReadyToSubmit() function in the hpp.js.
9. The payer fills in payment method information and then clicks the "submit" button which invokes the submit function of the Hosted Payment Page.
10. The Hosted Payment Page validates the entered information and if passes processes the request.
    a. If the information does not pass validation the form is not submitted and the form displays the error to the user to correct the information entered.
11. Upon successful submission the Hosted Payment Page raises the submitted event on the merchant's checkout page.
12. The merchants system is notified of the successful submission of the Hosted Payment Page.
    a. This should be tied into the formSubmittedSuccessfully() function in the hpp.js.
13. The merchants system gets the results of the Hosted Transaction using ProtectPay API Method 4.7.3 'Get Hosted Transaction Results'.
14. The ProtectPay API returns the results of the Hosted Transaction.
15. The merchants system serves the success page.

Summary of Processing Flow Diagram


Merchant System
Shopping Page
1. Request Checkout
<< Interface>>
ProtectPay
2. Create HostedTransactionIdentifier
3. Return HostedTransactionIdentifier
+ Create HostedTransactionIdentifier
+ Get Hosted Transaction Results
4. Serve Checkout Page
Merchant Checkout Page
iFrame
ProtectPay®
5. Establish SignalR Transport Connection
6. Load HPP in iframe
7. SignalR Communication Validation
8. Enable Submit Button
9. On Click Invokes submit function of HPP
Submit
10. HPP Validation and Processing
11. Raises "Submitted" Event to Merchant Checkout Page
Merchant System
12. On receiving "Submitted"
<< Interface>>
ProtectPay
13. Get Hosted Transaction Results
14. Return Transaction Results
+ Create HostedTransactionIdentifier
+ Get Hosted Transaction Results
Success Page
15. Serve Success Page

2.0 Interface Testing and Certification

To improve the customer experience, ProPay requires that new developers test their software solutions before receiving credentials to process live transactions. This integration process is designed to assist the developer in building a robust solution that can handle and process all the various responses that come from real time credit card and ACH processing. This process ultimately improves the end-user experience. Please plan accordingly when developing timelines and schedules to accommodate for testing against the ProPay Integration environment. Negotiated fees are not refunded in the production environment.

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.

Integrating a developed software solution to the ProPay web integration requires the following steps:

1. Request from a ProPay sales representative and/or account manager that integration API credentials be sent.
2. Begin interfacing the appropriate ProtectPay API methods for specific project scope.
    - ❖ A ProPay sales representative and/or account manager will help determine which methods are required for the specific project scope.
3. Design, develop, build and test the software solution using the ProtectPay integration environment.
    a. The ProtectPay integration Hosted Payment Page base URI: https://protectpaytest.propay.com/hpp/
    - ❖ Testing payment processing requires additional information provided by the client's chosen processor.
4. Certify the developed software solution against the ProPay integration environment.

*Review the status of the integration certification with a ProPay sales representative and/or account manager.

5. Request Production (Live) Credentials from a ProPay sales representative and/or account manager.

    - ❖ Live Credentials MUST be kept confidential

For additional information about ProPay testing and live environments see: ProPay Server Environments.

3.0 Technical Implementation

Required Dependencies:

- Query v. 1.6.4 or later available from
  - Direct download link https://code.jquery.com/jquery-2.2.2.min.js
- Microsoft® SignalR v. 2.0.0 or later available from signalr.net
  - Direct download link: https://github.com/SignalR/SignalR/zipball/master

*The jQuery.js file must be loaded by the browser before the SignalR.js file.

Required ProtectPay API Methods:

- ProtectPay API Method 4.2.1 'Create PayerId'
- ProtectPay API Method 4.7.2 'Create HostedTransactionIdentifier'
- ProtectPay API Method 4.7.3 'Get Hosted Transaction Results'

Web Browser support:

The ProtectPay SignalR server and Hosted Payment Page server utilize Cross Origin Resource Sharing (CORS). They disallow the use of JSONP to connect to and communicate with to increase the level of security provided by the interface. Older web browsers that do not support CORS will be unable to establish a connection to the SignalR server. The following web browsers natively support CORS.

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 and only covers major browsers

Secure Sockets Layer (SSL):

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 Secure Sockets Layer (SSL) encryption practices. SSL 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.

Microsoft® SignalR:

ProPay uses Microsoft SignalR in order to facilitate real time web communication between the merchant's checkout page and the hosted payment page. SignalR is a server side library that takes advantage of several transport layers selecting the best communication layer between the browser client and server transport. In the use case of ProPay the SignalR connection is used to invoke methods between the merchant's checkout page and the Hosted Payment Page that is served by ProtectPay.

ProPay provides a JavaScript library that is used to create the browser client connection to the ProtectPay server. When the hosting page connects to the ProPay SignalR server using the provided JavaScript library it will subscribe itself to the correct hub identified by the HostedTransactionIdentifier. If a client chooses to create their own library their hosting page must do the same and after establishing connection to the ProPay SignalR server must subscribe itself to the correct communication hub identified by the HostedTransactionIdentifier. The Hosted Payment Page, which is the ProPay page, subscribe itself to this same hub once loaded.

- ❖ For additional information see: http://www.asp.net/signalr/overview/getting-started/introduction-to-signalr

Client Side Scripting:

In order to facilitate communication between the Hosted Payment Page and the merchant's checkout page a developer will be required to write some browser client scripting. The scripting should include methods to load the Hosted Payment Page, verify communication and invoke the "Submit" function of the Hosted Payment Page. ProPay offers sample code that accomplishes the required functionality written in JavaScript. The Microsoft SignalR file included is a JavaScript file which is dependent on jQuery and so it makes sense to write client side code in JavaScript. However there are other scripting languages that will accomplish the same functionality and so it is up to the developer to decide based on his or her system which should be used. Whichever scripting language is used, it must be client side as to separate communication and maintain PCI compliance and scope.

- ❖ See Section 3.2.3 hpp.js for the sample JavaScript file.

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 HostedTransactionId is the 'CSSUrl'. If passed the Hosted Payment Page will load in the iframe referencing this CSS file, the file must be publically accessible to the Hosted Payment Page. If the Hosted Payment Page is unable to reach the CSS URL or there was no URL passed the Hosted Payment Page will load with the default style. ProPay offers both a style guide and Hosted Payment Page HTML file to assist developers in styling the Hosted Payment Page to the merchant's specifications.

- ❖ See Section 3.2.4 Customization of the Hosted Payment Page for additional information.

ACH Processing:

The Hosted Payment Page can be used for ACH processing. ACH processing is limited to the Legacy ProPay processor and the receiving ProPay account must be ACH payment enabled with an approved and set SEC code of 'WEB'. Attempts to process against another processor will result in an error being returned after submission. Attempts to process against a ProPay merchant or business account that is not setup for WEB enabled

ACH payments will result in a processor decline of ‘Invalid Standard Class Entry’ when using ProtectPay API method 4.7.3 ‘Get Hosted Transaction Results’.

‘Name’ is a required field for all ACH transaction requests and will be displayed as required independent of the submitted HostedTransactionIdentifier settings as well as AVS information being hidden.

3.1 Best Practices

- A PayerId is required when creating a PaymentMethodId. A PayerId can be created using either ProtectPay API method 4.2.1 'Create PayerId' or by using ProtectPay API method 4.7.1 'Create TempToken'. Once a PayerId is created it should be used in subsequent transaction requests instead of creating a new one for each transaction.

- 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 recommends that developers provide cardholders with a warning against clicking the back button, or refresh on their browser or pressing the rendered 'submit' button while a payment method is processing. ProPay recommends that developers generate a control that displays such a warning during the period of time it takes to receive a response.

- The Hosted Payment Page is served on an SSL connection. To prevent web browsers from automatically blocking the SignalR transport connection and loading of the Hosted Payment Page due to mixed content, the merchant's checkout page should be served on an SSL connection (https). A Developer should direct users who are using browsers that do not support Cross Origin Resource Sharing (CORS) to use a browser that supports CORS prior to serving the merchant's checkout page.

- Do not create a HostedTransactionIdentifier with the Name or AVS fields set to required and then use CSS to hide them. This will create undesired effects.

Recommended API Request Best Practices Use Cases

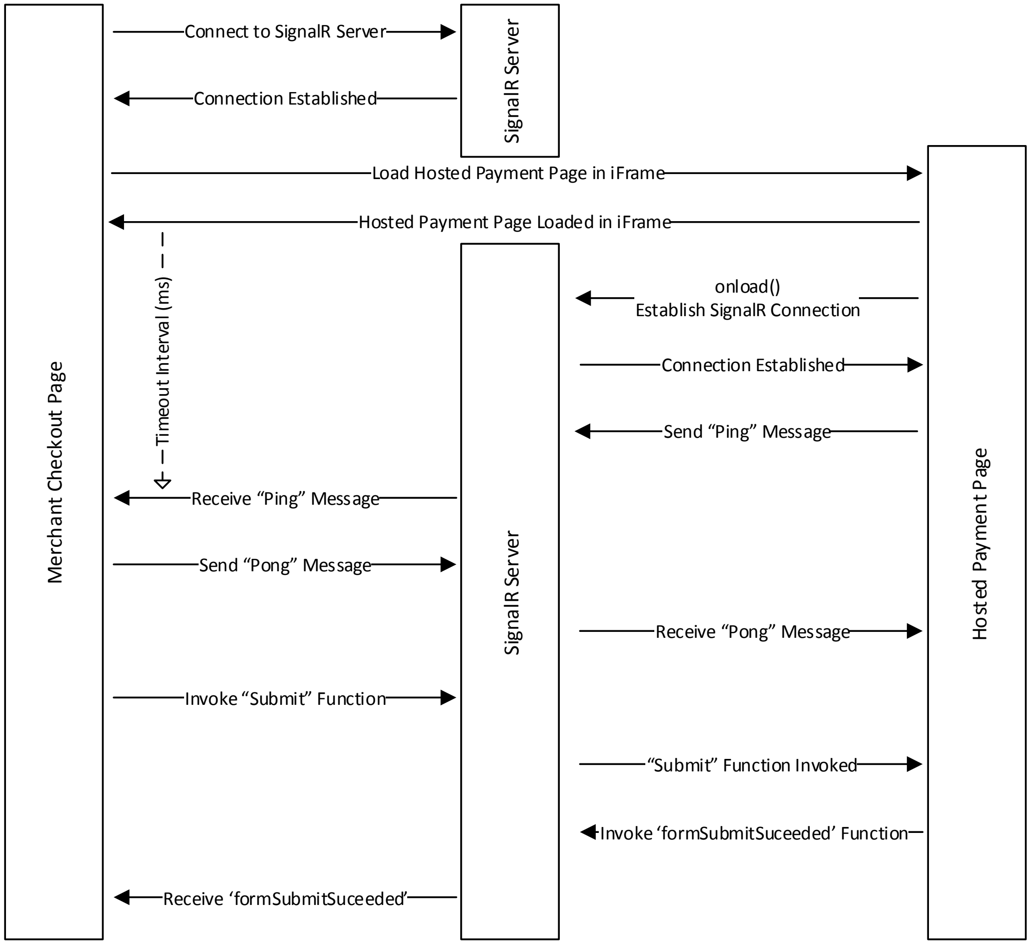
1. Merchant known payer wants to use a merchant stored payment method
   a. Use ProtectPay API method 4.4.1 'Authorize a PaymentMethodId' or 4.5.1 'Process a PaymentMethodId'
2. Merchant known payer wants to add an additional stored payment method to merchant system
   a. Use ProtectPay API method 4.7.1 'Create TempToken' passing in known PayerId as parameter
   b. Set HPP parameter to store payment method
   c. Set HPP parameter to process payment method or not process payment method
3. Merchant unknown payer wants to process a payment method for future use
   a. Use ProtectPay API method 4.7.1 'Create TempToken' passing in unknown payer name as parameter
   b. Set HPP parameter to store payment method
   c. Set HPP parameter to process payment method or not process payment method

4. Merchant known or unknown payer wants to process a payment method and not store it for future use
    a. Use ProtectPay API method 4.7.1 ‘Create TempToken’ with appropriate parameters
    b. Set HPP Parameter to not store payment method
    c. Set HPP parameter to process payment method

3.2 Interface

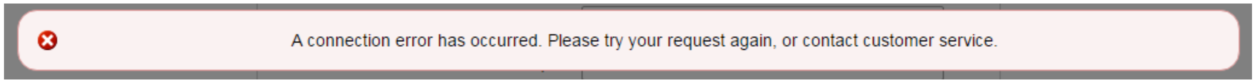
The Hosted Payment Page serves as dual pages in a single browser window from two separate origins communicating through a SignalR server hosted by ProtectPay. The Hosted Payment Page itself has an onload() function to establish communication to the same SignalR server instance the merchant's checkout page is connected to. Once two-way communication is established the form is ready to be submitted.

Overview of Communication



Communication Errors

If in the event of communication negotiation between the merchant's checkout page and the Hosted Payment Page the merchant's checkout page does not respond to the 'ping' request from the Hosted Payment Page through the ProtectPay SignalR server to the merchant's checkout page the Hosted Payment Page will automatically display the following unalterable error:



A connection error has occurred. Please try your request again, or contact customer service.

This error is populated by the Hosted Payment Page itself and should not be confused with a communication timeout. This error indicates the merchant's checkout page is experiencing an inability to connect to the ProtectPay SignalR server, subscribe to the correct SignalR proxy connection Hub, communicate through the SignalR connection or respond to requests from the SignalR connection. It can also indicate that the Hosted Payment Page itself is unable to connect to the ProtectPay SignalR script.

This will raise the **signalR_OnConnectionFailed(**) event in the hpp.js script and can be caused by the following:

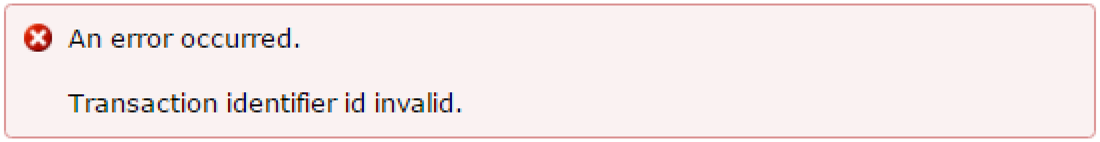
- The ProtectPay SignalR server is unavailable
- The client attempted to subscribe to the incorrect SignalR hub
- The client version of the SignalR.js file and/or the jQuery.js file is incorrect.
- The SignalR transport connection failed due to incorrect parameters
- Incorrect usage of the hpp.js file
- Incorrect usage of the SignalR.js file

The connection should be re-attempted once all the aforementioned possible causes have been investigated and resolved.

- ❖ The reference to contact customer service refers to the clients customer service representatives and not ProPay's.

Invalid HostedTransactionIdentifier Errors

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



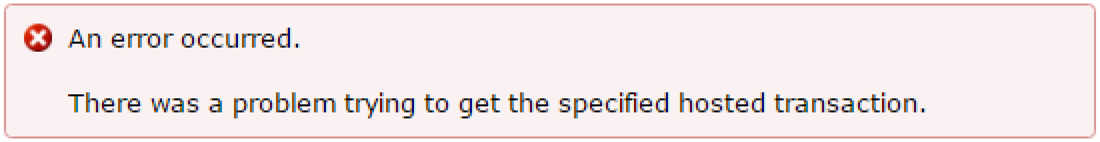
An error occurred.

Transaction identifier id invalid.

This will raise the **signalR_OnFormCommunicationTimeout()** event in the hpp.js based on the value set for '**TimeoutInterval**' in the hpp.js. The SignalR connection will be terminated.

Using an Already Used HostedTransactionIdentifier Error

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:



An error occurred.

There was a problem trying to get the specified hosted transaction.

This will raise the **signalR_OnFormSubmitErrord()** in the hpp.js. A new HostedTransactionIdentifier must be created using ProtectPay API method 4.7.2 Create HostedTransactionIdentifier.

3.2.1 Hosted Payment Page Configurations

The Hosted Payment Page configuration is set when creating the HostedTransactionIdentifier.

Store Payment Method Only

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.

Attribute	Value
StoreCard	True
ProcessCard	False
OnlyStoreCardOnSuccessfulProcess	False

- ❖ An minimum amount of 100 ($1.00) is required when creating a HostedTransactionIdentier and will be displayed to the payer, this can be hidden using the following custom CSS in a custom CSS file to prevent a payer from believing their payment method will be charged when setting the process and authorize values to false.

```
.FormAmount {display: none;}
```

For additional information about using a custom CSS file please review section 3.2.3 Customization of the Hosted Payment Page

Process Payment Method Only

In order to use the Hosted Payment Page to process a payment method only and not store it please set the following attributes when creating the HostedTransactionIdentifier. The Hosted Payment Page will attempt to process the card (Auth or Auth and Capture) and the result will return the result of the transaction only.

Attribute	Value
StoreCard	False
ProcessCard	True
OnlyStoreCardOnSuccessfulProcess	False

Process and Store Payment Method

In order to use the Hosted Payment Page to process a payment method and store the payment method information please set the following attributes when creating the HostedTransactionIdentifier. The Hosted Payment Page will attempt to process the payment method (Auth or Auth and Capture) and also store the payment method. The results will return both the results of the process attempt and the PaymentMethodId

Attribute	Value
StoreCard	True
ProcessCard	True
OnlyStoreCardOnSuccessfulProcess	False

Process and Store Payment Method only if the Payment Method Processed Successfully

In order to use the Hosted Payment Page to process a payment method and store the payment method information only if the payment method was successfully processed (Auth or Auth and Capture) please set the following attributes when creating the HostedTransactionIdentifier. The Hosted Payment Page will attempt to process the payment method (Auth or Auth and Capture) and only store the payment method if the process request was successful. The results will return both the results of the process attempt and the PaymentMethodId only if the process request was successful.

Attribute	Value
ProcessCard	True
OnlyStoreCardOnSuccessfulProcess	True

Custom Prefilling of Non-Sensitive Information

In order to provide an enhanced customer user experience, the Hosted Payment Page can load with non-sensitive payment information prefilled. This is useful in the following cases:

1. Merchant known payer wants to add an additional stored payment method to merchant system.
2. The Hosted Payment Page was unable to process the payment method.

The following fields can be prefilled into the Hosted Payment Page.

- Payer Name
- Payment Method Description
- Address 1
- Address 2
- City
- State
- Postal Code
- Country

In order to pre-fil the Payer Name field the **CardHolderNameRequirementType** must be set to 1 or 2 when creating the HostedTransactionIdentifier. In order to pre-fill the Address 1, Address 2, City, State, Postal Code and Country fields the **AvsRequirementType** must be set to 1 or 2 when creating the HostedTransactionIdentifier.

In the case of a payment method processing failure the ProtectPay API method 4.7.3 'Get Hosted Transaction Results' will return all the non-sensitive payment method information it was configured to collect. In this case the information returned can be used to prefill a new Hosted Payment Page. The following chart indicates which response attribute returned should be passed to the appropriate attribute in ProtectPay API method 4.7.2 'Create HostedTransactionIdentifier'.

Response – Create Attribute Correlation

Hosted Transaction Result Attribute	Create HostedTransactionIdentifier 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

3.2.2 Sample Development HTML Page

The HPP-Dev sample application is designed to be used by developers integrating the Hosted Payment Page into their software solutions.
To use this sample, put a Valid HostedTransactionIdentifier in the text input field, then press the start button.
In a production setting this value is best stored in the browser session as it is needed to get the results of the transaction once the Hosted Payment Page process is complete.

All three included Javascript files must load, so the solution should be loaded from a web server, rather than the file system directly.
To change the endpoint, update the 'baseURI' in the hpp.js file according to the notes in the hpp.js.
To change the element ID of the iFrame to load the Hosted Payment Page, update the 'iFrameId' variable.
To change the Timeout Value, update the 'TimeoutInterval' variable. Note that this is an integer and is expressed in milliseconds.
Additional Information about the timeout value can be found in section 3.2 of the Hosted Payment Page Manual version 4.x.x

There are several helper functions that are optional for creating and controlling the user experience.
These are particularly useful for development but may or may not be included in your production solution.
There are several underlying functions in the hpp.js file that should not be altered, except by advanced users who understand well how SignalR and the Hosted Payment Page work.
For an overview of how the Hosted Payment Page works, please read section 3.0 of the Hosted Payment Page Manual version 4.x.x.
If changes are made to the functions in the sample files where it is indicated they should not be changed, ProPay technical support may be

```
<html>
    <head>
        <title>ProPay - PMI</title>
        <!-- The defaultStyle.css will be replaced if you pass a custom CSS URL when creating the HostedTransactionIdentifier-->
        <link type="text/css" rel="stylesheet" href="CSS/defaultStyle.css">
        <style>
            html { height: 100%; min-height: 100%; }
            body { height: 100%; min-height: 100%; }
            .PageOverlay { filter: alpha(opacity=50); filter: progid:DXImageTransform.Microsoft.Alpha(opacity=50); -webkit-
opacity: 0.5; -moz-opacity: .50; -ms-opacity: 0.5; -o-opacity: 0.5; -khtml-opacity: 0.5; opacity: 0.5; background-color:#000000;
position:fixed; top:0; left:0; width:100%; height:100%; text-align:center; vertical-align:middle; z-index: 10000; }
            .PageMessageContainer { position:fixed; top:0; left:0; width:100%; height:100%; text-align:center; vertical-
align:middle; z-index: 10001; }
            .PageMessage { left: 50%; top: 50%; width: 50%; position: absolute; padding: 12px; margin-left: auto; margin-right:
auto; text-align: center; background-color: white; font-family: Arial,Verdana; font-size: 14px; border: 1px solid dimgrey; -
webkit-border-radius: 16px; -moz-border-radius: 16px; -ms-border-radius: 16px; -o-border-radius: 16px; -khtml-border-radius: 16px;
border-radius: 16px; -webkit-transform: translate(-50%, -50%); -moz-transform: translate(-50%, -50%); -ms-transform: translate(-
50%, -50%); -o-transform: translate(-50%, -50%); -khtml-transform: translate(-50%, -50%); transform: translate(-50%, -50%); }
        </style>
    </head>
    <body class="PageBody">
```

```
        <!-- BEGIN Error Overlay
             The following Error Overlay will display when there is an Error in Communication.
             The Styles CANNOT be modified.
             They are included For your Benefit to view When Styling
        -->
        <div class="PageOverlay" style="display:none;"></div>
        <div class="PageMessageContainer" style="display:none;" >
            <div class="PageMessage" style="display: none; padding: 16px 16px 16px 43px; border: 1px solid rgb(219, 148, 148);
background-image: url(&quot;images/Error.png&quot;); background-color: rgb(250, 242, 242); background-position: 16px 16px;
background-repeat: no-repeat;">A connection error has occurred. Please try your request again, or contact customer service.</div>
        </div>
        <!--END Error Overlay-->
        <div class="PageContent">
            <div class="PageHeader"></div>
            <div id="ValidationMessage" class="ErrorMessage">TEST ERROR</div>
            <div class="FormContainer">
                <form id="FormElement">
                    <fieldset class="FieldSetOrder FieldSet">
                        <legend class="LegendOrder Legend LegendIe">Order Information</legend>
                        <div class="FormAmount FormRow">
                            <span class="FormAmountLabel FormLeftColumn">
                            Amount
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormAmountValue FormRightColumn">
                                <span class="FormAmountField Label">$10.00 USD</span>
                            </span>
                        </div>
                        <div class="FormInvoice FormRow">
                            <span class="FormInvoiceLabel FormLeftColumn">
                            Invoice
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormInvoiceValue FormRightColumn">
                                <span class="FormInvoiceField Label">Invoice</span>
                            </span>
                        </div>
                        <div class="FormComment1 FormRow">
                            <span class="FormComment1Label FormLeftColumn">
                            Comment 1
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormComment1Value FormRightColumn">
                                <span class="FormComment1Field Label">Comment 1</span>
                            </span>
                        </div>
                    </fieldset>
                    <fieldset class="FieldsetCreditCard FieldSet">
                        <legend class="LegendCreditCard Legend LegendIe">Card Information</legend>
                        <div class="FormName FormRow">
                            <span class="FormNameLabel FormLeftColumn">
                                <span class="RequiredField">*</span>
                                Name
                                <span class="FormNameLabel1">(as it appears on card)</span>
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormNameValue FormRightColumn">
                                <input class="FormNameField TextBox" maxlength="50" />
                            </span>
                        </div>
                        <div class="FormCardNumber FormRow">
                            <span class="FormCardNumberLabel FormLeftColumn">
                                <span class="RequiredField">*</span>
                                Card Number
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormCardNumberValue FormRightColumn">
                                <input class="FormCardNumberField TextBox" maxlength="16" autocomplete="off" />
                            </span>
                        </div>
                        <div class="FormExpDate FormRow">
                            <span class="FormExpDateLabel FormLeftColumn">
                                <span class="RequiredField">*</span>
                                Expiration Date
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormExpDateValue FormRightColumn">
                                <select class="FormExpDateMonthField DropDown">

<option>Jan</option><option>Feb</option><option>Mar</option><option>Apr</option><option>May</option><option>Jun</option><option>Ju
l</option><option>Aug</option><option>Sep</option><option>Oct</option><option>Nov</option><option>Dec</option>
                                </select>
                                &nbsp;/&nbsp;
                                <select class="FormExpDateYearField DropDown">
```

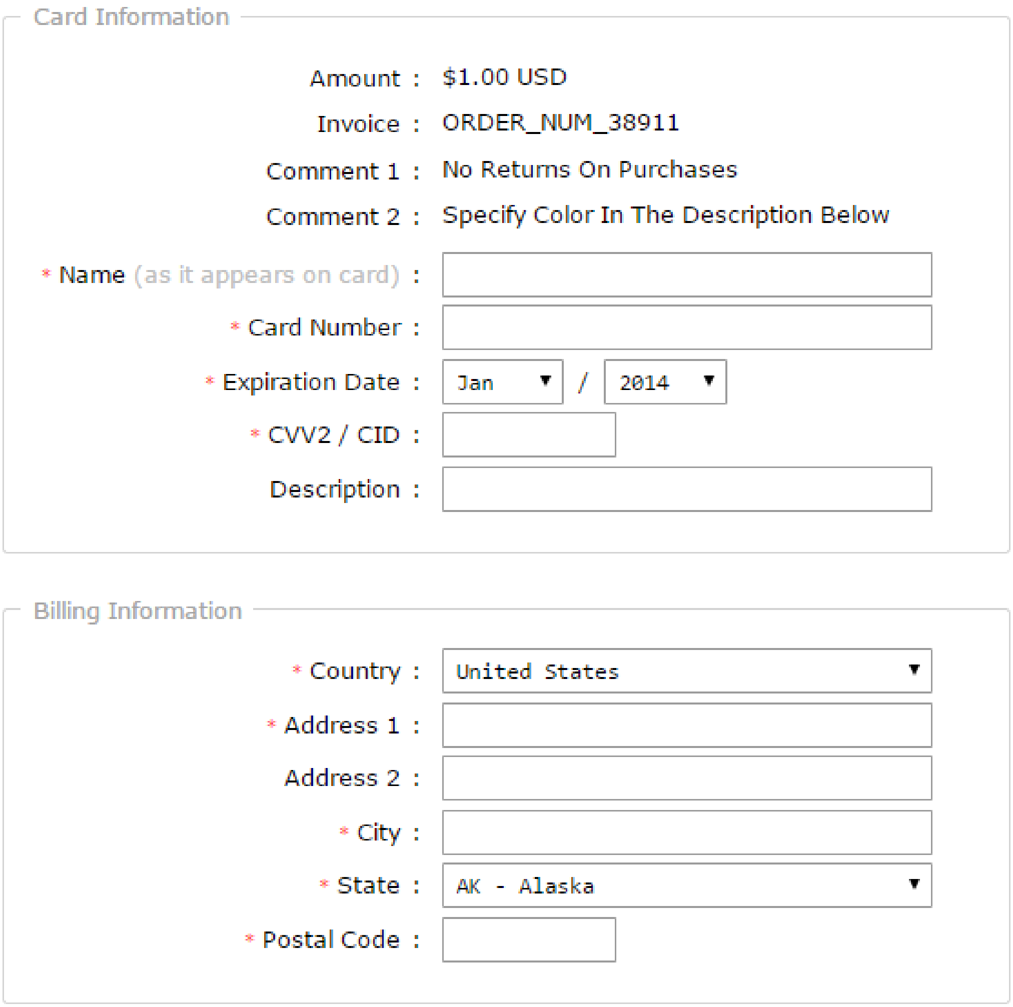
```
<option>2016</option><option>2017</option><option>2018</option><option>2019</option><option>2020</option><option>2021</option><opt
ion>2022</option><option>2023</option><option>2024</option><option>2025</option><option>2026</option>
                                </select>
                            </span>
                        </div>
                        <div class="FormCvv FormRow">
                            <span class="FormCvvLabel FormLeftColumn">
                                <span class="RequiredField">*</span>
                                CVV2 / CID
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormCvvValue FormRightColumn">
                                <input class="FormCvvField TextBox" maxlength="4" autocomplete="off" />
                            </span>
                        </div>
                        <div class="FormDescription FormRow">
                            <span class="FormDescriptionLabel FormLeftColumn">
                            Description
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormDescriptionValue FormRightColumn">
                                <input class="FormDescriptionField TextBox" maxlength="25" />
                            </span>
                        </div>
                    </fieldset>
                    <fieldset class="FieldsetCreditCard FieldSet">
                        <legend class="LegendCreditCard Legend LegendIe">Bank Account Information</legend>
                        <div class="FormName FormRow">
                            <span class="FormNameLabel FormLeftColumn">
                                    <span class="RequiredField">*</span>
                                Name <span class="FormNameLabel1">(as it appears on
                                     account
                                         )</span>
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormNameValue FormRightColumn">
                                <input class="FormNameField TextBox" data-val="true" data-val-length="The value specified in the
Name field is longer than the maximum length of 50 characters" data-val-length-max="50" id="txtName" maxlength="50" name="txtName"
required="True" type="text" value="Jack Sparrow"> <span class="field-validation-valid" data-valmsg-for="txtName" data-valmsg-
replace="true"></span>
                            </span>
                        </div>
                        <div class="FormBankAccountType FormRow">
                            <span class="FormBankAccountTypeLabel FormLeftColumn"><span class="RequiredField">*</span> Account Type <span class="LabelColon">:</span></span>
                            <span class="FormBankAccountValue FormRightColumn">
                                <label><input data-val="true" data-val-required="The BankAccountType field is required." id="Checking" name="BankAccountType" type="radio" value="Checking"> Checking </label>
                                <label><input id="Savings" name="BankAccountType" type="radio" value="Savings"> Savings </label>
                            </span>
                        </div>

                        <div class="FormBankRoutingNumber FormRow">
                            <span class="FormBankRoutingNumberLabel FormLeftColumn"><span class="RequiredField">*</span> Bank Routing Number <span class="LabelColon">:</span></span>
                            <span class="BankRoutingNumberNumberValue FormRightColumn">
                                <input autocomplete="off" class="FormBankRoutingNumberField TextBox AcceptOnlyNumbers" data-val="true" data-val-length="Routing Number must be 9 digits." data-val-length-max="9" data-val-length-min="9" data-val-regex="Please specify a valid Bank Routing Number." data-val-regex-pattern="^[0-9]+$" id="txtBankRoutingNumber" maxlength="9" name="txtBankRoutingNumber" required="True" type="text" value=""> <span class="field-validation-valid" data-valmsg-for="txtBankRoutingNumber" data-valmsg-replace="true"></span>
                            </span>
                        </div>
                        <div class="FormBankRoutingNumber FormRow">
                            <span class="FormBankRoutingNumberLabel FormLeftColumn"><span class="RequiredField">*</span> Confirm Bank Routing Number <span class="LabelColon">:</span></span>
                            <span class="BankRoutingNumberNumberValue FormRightColumn">
                                <input autocomplete="off" class="FormBankRoutingNumberField TextBox AcceptOnlyNumbers" data-val="true" data-val-equalto="Routing numbers do not match" data-val-equalto-other="*.txtBankRoutingNumber" data-val-length="Routing Number must be 9 digits." data-val-length-max="9" data-val-length-min="9" data-val-regex="Please specify a valid Bank Routing Number." data-val-regex-pattern="^[0-9]+$" id="txtConfirmBankRoutingNumber" maxlength="9" name="txtConfirmBankRoutingNumber" required="True" type="text" value=""> <span class="field-validation-valid" data-valmsg-for="txtConfirmBankRoutingNumber" data-valmsg-replace="true"></span>
                            </span>
                        </div>
                        <div class="FormBankAccountNumber FormRow">
                            <span class="FormBankAccountNumberLabel FormLeftColumn"><span class="RequiredField">*</span> Account Number <span class="LabelColon">:</span></span>
                            <span class="BankAccountNumberNumberValue FormRightColumn">
                                <input autocomplete="off" class="FormBankAccountNumberField TextBox AcceptOnlyNumbers" data-val="true" data-val-regex="Please specify a valid Bank Account Number." data-val-regex-pattern="^[0-9]+$" id="txtBankAccountNumber" name="txtBankAccountNumber" required="True" type="text" value=""> <span class="field-validation-valid" data-valmsg-for="txtBankAccountNumber" data-valmsg-replace="true"></span>
                            </span>
                        </div>
                        <div class="FormBankAccountNumber FormRow">
                            <span class="FormBankAccountNumberLabel FormLeftColumn"><span class="RequiredField">*</span> Confirm
Account Number <span class="LabelColon">:</span></span>
                            <span class="BankAccountNumberNumberValue FormRightColumn">
                                <input autocomplete="off" class="FormBankAccountNumberField TextBox AcceptOnlyNumbers" data-
val="true" data-val-equalto="Account Numbers do not match" data-val-equalto-other="*.txtBankAccountNumber" data-val-regex="Please
specify a valid Bank Account Number." data-val-regex-pattern="^[0-9]+$" id="txtConfirmBankAccountNumber"
name="txtConfirmBankAccountNumber" required="True" type="text" value=""> <span class="field-validation-valid" data-valmsg-
for="txtConfirmBankAccountNumber" data-valmsg-replace="true"></span>
                            </span>
                        </div>
                    </fieldset>
                    <fieldset class="FieldsetBillingInfo FieldSet">
                        <legend class="LegendBillingInfo Legend LegendIe">Billing Information</legend>
                        <div class="FormCountry FormRow">
                            <span class="FormCountryLabel FormLeftColumn">
                                <span class="RequiredField">*</span>
                                Country
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormCountryValue FormRightColumn">
                                <select class="FormCountryField DropDown">
                                    <option>United States</option><option>Canada</option><option>American
Samoa</option><option>Andorra</option><option>Argentina</option><option>Aruba</option><option>Australia</option><option>Austria</o
ption><option>Bahamas</option><option>Barbados</option><option>Belgium</option><option>Benin</option><option>Bermuda</option><opti
on>Bolivia</option><option>Brazil</option><option>British Indian Ocean
Territory</option><option>Bulgaria</option><option>Chile</option><option>China</option><option>Colombia</option><option>Costa
Rica</option><option>Croatia</option><option>Cyprus</option><option>Czech
Republic</option><option>Denmark</option><option>Dominican
Republic</option><option>Ecuador</option><option>Egypt</option><option>El
Salvador</option><option>Estonia</option><option>Fiji</option><option>Finland</option><option>France</option><option>French
Guiana</option><option>French
Polynesia</option><option>Germany</option><option>Ghana</option><option>Greece</option><option>Greenland</option><option>Guam</opt
ion><option>Guatemala</option><option>Holy See (Vatican City State)</option><option>Honduras</option><option>Hong
Kong</option><option>Hungary</option><option>Iceland</option><option>India</option><option>Indonesia</option><option>Ireland</opti
on><option>Israel</option><option>Italy</option><option>Japan</option><option>Korea, Republic
Of</option><option>Latvia</option><option>Liechtenstein</option><option>Luxembourg</option><option>Malaysia</option><option>Marsha
ll Islands</option><option>Martinique</option><option>Mexico</option><option>Netherlands</option><option>Netherlands
Antilles</option><option>New
Zealand</option><option>Nicaragua</option><option>Norway</option><option>Panama</option><option>Paraguay</option><option>Peru</opt
```

```
ion><option>Philippines</option><option>Pitcairn</option><option>Poland</option><option>Portugal</option><option>Puerto
Rico</option><option>Romania</option><option>Russia</option><option>Samoa</option><option>Saudi
Arabia</option><option>Singapore</option><option>Slovakia</option><option>Slovenia</option><option>Solomon
Islands</option><option>South Africa</option><option>Spain</option><option>Sri
Lanka</option><option>Swaziland</option><option>Sweden</option><option>Switzerland</option><option>Taiwan</option><option>Thailand
</option><option>Tonga</option><option>Trinidad And
Tobago</option><option>Turkey</option><option>Uganda</option><option>Ukraine</option><option>United Arab
Emirates</option><option>United Kingdom</option><option>Uruguay</option><option>Us Minor Outlying
Islands</option><option>Venezuela</option><option>Viet Nam</option><option>Virgin Islands, British</option><option>Virgin Islands,
U.S.</option>
                                </select>
                            </span>
                        </div>
                        <div class="FormAddress1 FormRow">
                            <span class="FormAddress1Label FormLeftColumn">
                                <span class="RequiredField">*</span>
                                Address 1
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormAddress1Value FormRightColumn">
                                <input class="FormAddress1Field TextBox" maxlength="50" />
                            </span>
                        </div>
                        <div class="FormAddress2 FormRow">
                            <span class="FormAddress2Label FormLeftColumn">
                            Address 2
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormAddress2Value FormRightColumn">
                                <input class="FormAddress2Field TextBox" maxlength="50" />
                            </span>
                        </div>
                        <div class="FormCity FormRow">
                            <span class="FormCityLabel FormLeftColumn">
                                <span class="RequiredField">*</span>
                                City
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormCityValue FormRightColumn">
                                <input class="FormCityField TextBox" maxlength="25" />
                            </span>
                        </div>
                        <div class="FormState FormRow">
                            <span class="FormStateLabel FormLeftColumn">
                                <span class="RequiredField">*</span>
                                State
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormStateValue FormRightColumn">
                                <select id="ddlState" name="ddlState" class="FormStateDropDown DropDown" style="display: inline-
block;">
                                    <option>AA - Armed Forces Americas</option><option>AE - Armed Forces Europe</option><option>AK
- Alaska</option><option>AL - Alabama</option><option>AP - Armed Forces Pacific</option><option>AR - Arkansas</option><option>AS -
American Samoa</option><option>AZ - Arizona</option><option>CA - California</option><option>CO - Colorado</option><option>CT -
Connecticut</option><option>DC - District of Columbia</option><option>DE - Delaware</option><option>FL -
Florida</option><option>GA - Georgia</option><option>GU - Guam</option><option>HI - Hawaii</option><option>IA -
Iowa</option><option>ID - Idaho</option><option>IL - Illinois</option><option>IN - Indiana</option><option>KS -
Kansas</option><option>KY - Kentucky</option><option>LA - Louisiana</option><option>MA - Massachusetts</option><option>MD -
Maryland</option><option>ME - Maine</option><option>MI - Michigan</option><option>MN - Minnesota</option><option>MO -
Missouri</option><option>MS - Mississippi</option><option>MT - Montana</option><option>NC - North Carolina</option><option>ND -
North Dakota</option><option>NE - Nebraska</option><option>NH - New Hampshire</option><option>NJ - New Jersey</option><option>NM -
New Mexico</option><option>NV - Nevada</option><option>NY - New York</option><option>OH - Ohio</option><option>OK -
Oklahoma</option><option>OR - Oregon</option><option>PA - Pennsylvania</option><option>PR - Puerto Rico</option><option>RI - Rhode
Island</option><option>SC - South Carolina</option><option>SD - South Dakota</option><option>TN - Tennessee</option><option>TX -
Texas</option><option>UT - Utah</option><option>VA - Virginia</option><option>VT - Vermont</option><option>WA -
Washington</option><option>WI - Wisconsin</option><option>WV - West Virginia</option><option>WY - Wyoming</option><option>VI -
Virgin Islands</option>
                                </select>
                                <input class="FormStateField TextBox" style="display:none;" maxlength="3" />
                            </span>
                        </div>
                        <div class="FormZip FormRow">
                            <span class="FormZipLabel FormLeftColumn">
                                <span class="RequiredField">*</span>
                                Postal Code
                                <span class="LabelColon">:</span>
                            </span>
                            <span class="FormZipValue FormRightColumn">
                                <input class="FormZipField TextBox" maxlength="10" />
                            </span>
                        </div>
                    </fieldset>
                </form>
            </div>
            <div class="PageFooter"></div>
        </div>
    </body>
</html>
```

3.2.3 Customization of the Hosted Payment Page

The Hosted Payment Page can be styled to match a merchant's checkout page. A custom CSS can be used to replace the default CSS when the Hosted Payment Page loads on a merchant's checkout page. ProPay offers a sample HTML file and the default CSS file to assist in styling the Hosted Payment Page. The custom CSS file must be publically accessible and declared when creating the HostedTransactionIdentifier.

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



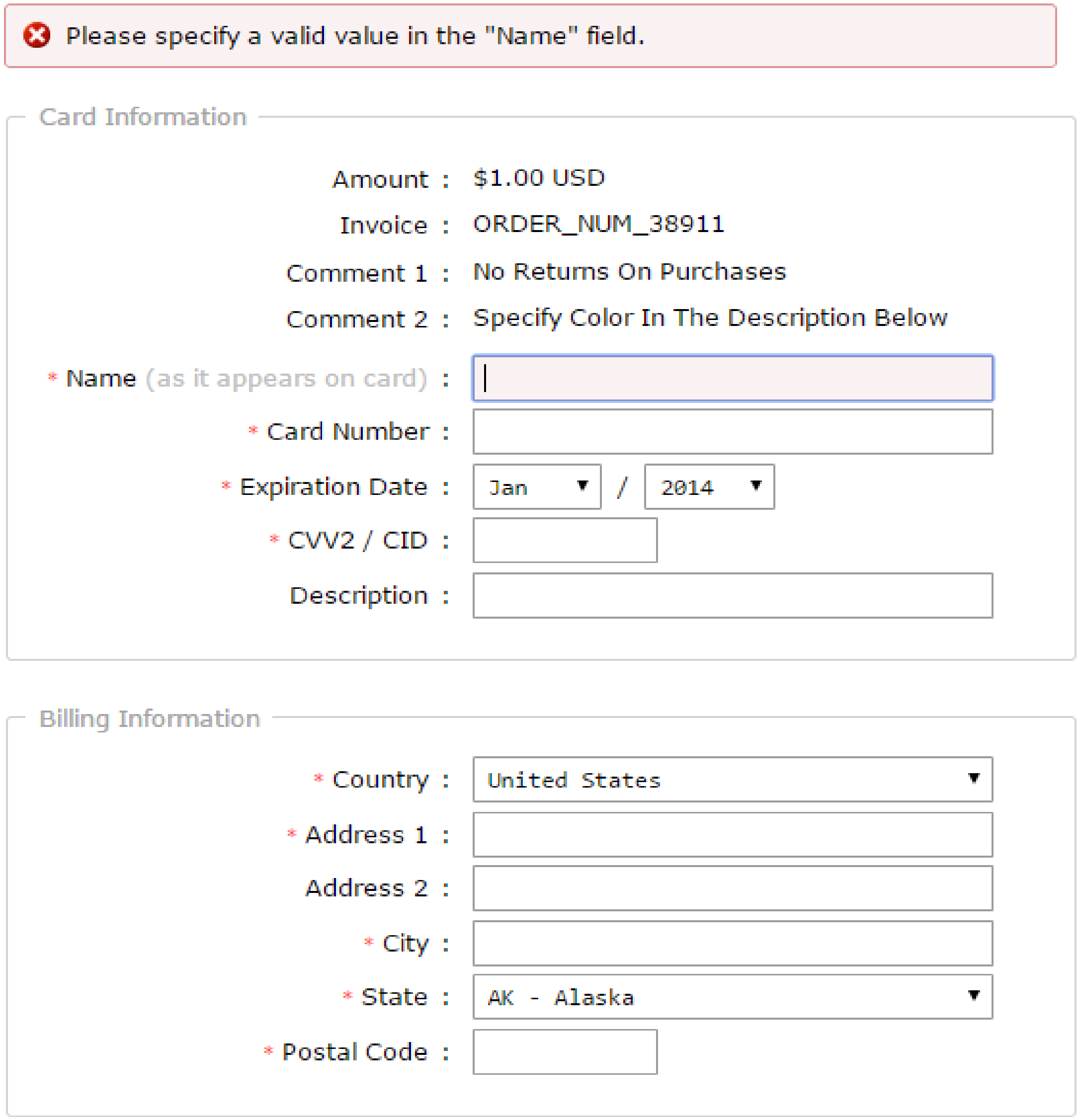
Card Information

Amount :	$1.00 USD
Invoice :	ORDER_NUM_38911
Comment 1 :	No Returns On Purchases
Comment 2 :	Specify Color In The Description Below
* Name (as it appears on card) :	
* Card Number :	
* Expiration Date :	Jan / 2014
* CVV2 / CID :	
Description :	

Billing Information

* Country :	United States
* Address 1 :	
Address 2 :	
* City :	
* State :	AK - Alaska
* Postal Code :	

Default appearance of HPP when failing form validation



Please specify a valid value in the "Name" field.

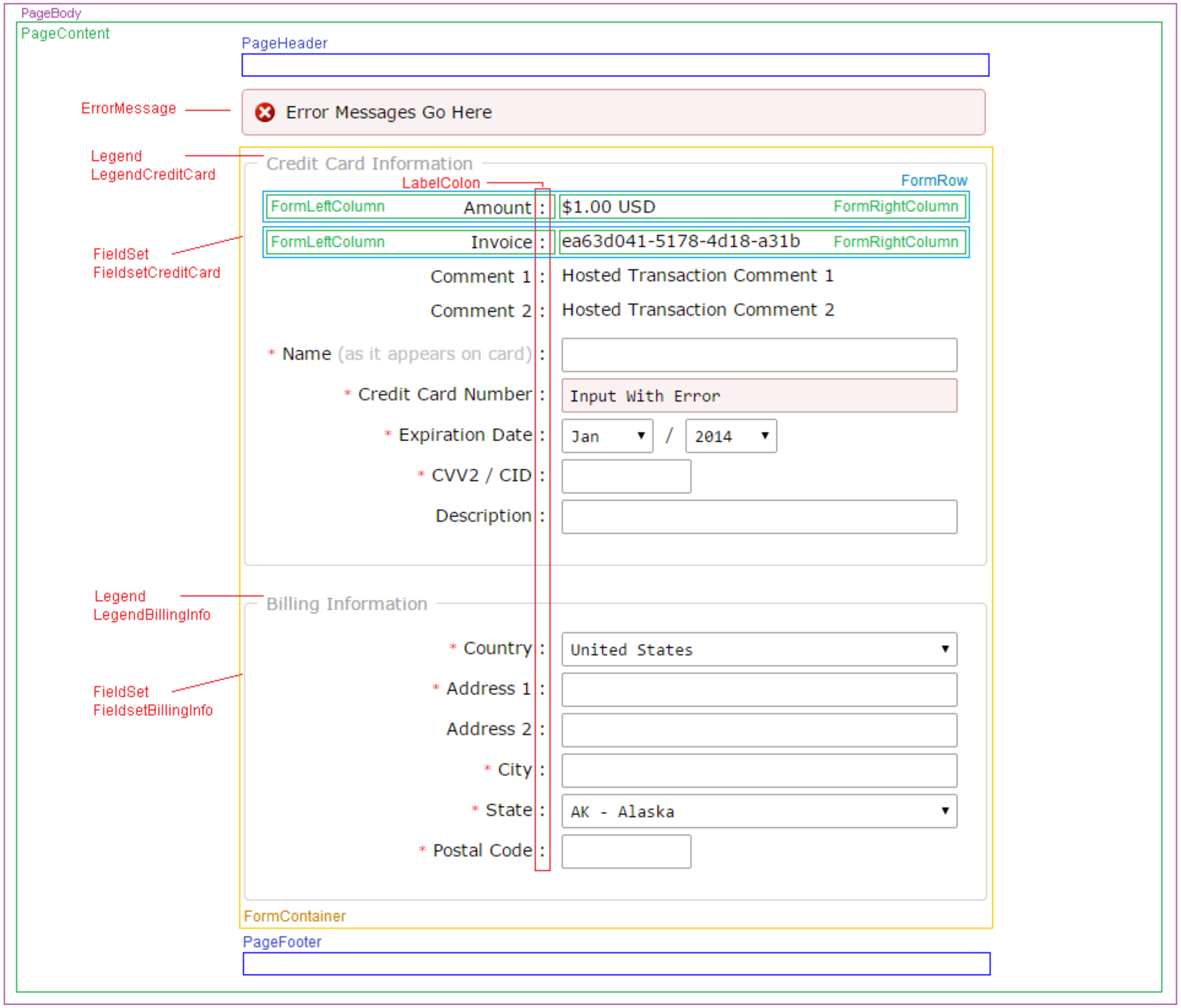
Card Information

Amount :	$1.00 USD
Invoice :	ORDER_NUM_38911
Comment 1 :	No Returns On Purchases
Comment 2 :	Specify Color In The Description Below
* Name (as it appears on card) :	
* Card Number :	
* Expiration Date :	Jan / 2014
* CVV2 / CID :	
Description :	

Billing Information

* Country :	United States
* Address 1 :	
Address 2 :	
* City :	
* State :	AK - Alaska
* Postal Code :	

HPP CSS Style Section Guide



Error Messages Go Here

Credit Card Information

Amount : $1.00 USD

Invoice : ea63d041-5178-4d18-a31b

Comment 1 : Hosted Transaction Comment 1

Comment 2 : Hosted Transaction Comment 2

* Name (as it appears on card) :

* Credit Card Number : Input With Error

* Expiration Date : Jan ▼ / 2014 ▼

* CVV2 / CID :

Description :

Billing Information

* Country : United States ▼

* Address 1 :

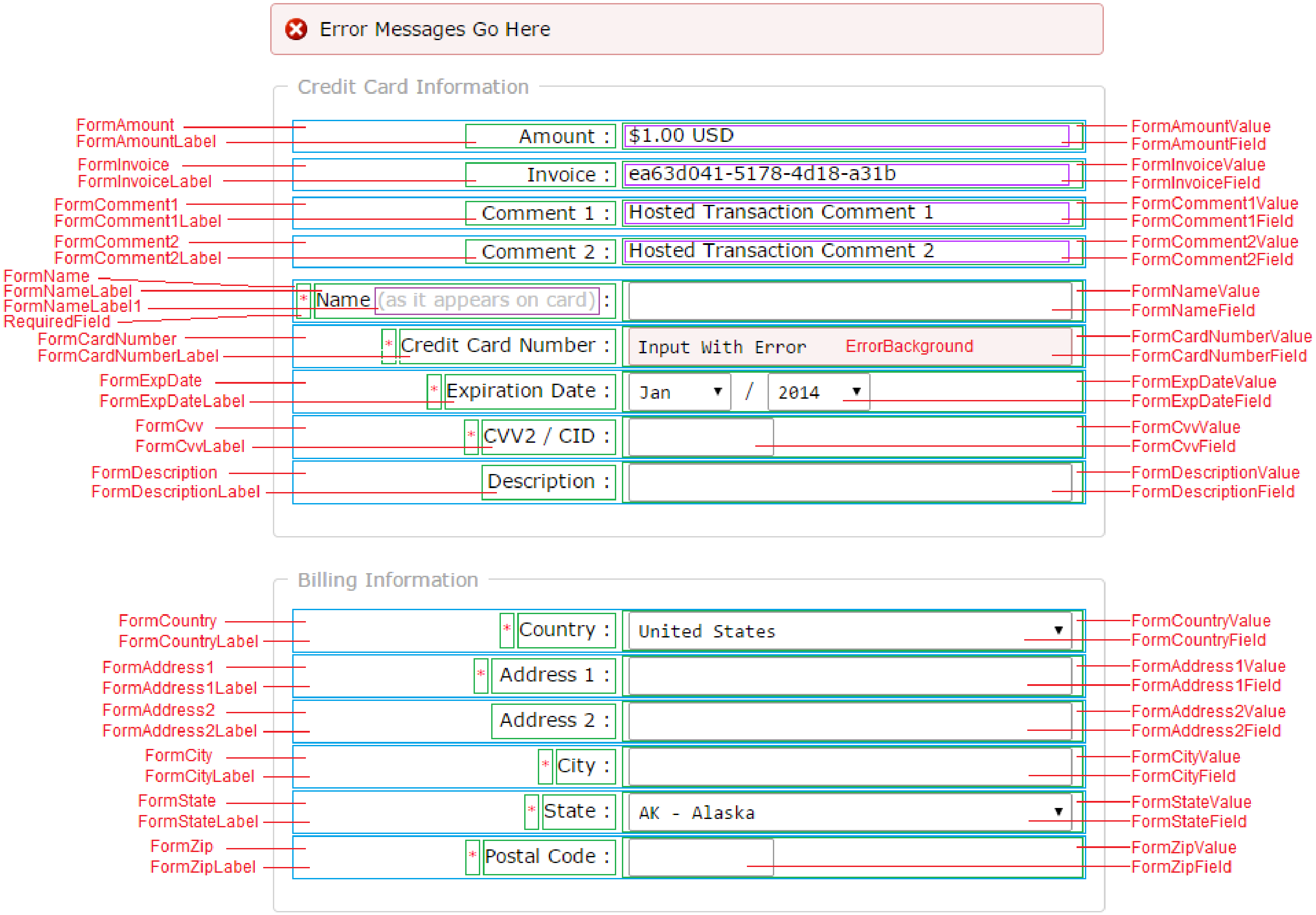
Address 2 :

* City :

* State : AK - Alaska ▼

* Postal Code :

HPP CSS Style Element Guide (Credit Card Transaction)



Error Messages Go Here

Credit Card Information

Amount : $1.00 USD

Invoice : ea63d041-5178-4d18-a31b

Comment 1 : Hosted Transaction Comment 1

Comment 2 : Hosted Transaction Comment 2

* Name (as it appears on card) :

* Credit Card Number : Input With Error ErrorBackground

* Expiration Date : Jan ▾ / 2014 ▾

* CVV2 / CID :

Description :

Billing Information

* Country : United States ▾

* Address 1 :

Address 2 :

* City :

* State : AK - Alaska ▾

* Postal Code :

- All labels on the right have the "Label" class.
- All text fields on the right have the "TextBox" class.
- All drop down list fields on the right have the "DropDown" class.

HPP CSS Style Element Guide (ACH Transactions)

The following CSS Classes have been added:

```
.FormBankAccountType{}
.FormBankAccountTypeLabel{}
.FormBankAccountValue{}
.FormRadioButtonField{}

.FormBankAccountNumber{}
.FormBankAccountNumberLabel{}
.BankAccountNumberNumberValue{}
.FormBankAccountNumberField{}

.FormBankRoutingNumber{}
.FormBankRoutingNumberLabel{}
.BankRoutingNumberNumberValue{}
.FormBankRoutingNumberField{}

.AcceptOnlyNumbers{}
```

[To Be Added]

defaultStyle.css File

The following CSS file is provided to assist in creating custom styles for the Hosted Payment Page when loaded into the iFrame on the merchant's checkout page. A custom CSS file URI must be declared when creating the HostedTransactionIdentifier and it must be publically accessible to ProtectPay

hppLayout.html File (Credit Card Transactions)

The following HTML page is provided to assist in the custom styling of the Hosted Payment Page when it is loaded into the iframe on the merchant checkout page.

```html
<html>
        <!--(c)2016-ProPay -->
        <head>
        <title>ProPay - PMI</title>
        <link type="text/css" rel="stylesheet" href="default.css">
    </head>
    <body class="PageBody">
                <div class="FormContainer">
                         <form>
                                 <fieldset class="FieldsetCreditCard FieldSet">
                    <legend class="LegendCreditCard Legend LegendIe">Card Information</legend>
                    <div class="FormAmount FormRow">
                        <span class="FormAmountLabel FormLeftColumn"> Amount <span class="LabelColon">:</span></span>
                        <span class="FormAmountValue FormRightColumn"><span class="FormAmountField Label">$10.00 USD</span></span>
                    </div>
                        <div class="FormInvoice FormRow">
                            <span class="FormInvoiceLabel FormLeftColumn"> Invoice <span class="LabelColon">:</span></span>
                            <span class="FormInvoiceValue FormRightColumn"><span class="FormInvoiceField Label">Test
Invoice</span></span>
                        </div>
                        <div class="FormComment1 FormRow">
                            <span class="FormComment1Label FormLeftColumn"> Comment 1 <span class="LabelColon">:</span></span>
                            <span class="FormComment1Value FormRightColumn"><span class="FormComment1Field Label">Test
Comment1</span></span>
                        </div>
                        <div class="FormComment2 FormRow">
                            <span class="FormComment2Label FormLeftColumn"> Comment 2 <span class="LabelColon">:</span></span>
                            <span class="FormComment2Value FormRightColumn"><span class="FormComment2Field Label">Test
Comment2</span></span>
                        </div>
                        <div class="FormName FormRow">
                            <span class="FormNameLabel FormLeftColumn">
                                <span class="RequiredField">*</span>
                                Name <span class="FormNameLabel1">(as it appears on card)</span>
                            <span class="LabelColon">:</span></span>
                            <span class="FormNameValue FormRightColumn"><input type="text" id="txtName" name="txtName"
class="FormNameField TextBox" maxlength="50" reqfield="True" value=""></span>
                        </div>
                    <div class="FormCardNumber FormRow">
                        <span class="FormCardNumberLabel FormLeftColumn"><span class="RequiredField">*</span> Card Number <span
class="LabelColon">:</span></span>
                        <span class="FormCardNumberValue FormRightColumn"><input type="text" id="txtCardNumber"
name="txtCardNumber" class="FormCardNumberField TextBox" maxlength="16" autocomplete="off" reqfield="True" value=""></span>
                    </div>
                    <div class="FormExpDate FormRow">
                        <span class="FormExpDateLabel FormLeftColumn"><span class="RequiredField">*</span> Expiration Date <span
class="LabelColon">:</span></span>
                        <span class="FormExpDateValue FormRightColumn">
                            <select id="ddlExpDateMonth" name="ddlExpDateMonth" class="FormExpDateMonthField DropDown"
reqfield="True">
                                    <option value="01">Jan</option>
                                    <option value="02">Feb</option>
                                    <option value="03">Mar</option>
                                    <option value="04">Apr</option>
                                    <option value="05">May</option>
                                    <option value="06">Jun</option>
                                    <option value="07">Jul</option>
                                    <option value="08">Aug</option>
                                    <option value="09">Sep</option>
                                    <option value="10">Oct</option>
                                    <option value="11">Nov</option>
                                    <option value="12">Dec</option>
                            </select>
                            &nbsp;/&nbsp;
                            <select id="ddlExpDateYear" name="ddlExpDateYear" class="FormExpDateYearField DropDown"
reqfield="True">
                                    <option value="2016">2016</option>
                                    <option value="2017">2017</option>
                                    <option value="2018">2018</option>
                                    <option value="2019">2019</option>
                                    <option value="2020">2020</option>
                                    <option value="2021">2021</option>
                                    <option value="2022">2022</option>
                                    <option value="2023">2023</option>
                                    <option value="2024">2024</option>
                                    <option value="2025">2025</option>
                                    <option value="2026">2026</option>
                            </select>
                        </span>
                    </div>
                    <div class="FormCvv FormRow">
                        <span class="FormCvvLabel FormLeftColumn">
                            <span class="RequiredField">*</span>
                            CVV2 / CID
                        <span class="LabelColon">:</span></span>
                        <span class="FormCvvValue FormRightColumn"><input type="text" id="txtCvv" name="txtCvv"
class="FormCvvField TextBox" maxlength="4" autocomplete="off" reqfield="True" value=""></span>
                    </div>
                    <div class="FormDescription FormRow">
                        <span class="FormDescriptionLabel FormLeftColumn">Description <span class="LabelColon">:</span></span>
                        <span class="FormDescriptionValue FormRightColumn"><input type="text" id="txtDescription"
name="txtDescription" class="FormDescriptionField TextBox" maxlength="25" reqfield="False" value=""></span>
                    </div>
                </fieldset>
                                 <fieldset class="FieldsetBillingInfo FieldSet">
                                         <legend class="LegendBillingInfo Legend LegendIe">Billing Information</legend>
                                         <div class="FormCountry FormRow">
                                                 <span class="FormCountryLabel FormLeftColumn">
                                                         <span class="RequiredField">*</span>
                                                         Country
                                                         <span class="LabelColon">:</span>
                                                 </span>
                                                 <span class="FormCountryValue FormRightColumn">
                                                         <select id="ddlCountry" name="ddlCountry" class="FormCountryField
DropDown" reqfield="True" onchange="ddlCountry_OnChange()" onclick="ddlCountry_OnChange()">
                                                                         <option value="USA" selected="selected">United
States</option>

                                                                         <option value="CAN">Canada</option>
                                                                         <option value="ASM">American Samoa</option>
                                                                         <option value="AND">Andorra</option>
                                                                         <option value="ARG">Argentina</option>
                                                                         <option value="ABW">Aruba</option>
                                                                         <option value="AUS">Australia</option>
                                                                         <option value="AUT">Austria</option>
                                                                         <option value="BHS">Bahamas</option>
                                                                         <option value="BRB">Barbados</option>
                                                                         <option value="BEL">Belgium</option>
                                                                         <option value="BEN">Benin</option>
                                                                         <option value="BMU">Bermuda</option>
                                                                         <option value="BOL">Bolivia</option>
                                                                         <option value="BRA">Brazil</option>
```

```
                                                                           <option value="IOT">British Indian Ocean
Territory</option>
                                                                           <option value="BGR">Bulgaria</option>
                                                                           <option value="CHL">Chile</option>
                                                                           <option value="CHN">China</option>
                                                                           <option value="COL">Colombia</option>
                                                                           <option value="CRI">Costa Rica</option>
                                                                           <option value="HRV">Croatia</option>
                                                                           <option value="CYP">Cyprus</option>
                                                                           <option value="CZE">Czech Republic</option>
                                                                           <option value="DNK">Denmark</option>
                                                                           <option value="DOM">Dominican Republic</option>
                                                                           <option value="ECU">Ecuador</option>
                                                                           <option value="EGY">Egypt</option>
                                                                           <option value="SLV">El Salvador</option>
                                                                           <option value="EST">Estonia</option>
                                                                           <option value="FJI">Fiji</option>
                                                                           <option value="FIN">Finland</option>
                                                                           <option value="FRA">France</option>
                                                                           <option value="GUF">French Guiana</option>
                                                                           <option value="PYF">French Polynesia</option>
                                                                           <option value="DEU">Germany</option>
                                                                           <option value="GHA">Ghana</option>
                                                                           <option value="GRC">Greece</option>
                                                                           <option value="GRL">Greenland</option>
                                                                           <option value="GUM">Guam</option>
                                                                           <option value="GTM">Guatemala</option>
                                                                           <option value="VAT">Holy See (Vatican City
State)</option>
                                                                           <option value="HND">Honduras</option>
                                                                           <option value="HKG">Hong Kong</option>
                                                                           <option value="HUN">Hungary</option>
                                                                           <option value="ISL">Iceland</option>
                                                                           <option value="IND">India</option>
                                                                           <option value="IDN">Indonesia</option>
                                                                           <option value="IRL">Ireland</option>
                                                                           <option value="ISR">Israel</option>
                                                                           <option value="ITA">Italy</option>
                                                                           <option value="JPN">Japan</option>
                                                                           <option value="KOR">Korea, Republic Of</option>
                                                                           <option value="LVA">Latvia</option>
                                                                           <option value="LIE">Liechtenstein</option>
```

```
<option value="LUX">Luxembourg</option>
<option value="MYS">Malaysia</option>
<option value="MHL">Marshall Islands</option>
<option value="MTQ">Martinique</option>
<option value="MEX">Mexico</option>
<option value="NLD">Netherlands</option>
<option value="ANT">Netherlands Antilles</option>
<option value="NZL">New Zealand</option>
<option value="NIC">Nicaragua</option>
<option value="NOR">Norway</option>
<option value="PAN">Panama</option>
<option value="PRY">Paraguay</option>
<option value="PER">Peru</option>
<option value="PHL">Philippines</option>
<option value="PCN">Pitcairn</option>
<option value="POL">Poland</option>
<option value="PRT">Portugal</option>
<option value="PRI">Puerto Rico</option>
<option value="ROU">Romania</option>
<option value="RUS">Russia</option>
<option value="WSM">Samoa</option>
<option value="SAU">Saudi Arabia</option>
<option value="SGP">Singapore</option>
<option value="SVK">Slovakia</option>
<option value="SVN">Slovenia</option>
<option value="SLB">Solomon Islands</option>
<option value="ZAF">South Africa</option>
<option value="ESP">Spain</option>
<option value="LKA">Sri Lanka</option>
<option value="SWZ">Swaziland</option>
<option value="SWE">Sweden</option>
<option value="CHE">Switzerland</option>
<option value="TWN">Taiwan</option>
<option value="THA">Thailand</option>
<option value="TON">Tonga</option>
<option value="TTO">Trinidad And Tobago</option>
<option value="TUR">Turkey</option>
<option value="UGA">Uganda</option>
<option value="UKR">Ukraine</option>
<option value="ARE">United Arab Emirates</option>
<option value="GBR">United Kingdom</option>
<option value="URY">Uruguay</option>
```

```
                                                                        <option value="UMI">Us Minor Outlying Islands</option>
                                                                        <option value="VEN">Venezuela</option>
                                                                        <option value="VNM">Viet Nam</option>
                                                                        <option value="VGB">Virgin Islands, British</option>
                                                                        <option value="VIR">Virgin Islands, U.S.</option>
                                                        </select>
                                                </span>
                                        </div>
                                        <div class="FormAddress1 FormRow">
                                                <span class="FormAddress1Label FormLeftColumn">
                                                        <span class="RequiredField">*</span>
                                                        Address 1
                                                        <span class="LabelColon">:</span>
                                                </span>
                                                <span class="FormAddress1Value FormRightColumn">
                                                        <input type="text" id="txtAddress1" name="txtAddress1" 
class="FormAddress1Field TextBox" maxlength="50" reqfield="True" value="">
                                                </span>
                                        </div>
                                        <div class="FormAddress2 FormRow">
                                                <span class="FormAddress2Label FormLeftColumn">
                                                        Address 2
                                                        <span class="LabelColon">:</span>
                                                </span>
                                                <span class="FormAddress2Value FormRightColumn">
                                                        <input type="text" id="txtAddress2" name="txtAddress2" 
class="FormAddress2Field TextBox" maxlength="50" reqfield="False" value="">
                                                </span>
                                        </div>
                                        <div class="FormCity FormRow">
                                                <span class="FormCityLabel FormLeftColumn">
                                                        <span class="RequiredField">*</span>
                                                        City
                                                        <span class="LabelColon">:</span>
                                                </span>
                                                <span class="FormCityValue FormRightColumn">
                                                        <input type="text" id="txtCity" name="txtCity" class="FormCityField 
TextBox" maxlength="25" reqfield="True" value="">
                                                </span>
                                        </div>
                                        <div class="FormState FormRow">
                                                <span class="FormStateLabel FormLeftColumn">
```

```
                                                          <span class="RequiredField">*</span>
                                                          State
                                                          <span class="LabelColon">:</span>
                                                  </span>
                                                  <span class="FormStateValue FormRightColumn">
                                                          <select id="ddlState" name="ddlState" class="FormStateDropDown DropDown"
reqfield="True" style="display: inline-block;">
                                                                  <option value="AA">AA - Armed Forces Americas</option>
                                                                  <option value="AE">AE - Armed Forces Europe</option>
                                                                  <option value="AK">AK - Alaska</option>
                                                                  <option value="AL">AL - Alabama</option>
                                                                  <option value="AP">AP - Armed Forces Pacific</option>
                                                                  <option value="AR">AR - Arkansas</option>
                                                                  <option value="AS">AS - American Samoa</option>
                                                                  <option value="AZ">AZ - Arizona</option>
                                                                  <option value="CA">CA - California</option>
                                                                  <option value="CO">CO - Colorado</option>
                                                                  <option value="CT">CT - Connecticut</option>
                                                                  <option value="DC">DC - District of Columbia</option>
                                                                  <option value="DE">DE - Delaware</option>
                                                                  <option value="FL">FL - Florida</option>
                                                                  <option value="GA">GA - Georgia</option>
                                                                  <option value="GU">GU - Guam</option>
                                                                  <option value="HI">HI - Hawaii</option>
                                                                  <option value="IA">IA - Iowa</option>
                                                                  <option value="ID">ID - Idaho</option>
                                                                  <option value="IL">IL - Illinois</option>
                                                                  <option value="IN">IN - Indiana</option>
                                                                  <option value="KS">KS - Kansas</option>
                                                                  <option value="KY">KY - Kentucky</option>
                                                                  <option value="LA">LA - Louisiana</option>
                                                                  <option value="MA">MA - Massachusetts</option>
                                                                  <option value="MD">MD - Maryland</option>
                                                                  <option value="ME">ME - Maine</option>
                                                                  <option value="MI">MI - Michigan</option>
                                                                  <option value="MN">MN - Minnesota</option>
                                                                  <option value="MO">MO - Missouri</option>
                                                                  <option value="MS">MS - Mississippi</option>
                                                                  <option value="MT">MT - Montana</option>
                                                                  <option value="NC">NC - North Carolina</option>
                                                                  <option value="ND">ND - North Dakota</option>
                                                                  <option value="NE">NE - Nebraska</option>
```

```
                                <option value="NH">NH - New Hampshire</option>
                                <option value="NJ">NJ - New Jersey</option>
                                <option value="NM">NM - New Mexico</option>
                                <option value="NV">NV - Nevada</option>
                                <option value="NY">NY - New York</option>
                                <option value="OH">OH - Ohio</option>
                                <option value="OK">OK - Oklahoma</option>
                                <option value="OR">OR - Oregon</option>
                                <option value="PA">PA - Pennsylvania</option>
                                <option value="PR">PR - Puerto Rico</option>
                                <option value="RI">RI - Rhode Island</option>
                                <option value="SC">SC - South Carolina</option>
                                <option value="SD">SD - South Dakota</option>
                                <option value="TN">TN - Tennessee</option>
                                <option value="TX">TX - Texas</option>
                                <option value="UT">UT - Utah</option>
                                <option value="VA">VA - Virginia</option>
                                <option value="VT">VT - Vermont</option>
                                <option value="WA">WA - Washington</option>
                                <option value="WI">WI - Wisconsin</option>
                                <option value="WV">WV - West Virginia</option>
                                <option value="WY">WY - Wyoming</option>
                                <option value="VI">VI - Virgin Islands</option>
                            </select>
                            <input type="text" id="txtState" name="txtState" class="FormStateField
TextBox" style="display:none;" maxlength="3" reqfield="True" value="">
                        </span>
                    </div>
                    <div class="FormZip FormRow">
                        <span class="FormZipLabel FormLeftColumn">
                            <span class="RequiredField">*</span>
                            Postal Code
                            <span class="LabelColon">:</span>
                        </span>
                        <span class="FormZipValue FormRightColumn"><input type="text" id="txtZip"
name="txtZip" class="FormZipField TextBox" maxlength="10" reqfield="True" value=""></span>
                    </div>
                </fieldset>
            </form>
            <div class="PageFooter"></div>
    </div>
    </body>
```

```
</html>
```

hppLayout.html File (ACH Transactions)

[To Be Added]

3.2.4 ProPay hpp.js

ProPay provides the hpp.js JavaScript file in order to facilitate the usage of the Hosted Payment Page. This file contains all the methods to connect a merchant checkout page to the ProPay SignalR server and handle the events raised by the SignalR server connection and Hosted Payment Page. A developer experienced with Microsoft SignalR can develop his or her own scripting file using the hpp.js as an example.

Merchant Checkout Page Required Functions

The following functions are made available to utilize the Hosted Payment Page and must be declared and/or called to make use of the hpp.js file.

Merchant Checkout Page Callable functions

The following functions are to be called from the Merchant Checkout Page in order to load the Hosted Payment Page and submit it to ProtectPay

Function	Notes
HPP_Load(hostedTransactionIdentifier, enableConsoleLogging**)**;	Establishes the SignalR connection and loads the iframe with the Hosted Payment Page
signalR_SubmitForm();	Invokes the Hosted Payment Page ‘submit’ function and submits HPP to ProtectPay for processing

Merchant Checkout Page Invoked Functions

The following functions are invoked for specified reasons to assist with development and debugging as well as to allow a developer to create an optimal user experience. They must be declared on the merchant checkout page unless removed by the developer in the hpp.js Not all functions are required, they are simply made available.

Function	Raised By	Notes
formIsReadyToSubmit();	signalR_OnPing();	Raised when the merchant checkout page receives the “Ping” message from the Hosted Page through the SignalR server connection after being loaded in the iframe; The merchant checkout page should now enable the “submit” button on this event.

hpp.js Customization

The hpp.js can be customized for an optimal user experience. The following variables and/or functions can be modified to suit business needs and logic.

Client Variables

Event	Default	Notes
TimeoutInterval	10000	In (ms); This value should be increased depending on connectivity and number of browser session and not to exceed 30000ms
iFrameId	hppFrame	This is the HTML element ID of the iFrame to display the Hosted Payment Page
baseURI	https://protectpaytest.propay.com	This is the environment to point the Hosted Payment Page to

Customizable Functions

Event	Raised By	Notes
fixMissingBrowserFunctionality(){}	Hpp_Load();	This function adds missing browser functionality. Default it adds a version of String.Trim() for browsers that do not have a definition of it. Use this function to add additional missing browser functionality for browsers intended to be supported.
window_OnError(){}	JavaScript Errors	Use this function to handle scripting errors

SignalR Asynchronous Events Declared

The following events are declared and can further be used to call custom functions when an event is raised. If a specific event will not be handled specially the function should have the echoMessage() and/or throwError() functions removed or modified. Please do not modify the functions further as this may cause unintentional behavior.

SignalR Connection Asynchronous Events

Function	Event	Function Calls	Notes
signalR_OnStarting(e, data**){}**	Connection.starting()	echoMessage();	Raised one time when the connection starts
signalR_OnStateChanged(e, data**){}**	Connection.stateChanged()	echoMessage();	Raised each time the connection state of the connection changes
signalR_OnConnected(e, data**){}**	Connection.start.done()	echoMessage();	Raised one time when the connection enters a state of 'connected' prior to communication
signalR_OnConnectionFailed(e, data**){}**	Connection.start.failed()	**throwError();**	Raised one time when the connection fails to connect
signalR_OnError(e, data**){}**	Connection.Error()	**throwError();**	Raised any time the connection encounters and error
signalR_OnReceived(e, data**){}**	Connection.received()	echoMessage();	Raised any time the connection receives data from the server
signalR_OnConnectionSlow(e, data**){}**	Connection.connectionSlow()	echoMessage();	Raised any time the threshold for non-communication is met
signalR_OnDisconnect(e, data**){}**	Connection.disconnected()	echoMessage();	Raised one time with the connection is disconnected
signalR_OnReconnect(e, data**){}**	Connection.reconnected	echoMessage();	Raised anytime the connection state changes to 'Reconnecting' This occurs when transportConnectionDelay threshold has been met.
signalR_OnReconnecting(e, data**){}**	Connection.reconnecting()	echoMessage();	Raise anytime the connection state changes from 'Reconnecting' to 'Reconnected'. If the transportConnectionReconnectDelay threshold is met during reconnection the connection will terminate as lost

hpp.js Synchronous Events

The following functions are declared and called in a synchronous manner based on event raised by the SignalR server.

Synchronous Events

Function	Raised By	Function Calls	Notes
window.onerror()	JavaScript Error	**throwError();**	

signalR_OnEstablished(){}	Connection.Start();	echoMessage();
		throwError();
			Call back function when the SignalR Transport Connection has been completely established.

signalR_OnPing(e, data){}	Server 'Ping' Message	echoMessage();
		formIsReadyToSubmit();
			The Hosted Payment Page sends a 'ping' to the merchants checkout page once it is loaded into the iframe

signalR_OnFormSubmitSucceeded(e, data**){}**	Server 'formSubmitSucceeded' Message	echoMessage();
		formSubmitSucceeded();
			The Hosted Payment page sends a message to the clients page indicating the page was submitted successfully.

signalR_OnFormWasInvalid(e, data**){}**	Server 'formSubmitWasInvalid' Message	echoMessage();
		formWasInvalid();
			The Hosted Payment page sends a message to the clients page indicating the page contains input errors and failed validation

signalR_OnFormSubmitErrored(e, data**){}**	Server 'formSubmitErrored' Message	echoMessage();
		formEncounteredAnError();
			The Hosted Payment page sends a message to the clients page indicating the page was unable to be submitted

signalR_OnTimeout(){}	TimeoutTimerId.Tick	echoMessage();
		formCommunicationTimeout();
		throwError();
			The timeout period from the time the signalR connection is established to the reception of the 'ping' message from the Hosted Payment Page.

signalR_Connect(){}	hpp_Load();	echoMessage();	Called immediately when the SignalR transport connection first communicates with the ProtectPay SignalR Server
signalR_OnIFrameLoaded(){}	JavaScript Event iframe.onload()	echoMessage();	Will be called when the specified iframe completes loading
signalR_OnUnload(){}	JavaScript Event iframe.unload()	echoMessage();	Will be called whenever the specified iframe has been unloaded
signalR_OnDisconnect(){}	SignalR Connection Disconnected	echoMessage();	Will be called whenever the SignalR transport connection is completely disconnected.

hpp.js Dependent Functions

The following functions are declared and consumed by the functions contained in the hpp.js file.

Dependent Functions

Function	Called By	Notes

getStateName(state**){}**	signalR_OnStateChanged(e, data)
	signalR_OnConnected(e, data)
		Maps the integer value to a human readable version of the Transport Connection State

valueIsNumeric(value**){}**	signalR_OnPing(e, data)
	signalR_Disconnect()
		Checks that the supplied value is numeric

valueIsValid(value**){}**	signalR_OnConnectionFailed(e, data)
	signalR_OnError(e, data)
		Checks that the supplied value is not empty or null

hpp.js file

The hpp.js is provided to quicken development time making use of SignalR and JQuery and is included in plain text

```
/*================================================================================================================================
======
*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.
==================================================================================================================================
======*/

/*================================================================================================================================
======
This JavaScript file includes methods to initiate a Hosted Payment Page transaction through SignalR on your checkout page.

Required Dependencies
 -JQuery 1.6.4 or greater
 -Microsoft SignalR 2.0.0 or greater

*The following methods are defined in this file that should be called by your checkout page*

This method establishes the SignalR connection and loads the iFrame with the Hosted Payment Page
hpp_Load(hostedTransactionIdentifier, enableDeubgLogging)

This function should be called by your "submit" button to submit the Hosted Payment Page to ProtectPay(r)
signalR_SubmitForm()

*The following Global Variables are defined in this file and should be modified to meet your design needs*

var iFrameId - This is the ID of the HTML iFrame element to display the Hosted Payment Page
var baseURI - This is the base URI for ProtectPay
    *ProPay Integration Environment: https://protectpaytest.propay.com
    *ProPay Sandbox Environment: https://sbprotectpay.propay.com
    *ProPay Production Environment: https://protectpay.propay.com

var TimeoutInterval - This is the timeout value in ms for your checkout page to wait for the 'ping' message from the Hosted
Payment Page
```

```
*The following functions are defined in this file and should be modified to meet your design needs*

This custom function is used to fix missing browser functionality for browsers you intend to support
fixMissingBrowserFunctionality()

This function is invoked when the browser detects a scripting error
window_OnError(errorMsg, url, lineNumber, column, errorObj)

SignalR Connection Event Methods
  - signalR_OnStarting(e, data)
  - signalR_OnStateChanged(e, data)
  - signalR_OnConnected(e, data)
  - signalR_OnConnectionFailed(e, data)
  - signalR_OnConnectionSlow(e, data)
  - signalR_OnReconnecting(e, data)
  - signalR_OnReconnect(e, data)
  - signalR_OnDisconnect(e, data)

Hosted Payment Page Response Methods
  - signalR_OnFormWasInvalid()
  - signalR_OnFormSubmitErrored()
  - signalR_OnFormSubmitSucceeded()
  - signalR_OnFormCommunicationTimeout()

==================================================================================================================================
======*/

/*================================================================================================================================
======
*This file will invoke several methods to assist you in creating an optimal user experience the following Methods should be 
declared
 on the checkout page that references this file:

 //This function is invoked when the Hosted Payment Page and the Checkout Page are connected and the Hosted Payment Page is ready 
for submission
 function formIsReadyToSubmit() {
     //Do not allow the user to submit the Hosted Payment Page until this Method has been invoked
    document.getElementById('btnSubmit').disabled = false;
    document.getElementById('btnSubmit').className = 'SubmitButtonEnabled';
    document.getElementById('btnStart').disabled = true;
 }
```

```
==================================================================================================================================
======*/

var iFrameId = 'hppFrame';
var baseURI = 'https://protectpaytest.propay.com';
var TimeoutInterval = 30000; //in ms

function fixMissingBrowserFunctionality() {

    // If 'String.trim()' is not defined (like on IE8), define it here
    if (typeof String.prototype.trim !== 'function') {
        String.prototype.trim = function () {
            return this.replace(/^\s+|\s+$/g, '');
        }
    }

    //Add additional Checks for Browsers you intend to support
}

// Invoked after a JavaScript error occurs on the page.
function window_OnError(errorMsg, url, lineNumber, column, errorObj) {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: red">Error Thrown: </b> window_OnError() <b>Raised</b>');
        echoBrowserMessage('<b style="color: red"> &#149; Type: </b>' + errorObj);
        echoBrowserMessage('<b style="color: red"> &#149; Message: </b>' + errorMsg);
        echoBrowserMessage('<b style="color: red"> &#149; File: </b>' + url);
        echoBrowserMessage('<b style="color: red"> &#149; Line Number: </b>' + lineNumber);
        echoBrowserMessage('<b style="color: red"> &#149; Column: </b>' + column);
        echoBrowserMessage('<b style="color: red">Error Details: </b>');
        return true;
    }
}

// Invoked before anything is sent over the connection.
function signalR_OnStarting(e, data) {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: blue">Establishing SignalR Connection
Capability</span> signalR_OnStarting() <b>Raised</b>');
    }
}
```

```
// Invoked when the connection state changes.
function signalR_OnStateChanged(e, data) {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: blue">SignalR Connection State Change</span>
signalR_OnStateChanged() <b>Raised()</b>');
        //var oldState = getStateName(e.oldState);
        //var newState = getStateName(e.newState);
        echoSignalRMessage('<b style="color: Blue" >Changing state </b> from <b>' + getStateName(e.oldState) + '</b> to <b>' +
getStateName(e.newState) + '</b>');
    }
}

// Invoked when the 'start' method is called and succeeds in connecting to the server.
function signalR_OnConnected(e, data) {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: Blue ">SignalR Connection Established</span>
signalR_OnConnected() <b>Raised</b>');
        var msg = '<b style="color: Blue ">Connection Attributes</b>'
            // General Connection Stats
            + '<br> <b>&nbsp; &#149; Connection Protocol:</b> ' + e.protocol
            + '<br> <b>&nbsp; &#149; Connected Host:</b> ' + e.host
            + '<br> <b>&nbsp; &#149; Connected Host Resource URI:</b> ' + e.appRelativeUrl
            // SignalR Connection Stats
            + '<br> <b>&nbsp; &#149; SignalR Connection State:</b> ' + getStateName(e.state)
            + '<br> <b>&nbsp; &#149; SignalR Connection ID:</b> ' + e.id
            + '<br> <b>&nbsp; &#149; SignalR Connection Token:</b> ' + e.token
            + '<br> <b>&nbsp; &#149; SignalR Connection Timout:</b> ' + e.disconnectTimeout; + 'ms'
            + '<br> <b>&nbsp; &#149; SignalR Connection Timeout Threshold:</b> ' + ((parseInt(e.disconnectTimeout) * 2) / 3) +
'ms'
            // SignalR Transport Connection Stats
            + '<br> <b>&nbsp; &#149; Transport Connection Name:</b> ' + e.transport.name
            + '<br> <b>&nbsp; &#149; Transport Connection Reconnect Delay:</b> ' + e.reconnectDelay + 'ms'
            + '<br> <b>&nbsp; &#149; Transport Connection Reconnect Threshold:</b> ' + ((parseInt(e.reconnectDelay) * 2) / 3) +
'ms'
            + '<br> <b>&nbsp; &#149; Transport Connection Reconnect Delay:</b> ' + e.transport.reconnectDelay + 'ms'
            + '<br> <b>&nbsp; &#149; Transport Connection Reconnect Threshold:</b> ' + ((parseInt(e.transport.reconnectDelay) * 2)
/ 3) + 'ms'
            + '<br> <b>&nbsp; &#149; Logging SignalR Events to Console:</b> ' + e.logging;
        echoSignalRMessage(msg);
    }
}

// Invoked when the Connection.Start() method is called but fails to connect to the SignalR server.
function signalR_OnConnectionFailed(e, data) {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: red">Error Thrown: </b>Failed to connect to the server - Unknown,
signalR_OnConnectionFailed() <b>Raised</b>');
        if (valueIsValid(e) == true) {
            echoSignalRMessage('<b style="color: red">Error Thrown: </b>Failed to connect to the server: ' + e.message)
            echoSignalRMessage('<b style="color: red">Error Thrown: </b>Stack: ' + e.stack);
        }
        else {
            echoSignalRMessage('<b style="color: red">Error Thrown: </b>Failed to connect to the server - Unknown,
signalR_OnConnectionFailed() <b>Raised</b>');
        }
    }
}

// Invoked when the client detects a slow connection.
function signalR_OnConnectionSlow(e, data) {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: blue">Slow Connection Detected - </b>Keep Alive
Timout % Threshold Exceeded, signalR_OnConnectionSlow() <b>Raised</b>');
    }
}

// Invoked when the underlying transport begins reconnecting.
function signalR_OnReconnecting(e, data) {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: red">Connection Loss Threshold -
</span>Connection Lost OR Keep Alive Timout Exceede, signalR_OnReconnecting() <b>Raised</b>');
    }
}

// Invoked when the underlying transport reconnects.
function signalR_OnReconnect(e, data) {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: green">Connection Re-
established</span>signalR_OnReconnect() <b>Raised</b>');
    }
}

// Invoked when the client disconnects.
function signalR_OnDisconnect(e, data) {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: red">SignalR Connection
Disconnected</span>signalR_OnDisconnect() <b>Raised</b>');
    }
}

// This function is called when the server signalR issues a 'FormSubmitWasInvalid' message (indicating there was a problem
validating the form data).
function signalR_OnFormWasInvalid() {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: red">Form Submitted failed Validation -
</span>signalR_OnFormWasInvalid() <b>Raised</b>');
    }
}

// This function is called when the server signalR issues a 'FormSubmitErrored' message (indicating an error occurred when the
user submitted the form).
function signalR_OnFormSubmitErrored() {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: red">Form Submission Error -
</span>signalR_OnFormSubmitErrored() <b>Raised</b>');
    }
}

// This function is called when the server signalR issues a 'FormSubmitSucceeded' message (indicating the form was submitted
successfully).
function signalR_OnFormSubmitSucceeded() {
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: Green">Form Submission Succeeded -
</span>signalR_OnFormSubmitSucceeded() <b>Raised</b>');
```

```
    }
    //Use ProtectPay(r) API Method 4.7.3 'Get Hosted Transaction Results'
}

//This function is called when the timeout period elapses for communication from the Hosted Payment Page to your Checkout Page
function signalR_OnFormCommunicationTimeout(){
    //Custom Handler
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: Green">Form Submission Succeeded -
</span>signalR_OnFormCommunicationTimeout() <b>Raised</b>');
    }
 }

//=================================================================================================================
==============
// The following functions and variables must NOT be modified except by experienced developers. ProPay will not troubleshoot
changes made below
//=================================================================================================================
==============

var hppURI = baseURI + '/hpp/home/';
var signalrURI = baseURI + '/hpp/signalr';
var debug = false;

//Object Reference Variables
var Connection;
var signalR;
var TimeoutTimerId;

// This function is called to set up and initiate and validate the SignalR Connection, Load the Hosted Payment Page and verify
communication into the iFrame
function hpp_Load(hostedTransactionIdentifier, debugMode) {
    if(debugMode){
        debug = true;
    }

    // Add missing javascript functionality based on browser
    fixMissingBrowserFunctionality();

    // Wire up the scripting error handler
    window.onerror = window_OnError;
    // Wire up the window unload event
    window.onbeforeunload = signalR_OnUnload;

    // Set local variables
    hppURI = hppURI + hostedTransactionIdentifier;

    // Setup the connection to the server
    signalR_SetupConnection(hostedTransactionIdentifier);

    // Connect to the server
    signalR_Connect();
}

// This sets up the connection with the signalR on the server.
function signalR_SetupConnection(HID) {

    // Create a reference to the signalR
    Connection = $.hubConnection();
    Connection.url = signalrURI;
    Connection.qs = { 'hid': HID, 'c': '0' };

    if(debug){
        $.connection.fn.log = function (message) {
            echoSignalRConsoleMessage(message);
        }

    }

    // Get a reference to the signalR Proxy Connection
    signalR = Connection.createHubProxy('hostedTransaction');

    // Wire up all the SignalR events to local callback methods.
    Connection.starting(signalR_OnStarting);
    Connection.received(signalR_OnReceived);
    Connection.connectionSlow(signalR_OnConnectionSlow);
    Connection.reconnecting(signalR_OnReconnecting);
    Connection.reconnected(signalR_OnReconnect);
    Connection.stateChanged(signalR_OnStateChanged);
    Connection.disconnected(signalR_OnDisconnect);

    //Wire up all the SignalR Error Handler
    Connection.error(signalR_OnError);
    // Wire up the client-side function to receive calls from the server
    signalR.on('ping', signalR_OnPing);
    signalR.on('formSubmitSucceeded', signalR_OnFormSubmitSucceeded);
    signalR.on('formSubmitWasInvalid', signalR_OnFormWasInvalid);
    signalR.on('formSubmitErrored', signalR_OnFormSubmitErrored);
}

// This function starts the signalR and initialize the connection to the server signalR.
function signalR_Connect() {
    if (debug) {
        echoBrowserMessage('<b style="color: Green">Start: </b><span style="color: blue">Create SignalR Connection</span>');
    }
    Connection.start({}, signalR_OnEstablished)
        .done(signalR_OnConnected)
        .fail(signalR_OnConnectionFailed);
}

// Invoked after an error occurs with the connection.
function signalR_OnError(e, data) {
    if (e != null) {
        var msg = e.message;
        if (valueIsValid(e) == true) {
            if (String(e).indexOf('Access is denied') != -1 && navigator.userAgent.indexOf('MSIE 10.0') != -1) {
                msg = e + ' (If you are using IE 10, try pressing F12 and switch the "Browser Mode" or "Document Mode" to "IE9")';
            } else {
                msg = e;
            }
        }
        echoBrowserMessage('<b style="color: red">Error Thrown: </b>SignalR Connection Error Message: ' + msg);
        echoBrowserMessage('<b style="color: red">Error Thrown: </b>Stack: ' + e.stack);
    }
}

// Invoked when any data is received on the connection from the signalR server.
function signalR_OnReceived(e, data) {
    if (debug) {
        echoSignalRMessage('<span style="color: blue"><b>SignalR Data Received</b></span><br> <b>&nbsp; &#149; Hub:</b> ' + e.H +
'<br> <b>&nbsp; &#149; Method:</b> ' + e.M + '<br> <b>&nbsp; &#149; Callback Id:</b> ' + e.I);
    }
    //Unload the iFrame on successful submission
    if(e.M == 'formSubmitSucceeded'){
        document.getElementById(iFrameId).src = '';
        signalR_Disconnect();
    }
}

// Callback funrtion that is invoked on establishing a SignalR server connection this Loads the Hosted Payment Page into the
specified iFrame
function signalR_OnEstablished() {
    if (debug) {
        echoBrowserMessage('<b>Loading Hosted Payment Page</b></span><br> <b>&nbsp; &#149; Hosted Payment Page URL:</b> <u>' +
hppURI + '</u><br> <b>&nbsp; &#149; iFrame ID:</b> ' + iFrameId);
    }
    var iFrame = document.getElementById(iFrameId);
    if (iFrame) {
        iFrame.onload = signalR_OnIFrameLoaded;
        iFrame.src = hppURI;
    }
    else {
        if (debug) {
            echoBrowserMessage('<b style="color: red">Error Thrown: </b>Unable to locate an iFrame with element ID: ' + iFrameId);
        }
    }
}

// Invoked when the Hosted Payment Page in the specified iFrame has loaded.
// The Hosted Payment Page onLoad() Event sends a Ping Request through the SignalR Connection to this Page
// A Timeout is set for connection errors from the Hosted Payment Page to the SignalR Server
function signalR_OnIFrameLoaded() {
    if (debug) {
        echoBrowserMessage('<span style="color: blue"><b>Hosted Payment Page Form Loaded</b> - </span>Waiting for Ping Request
From Hosted Payment Page Through SignalR Server<br> <b>&nbsp; &#149</b>Setting Ping Request Timout Timer to:<b> ' +
TimeoutInterval + "ms</b>");
    }
    TimeoutTimerId = window.setTimeout(signalR_OnTimeout, TimeoutInterval);
}

// This function is invoked when the timer times out This indicates the Hosted Payment Page failed to send a Ping Request through
the SignalR Server Connection.
// This indicates a problem with the connection of the Hosted Payment Page to the SignalR Server
function signalR_OnTimeout() {
    if (debug) {
        echoBrowserMessage('<b style="color: red">Error Thrown: </b>Failed to Receive Ping Request after: <b>' + TimeoutInterval +
'ms</b> Raising <b>formCommunicationTimeout</b>');
    }
    signalR_OnFormCommunicationTimeout();
    signalR_Disconnect();
}

// This function is called when the SignalR server sends the 'Ping' message indicated the Hosted Payment Page is Connected to it.
function signalR_OnPing(e, data) {
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: Blue">SignalR Server Ping Recieved -
</span>Hosted Payment Page Connected to SignalR Server<br> <b>&nbsp; &#149</b>Sending Pong Response <br> <b>&nbsp;
&#149</b>Stopping Ping Request Timout Timer');

    }

    if (TimeoutTimerId && valueIsNumeric(TimeoutTimerId) == true) {
        window.clearTimeout(TimeoutTimerId);
        TimeoutTimerId = undefined;
    }

    signalR.invoke('pong');

    if(debug){
         echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: blue">Form Ready For Submission</span> Raising
<b>formIsReadyToSubmit()</b>');
    }

    formIsReadyToSubmit();
}

// This function sends a 'SubmitForm' message from the parent page to the server signalR... which relays a 'SubmitForm' message to
the child page.
function signalR_SubmitForm() {
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: blue">Submit Form Method Invoked -
</span>Sending Message to SignalRServer');
    }

    signalR.invoke('submitForm');
}
```

```
// Invoked when the page is unloaded to disconnect from the server.
function signalR_OnUnload() {
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: blue">Page Unloading');
    }

    signalR_Disconnect();
}

// This function stops the communication with the server signalR.
function signalR_Disconnect() {
    if (debug) {
        echoBrowserMessage('<b style="color: gold">Event: </b><span style="color: red">SignalR Connection Disconnecting</span>');
    }

    var iFrame = document.getElementById(iFrameId);
    if (iFrame) {
        iFrame.onload = null;
    }

    // If the Hosted Payment Page Communication Timer is active disable it.
    if (TimeoutTimerId && valueIsNumeric(TimeoutTimerId) == true) {
        window.clearTimeout(TimeoutTimerId);
        TimeoutTimerId = undefined;
    }

    // The 1st parameter of the 'stop' method is whether or not to asynchronously abort the connection.
    // The 2nd parameter of the 'stop' method is whether we want to notify the server that we are aborting the connection.
    Connection.stop(false, true);
}

// Gets a human readable string for the specified connection state ID
function getStateName(state) {
    switch (state) {
        case 0 : return 'Connecting';
        case 1 : return 'Connected';
        case 2 : return 'Reconnecting';
        case 4 : return 'Disconnected';
        default: return 'Unkown(' + state + ')';
    }
}
```

```
//This function checks if the value is Numberic
function valueIsNumeric(value) {
    return !isNaN(parseFloat(value)) && isFinite(value);
}

//The function checks if the value passed is not empty
function valueIsValid(value) {
    return (value && value != '' && value.trim != '');
}
```