Version 10.17.0

PROTECTPAY® PAYER MANAGEMENT INTERFACE: SEAMLESS PAYMENT INTERFACE (SPI)

Instructions to Interface with ProPay's 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 4.0.0 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
1/11/2017	4.17.1	Updates and Synchronizations with ProtectPay API Manual	Jared James Steven Barnett Tanner Olsen
2/07/2017	4.17.2	Enhanced to include Fraud Detection Elements	Jared James Steven Barnett Tanner Olsen Greg Maynes
03/14/2017	4.17.3	Updates and corrections Addition of additional error causes	Jared James Steven Barnett Tanner Olsen Greg Maynes
04/24/2017	4.17.4	Updates and corrections	Jared James Steven Barnett Tanner Olsen Greg Maynes

Contents

1.0 ProPay® ProtectPay Application Programming Interface

The ProtectPay Payer Management Interface: Seamless Payment Interface (SPI) 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 Seamless Payment Interface is based on an HTTP redirect to enable cross origin browser processes for a client system.

How to use this manual

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

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

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

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

Additional Resources

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

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.
- ProtectPay supports swipe transactions through integration of supported swipe devices.

Disclaimer

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

Updated manuals are available at www.propay.com/Resources.

1.1 Description of Seamless Payment Interface

The Seamless Payment Interface (SPI) 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 SPI enables a merchant to collect sensitive payment method information by redirecting a payer's browser to post the sensitive payment method information to a ProtectPay server for processing without having it traverse the client's system. This minimizes the merchants PCI compliance requirements and limits the risk and exposure of the merchant by not handling sensitive payment information, while allowing the customer to experience the payment process on the merchant's website.

Why the Seamless Payment Interface

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 perform a redirect in order to provide cross origin browser processes to provide a seamless payment experience to the payer.
The SPI is only needed when new payment method information must be collected. One of the SPI configuration options is 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.

SPI Processing Configurations

The SPI 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.

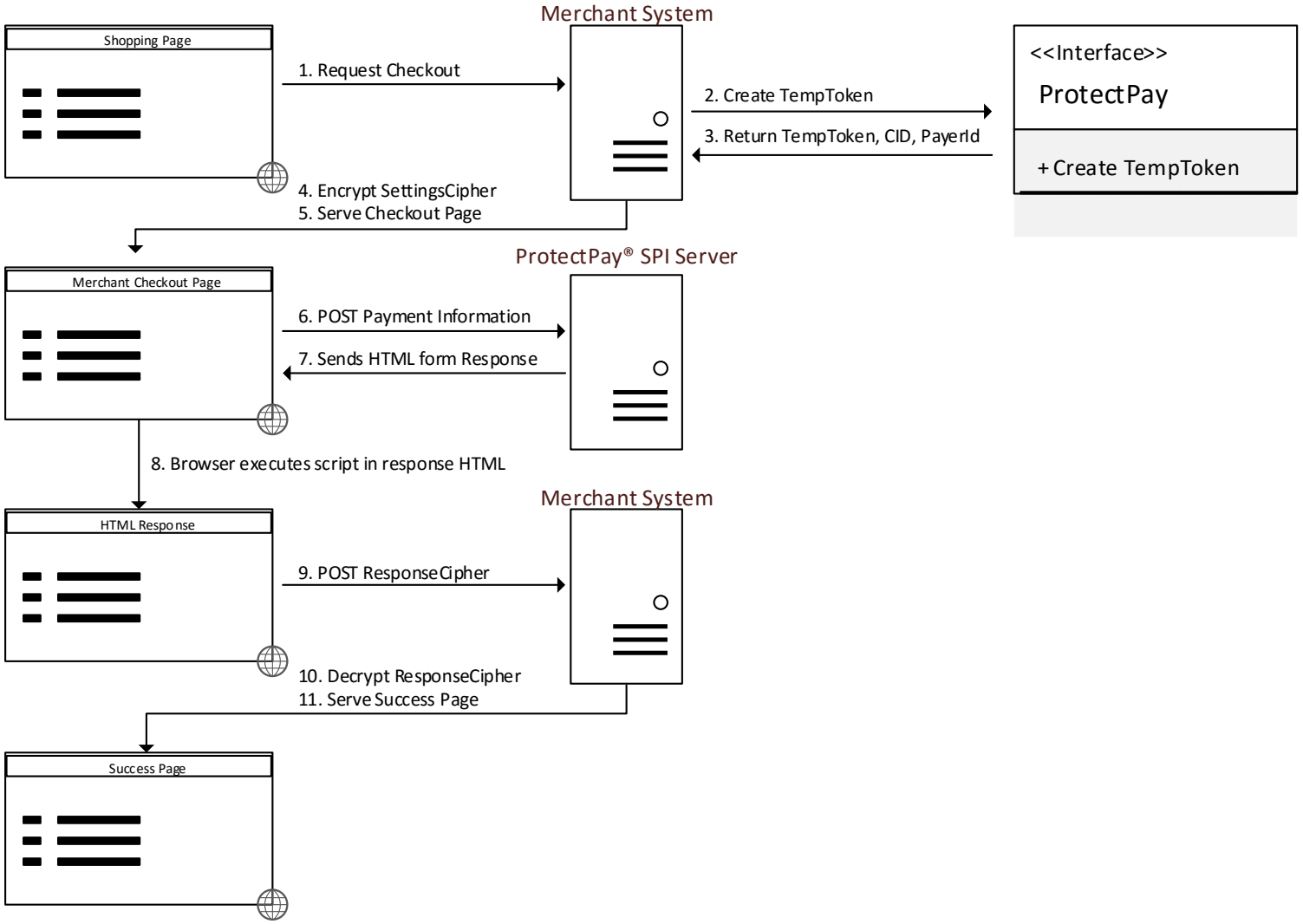
*See Section 3.2.2 Required Encrypted Parameters for configuration settings

1.2 Processing with the Seamless Payment Interface

Summary of Processing

1. Customer request checkout from the merchants system
2. The merchants system creates a TempToken using ProtectPay API Method 4.7.1 'Create TempToken'
3. The ProtectPay API returns the TempToken, Credential Id (CID) and PayerId.
4. The client system generates and encrypts a Key-Value Pair string called the 'SettingsCipher' that contains the merchant known information that cannot be altered by a payer. The SettingsCipher sets the processing parameters of the SPI.
   - ❖ See Section 3.2.2 'Required Encrypted Parameters'.
   - ❖ See Section 3.2.1 'Data Encryption Process'.
5. The merchants system serves a checkout page to the payer's browser including:
   a. Encrypted Key-Value Pair string (hidden on page).
   b. CredentialId (hidden on page).
   c. A form for a Payer to input payment information.
   - ❖ See Section 3.2.3 'API Post Elements'.
6. The payer fills in payment method information then clicks the 'submit' button executing client side scripting to form POST the information to the SPI.
   a. The SPI uses the CID to look up the TempToken used to encrypt the Key-Value Pair String and processes the payer information as instructed.
7. The SPI responds with a transitional HTML page that contains the response known as the 'ResponseCipher'.
8. The client's browser executes the script in the response transitional HTML
9. The client's browser Form POSTs the ResponseCipher to the Merchants System
10. The Merchants system decrypts the ResponseCipher using the same TempToken originally used to encrypt the Key-Value Pair string portion of the request.
    - ❖ See Section 3.2.5 'ResponseCipher Decrypted Values'.
11. The Merchants system serves the 'success' page

Process Flow



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 SPI POST URI: https://protectpaytest.propay.com/pmi/spr.aspx
   - ❖ 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 Preliminary ProtectPay API Methods:

- ProtectPay API Method 4.2.1 'Create PayerId' (PayerId may be created in same call as 'Create TempToken').
- ProtectPay API Method 4.7.1 'Create TempToken'.

ProtectPay API Authentication

ProPay will generate a unique Authentication Token which must be included in all API transactions. ProPay will assign you a unique BillerID which serves to represent the unique collections of MerchantProfileIds, PayerIds, and PaymentMethodIds that the BillerId will have access to and may create, edit and delete. The Authentication Token and BillerID are used to authenticate against the ProtectPay API and as the "user name" and "password". The Authentication Token and BillerId are identical for whichever interface is chosen to implement.

TempToken Security:

The Seamless Payment Interface uses a unique, one-time use working key known as a TempToken as part of the ProtectPay required API methods. This key is generated by setting a duration limit in seconds where it is valid. Once used, or once the time limit has expired, the token is no longer valid for use.

- ❖ Each TempToken must be kept confidential whether it is valid or not.

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.

Cross Origin Resource Sharing HTTP Headers

ProPay has added the following HTTP Headers to the response from the SPI prior to the redirection.

- Access-Control-Allow-Origin:*
- Access-Control-Allow-Methods:GET,POST,HEAD

This will allow a developer to receive the response from the SPI and perform checks against it before the browser is directed to the client-side results page.

- ❖ This should only be considered by more experienced developers

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 associated with the user and used in subsequent transaction requests instead of creating a new one for each transaction.

- The SPI is only required when creating a PaymentMethodId, or processing payment method information without wanting to store it. Once a PaymentMethodId is created, subsequent transactions should be processed using the ProtectPay API directly.

- Before form POSTing the payer-entered data to the SPI, the developer should validate the card number against a Mod 10 check using the LUHN algorithm, and should verify that the card number submitted conforms to rules established for the card type selected. The developer should also validate the expiration data is not past due and the CVV entered is the correct number of integers. This should be done before the cardholder submits the request to the SPI to avoid the customer waiting for an SPI response that indicates the card number, expiration date and CVV entered were incorrect. This will improve the end-user experience by not having to re-enter the information.

- 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.

- When using the SPI to only create a PaymentMethodId set the **StoreCard** option to: **always** and the process option to: none

- When using the SPI to process a payment method type and to store set the **StoreCard** option to: **OnSuccess** and the process option to:

  - ❖ See Section 3.2.6 for a sample checkout page that includes a JavaScript validation function

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 SPI parameter to store payment method.
   c. Set SPI 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 SPI parameter to store payment method.
   c. Set SPI 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 SPI Parameter to not store payment method.
    c. Set SPI parameter to process payment method.

3.2 Interface

3.2.1 Seamless Payment Interface Configurations

The Seamless Payment Interface is configured by the values set in the SettingsCipher.

Store Payment Method Only

In order to use the Seamless Payment Interface to store payment methods only please set the following attributes when creating the SettingsCipher. The Seamless Payment Interface will store the payment method information once stored the PaymentMethodId can be processed against directly using the ProtectPay API.

Attribute	Value
StoreCard	Always
ProcessMethod	None

Process Payment Method Only

In order to use the Seamless Payment Interface to process a payment method only and not store it please set the following attributes when creating the SettingsCipher. The Seamless Payment Interface 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	None
ProcessMethod	AuthOnly Capture

Process and Store Payment Method

In order to use the Seamless Payment Interface to process a payment method and store the payment method information if the payment method was successfully processed (Auth or Auth and Capture) please set the following attributes when creating the SettingsCipher. The Seamless Payment Interface 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
StoreCard	OnSuccess
ProcessMethod	AuthOnly Capture

3.2.2 Data Encryption Process

Create a key-value pair string that contains all of the data that cannot be altered by the cardholder. The required values that must be encrypted are known to the merchant without further input from a card holder and cannot be altered by the payer. The response from the SPI will be encrypted using the same information and verifies the response of the requested transaction type is from ProPay and not another entity.

- ❖ See section 3.2.2 Required Encrypted Parameters.

Encryption Process

Encrypt the Key-Value Pair using the following method:

1. UTF-8 encode the TempToken string and generate an MD5 hash of it.
2. UTF-8 encode the Key-Value Pair string and encrypt using AES-128 encryption using Cipher Block Chaining (CBC) mode.
    a. Set both the key and initialization vector (IV) equal to result from step 1.
3. Base64 encode the result of 2

This encrypted value is known as the 'SettingsCipher' and will be form POSTed to the SPI along with the cardholder information. The SPI will process the request and redirect the cardholder's browser to a response page set by the 'returnURL' parameter and form POST the response known as the 'ResponseCipher'.

Decryption Process

The 'ResponseCipher' is encrypted using the same process and TempToken used to encrypt the 'SettingsCipher'.

1. Base64 decode the response cipher.
2. UTF-8 encode the same TempToken used to encrypt and generate an MD5 hash of it.
3. Decrypt the result of step 1 using AES-128 decryption using Cipher Block Chaining (CBC) mode.
    a. Set both the Key and Initialization Vector (IV) equal to result from step 2.

- ❖ The decrypted response will be in the form of Key-Value Pairs and contain the response of the requested transaction.

Message Padding

AES 128 Encryption using Cipher Block Chaining requires the size of the message must be a multiple of the cipher block size. In this instance the block size is the same size as the MD5 Hash of the TempToken which is 16 bytes. Due to the variable nature of the Key-Value Pair that is to be encrypted, padding may need to be added in order to ensure the resulting message to be encrypted is a multiple of 16 bytes. If the string is padded in order to be encrypted the decrypted response will need to have any added padding removed before being converted back to a readable string.

3.2.3 Required Encrypted Parameters

The following table displays the required parameters and configuration options that must be encrypted and set to the value of the 'SettingsCipher' parameter when posting to ProPay. These should be generated and placed in a hidden field on the browser client page where the payer inputs the sensitive payment method information.

- The purpose of encrypting this data is to ensure the response the redirected page receives comes from ProPay and not another entity.

Using Fraud Detection

The Seamless Payment Interface has been can utilize the current Fraud Detection Provider elements supported by the ProtectPay Interface. These values are optional only if the fact a client chooses not to use them.

If being used each Fraud Provider has a unique set of required and optional variables that can be reviewed in ProtectPay API Appendix C Fraud Detection under the respective provider.

- **Each browser has specific limitations on the number of characters allowed in query-string submission. As such it is highly recommended if using Fraud Detection with the SPI a developer uses a Form POST rather than a query string.**

Required Encrypted Parameters

Parameter	Type	Max	Notes
AuthToken	string	-	Returned by 'Get a Temp Token' API call as "TempToken"
PayerID	long	-	Returned by ProtectPay API Method 'Create PayerId' as "ExternalAccountId" Returned by ProtectPay API Method 'Create TempToken' as "PayerId"
PaymentProcessType	String	-	Valid values are: ▪ ACH ▪ CreditCard
ProcessMethod	String	-	Valid values are: ▪ AuthOnly ▪ Capture ▪ None
PaymentMethodStorageOption	String	-	Valid values are: ▪ Always ▪ OnSuccess ▪ None
CurrencyCode	String	3	ISO 4217 standard 3 character currency code
Amount	long	-	The value representing the amount the for which the transaction should be processed *This amount must include a decimal point followed by two digits
StandardEntryClassCode	String	-	Standard Entry Class Code required for ACH Payment Processing Valid values are: ▪ PPD

			▪ CCD ▪ WEB ▪ TEL
InvoiceNumber	string	50	Recommended Transaction descriptor-only passed if supported by your gateway *ProPay rejects transactions as duplicate when the same card is charged for the same amount with the same invoice number, including blank invoices, in a 60 second period.
ReturnURL	string	-	Fully Qualified URL to direct client browser to redirect to when response is received *The URL cannot contain query string parameters, an error will be returned indicating an invalid SettingsCipher
ProfileId	long	-	Used to specify merchant account when your biller ID has access to multiple merchant accounts *If your account is set to point to multiples this value is required
Comment1	string	128	Transaction descriptor. Only passed if supported by your gateway *The following characters cannot be passed: "?"|"&"|" ="
Comment2	string	128	Transaction descriptor. Only passed if supported by your gateway *The following characters cannot be passed: "?"|"&"|" ="
echo	string	-	*Optional value that is not passed to gateway and is returned in the response *The following characters cannot be passed: "&"|"="
Protected	bool	-	True or False required if storing the payment method *Indicates whether a stored payment method can be deleted by the payer

Optional Fraud Parameters

SessionId	string		Session id for ThreatMetrix
InputIpAddress	string		InputIp Address for AmexEnhancedauth and ThreatMetrix
ShippingAddress1	string		Shipping Address1 for AmexEnhancedauth and ThreatMetrix
ShippingAddress2	string		Shipping Address2 for AmexEnhancedauth and ThreatMetrix
ShippingCity	string		Shipping City for AmexEnhancedauth and ThreatMetrix
ShippingState	string		Shipping State for AmexEnhancedauth and ThreatMetrix
ShippingZip	string		Shipping Zip for AmexEnhancedauth and ThreatMetrix
ShippingCountry	string		Shipping Country for AmexEnhancedauth and ThreatMetrix
ShippingFirstName	string		Shipping First Name for AmexEnhancedauth and ThreatMetrix
ShippingLastName	string		Shipping Last Name for AmexEnhancedauth and ThreatMetrix
ShippingPhoneNumber	string		Shipping Phone Number for AmexEnhancedauth and ThreatMetrix
ShippingMethod			Shipping Method for AmexEnhancedauth
CUA1	string		This is optional ThreatMetrix parameter CustomAttribute1
CUA2	string		This is optional ThreatMetrix parameter CustomAttribute2
CUA3	string		This is optional ThreatMetrix parameter CustomAttribute3
CUA4	string		This is optional ThreatMetrix parameter CustomAttribute4

CUA5	string		This is optional ThreatMetrix parameter CustomAttribute5
CUA6	string		This is optional ThreatMetrix parameter CustomAttribute6
CUA7	string		This is optional ThreatMetrix parameter CustomAttribute7
CUA8	string		This is optional ThreatMetrix parameter CustomAttribute8
CUA9	string		This is optional ThreatMetrix parameter CustomAttribute9
CUA10	string		This is optional ThreatMetrix parameter CustomAttribute10
CA1	string		This is optional ThreatMetrix parameter ConditionalAttribute1
CA2	string		This is optional ThreatMetrix parameter ConditionalAttribute2
CA3	string		This is optional ThreatMetrix parameter ConditionalAttribute3
CA4	string		This is optional ThreatMetrix parameter ConditionalAttribute4
CA5	string		This is optional ThreatMetrix parameter ConditionalAttribute5
CA6	string		This is optional ThreatMetrix parameter ConditionalAttribute6
CA7	string		This is optional ThreatMetrix parameter ConditionalAttribute7
CA8	string		This is optional ThreatMetrix parameter ConditionalAttribute8
CA9	string		This is optional ThreatMetrix parameter ConditionalAttribute9
CA10	string		This is optional ThreatMetrix parameter ConditionalAttribute10
CA11	string		This is optional ThreatMetrix parameter ConditionalAttribute11
CA12	string		This is optional ThreatMetrix parameter ConditionalAttribute12
CA13	string		This is optional ThreatMetrix parameter ConditionalAttribute13
CA14	string		This is optional ThreatMetrix parameter ConditionalAttribute14
CA15	string		This is optional ThreatMetrix parameter ConditionalAttribute15
CA16	string		This is optional ThreatMetrix parameter ConditionalAttribute16
CA17	string		This is optional ThreatMetrix parameter ConditionalAttribute17
CA18	string		This is optional ThreatMetrix parameter ConditionalAttribute18
CA19	string		This is optional ThreatMetrix parameter ConditionalAttribute19
CA20	string		This is optional ThreatMetrix parameter ConditionalAttribute20
CreditCardNumberHash	string		CreditCard Number Hash for ThreatMetrix
SocialSecurityNumberHash	string		SocialSecurityNumber Hash for ThreatMetrix
ACHAccountHash	string		ACHAccount Hash for ThreatMetrix
DriversLicenseHash	string		DriversLicenseHash for ThreatMetrix
Policy	string		Policy for ThreatMetrix
IsThreatMetrix	bool		True if ThreatMetrix Fraud detection is required, else false
IsAmexEnhancedAuth	bool		True if Amex enhanced Auth is required , else false

Example Key-Value Pair String before Encryption and Base64 encoding:

AuthToken=1f25d31c-e8fe-4d68-be73-f7b439bfa0a329e90de6-4e93-4374-8633-22cef77467f5
&PayerID=2833955147881261
&Amount=10.00
&CurrencyCode=USD
&ProcessMethod=Capture
&PaymentMethodStorageOption=None
&InvoiceNumber=Invoice123
&Comment1=comment1
&Comment2=comment2
&echo=echotest
&ReturnURL=https://il01addproc.propay.com:443/Return.aspx
&ProfileId=3351
&PaymentProcessType=CreditCard
&StandardEntryClassCode=
&DisplayMessage=True
&Protected=False

*The AuthToken value in the Key-Value pair string is set to the created TempToken and this value is used to encrypt.

Example Key-Value Pair String after Encryption and Base 64 encoding:

WD7n54SPFT4Pa/GdLy5Pg8rKnArxQkVQr+plCmj3Nc+vz8JZ0ugsKiFmiPw5roHKEjV7vaff1k+SG3Sxs1L9yfnnE1uLi/AVP4O1H/vpK+MOfPVFczXQ9TCPYnDT
w+r/A7c6nwUOnbEsO+xF++k0cuqEMGzaQxNV3kJfsGMegBvlzXH56jzZ39/S+p4g3PGbQ7ZP6K/bkF9URyBq2+gaDuEVWt1AF3v69CX7VVy45TTnU/zhCd8
PFLMh83lc0UJp0ZTIM60rMZOCJbGhccSZ6hujW0d4bz5qocpFxVA9lapSilrnKsFGp3a6njOMsFHgZznKgXaEAJmT59M30Uk+ml4uhKuj9Tx8n2DW6b3UVhqIvi
DXn4sXeQ1LXuOTskQJroBQzqrj9RYw/Dw7q2a2ubwr3GYVhq2fI1tZ2ohfFju4j9wRJ33tIlfs5OB0gP8R46Z2JYrWLNPPlh9ZGczrUM7sFplBepsyKlSPnw43zZek+3L
N/+Sr3nmFnxO4sQ1ZasuvxQ1L4auL6LJg1anBZcWkkNXkcFqRLaZ6LlF506t5hjl2xK3Lp8K4z4JJJ7i3

3.2.4 SPI Post Elements

The following table displays the post elements that are posted to the SPI by the client browser. These values are the sensitive payment method details that should be collected by the client's checkout page. The client checkout page can either form POST directly or submit the information in an AJAX request by setting the content-type in the request header to 'application/x-www-form-urlencoded'.

If this option is chosen the SettingsCipher must be URI encoded.

Request Values**

Element Name	Type	Max	Required	Notes
CID	Int32	-	Required	The 'CredentialId' of the Temp Token used to encrypt the settings cipher
SettingsCipher	string	-	Required	The SettingsCipher is the encrypted value of the parameters that cannot be changed by the cardholder See 3.3 Required Encrypted Parameters

- ❖ These values must be submitted for all SPI transactions

Request Values (Credit Card Transactions)

Element Name	Type	Max	Required	Notes
CardHolderName	string	50	Required*	The name on the card; *Required if Storing Payment Method
PaymentTypeId	string	-	Required	Valid values are: ▪ Visa ▪ MasterCard ▪ AMEX ▪ Discover ▪ DinersClub ▪ JCB
CardNumber	string	16	Required	*You should perform your own validation of card numbers lengths which are generally 15 or 16 digits
ExpMonth	Int32	2	Required	Month portion of credit card expiration date expressed in a 2 digit format
ExpYear	Int32	4	Required	Year portion of credit card expiration date expressed in a 2 digit format
CVV	string	4	Optional	Card security code ProtectPay will NOT store this value

Request Values (ACH Transactions)

Element Name	Type	Max	Required	Notes
BankName	string	-	Optional	The name of the financial institution *Recommended this be collected
RoutingNumber	string	-	Required	The routing number of the financial institution
Bank CountryCode	string	3	Required	The country of the financial institution *ISO 3166 standard 3 character country codes
NameOnBankAccount	string	50	Optional	The primary name on the account
Bank AccountNumber	string		Required	The account number at the financial institution

BankAccountType	string	-	Required	Valid values are: ▪ Checking ▪ Savings
StandardEntryClassCode	string	-	Required	Valid values are: ▪ PPD ▪ CCD ▪ WEB ▪ TEL ▪ IAT

Request Values (Payment Method Address Details)

Element Name	Type	Max	Required	Notes
Address1	string	50	Optional	Payer's address line 1; if your gateway supports AVS this value will be passed for AVS
Address2	string	50	Optional	Payer's address line 2; if your gateway supports AVS this value will be passed for AVS
Address3	string	50	Optional	Payer's address line 3; if your gateway supports AVS this value will be passed for AVS
City	string	25	Optional	Payer's address city; if your gateway supports AVS this value will be passed for AVS
State	string	25	Optional	Payer's address state; if your gateway supports AVS this value will be passed for AVS
PostalCode	string	10	Optional	Payer's address postal code; if your gateway supports AVS this value will be passed for AVS
Country	string	25	Optional	Payer's address country; if your gateway supports AVS this value will be passed for AVS *ISO 3166 standard 3 character country codes

Example of Form Post Data submitted to the SPI from the payer's browser

Element Name	Value
CardHolderName	John Q. Test
PaymentTypeId	Visa
CardNumber	4747474747474747
ExpMonth	12
ExpYear	20
CVV	999
Address1	123 A. Street
City	Orem
State	UT
PostalCode	84058
Country	USA
CID	2507629
SettingsCipher	mKP8qaBK+jDgpIt7NLT+eMG51qY5pPvX4a7OoaWCHryRNfTcLjXmHVyTMjooH49XCHMbVzdTWWWVxWfAqod0Hm0BS/KdPV00wYst4IwJ+n/TCExmS5VySGW0s0PVEtbSNUWN8qILJnnLUA7/QW+aS37Iwn6lPQajScBd7uYU+BCKfOyB7vZygsZS+L19UhC2LAuRXKispG3A8ilzKH5nniOlAmdJQwCRhxhaZEdCTINlovY+R8INFqb5QDhdQFgsnmRkP/AB7dGIL6TyJwF+ABm50U3XRUkVEFB2hguUa73H+KdrbLRwsPcmCrlyRwWFl4L+1rVU9RwwPlHkwoHxEE4diW9JKNLexfNaLtTZ5nCXgA/jSwFlDPrZWOd16Ibnx7M4kx4gy4ETKyv8hkBD6ydHgIaps3JBiftw+vorICvLjROt41FAdChcl5AofhzZ1HKMSEivkQVTvPLtzxSWCpidYMEGfBrFOp90Yqwr54tWe8X8TMYwRfdhk7M6wjcLiHYicd3yeurO+hai8gfrB9anzoHUulaXNrc05RdxEaQ=

- **CID maps to the following TempToken for decryption:** f2f4fcf8-75a9-4825-b7f1-c30aaaf3464601702afe-9a1e-4f3b-ba3d-9cb2120da12a

Example of AJAX Post Data submitted to the SPI from the payer's browser (SettingsCipher must be URI encoded)

CardHolderName=John Q. Test&PaymentTypeId=Visa&CardNumber=4747474747474747&ExpMonth=12&ExpYear=20&CVV=999&Address1=123 A. Street&City=Orem&State=UT&PostalCode=84058&Country=USA&CID=2507629&SettingsCipher=mKP8qaBK%2BjDgpIt7NLT%2BeMG51qY5pPvX4a7OoaWCHryRNfTcLjXmHVyTMjooH49XCHMbVzdTWWWVxWfAqod0Hm0BS%2FKdPV00wYst4lwJ%2Bn%2FTCExmS5VySGW0s0PVEtbSNUWN8qILJnnLUA7%2FQW%2BaS37lwn6lPQajScBd7uYU%2BBCKfOyB7vZygsZS%2BL19UhC2LAuRXKispG3A8ilzKH5nniOlAmdJQwCRhxhaZEdCTlNIovY%2BR8INFqb5QDhdQFgsnmRkP%2FAB7dGlL6TyJwF%2BABm50U3XRUkVEFB2hguUa73H%2BKdrbLRwsPcmCrlyRwWFI4L%2B1rVU9RwwPIHkwoHxEE4diW9JKNLexfNaLtTZ5nCXgA%2FjSwFlDPrZWOd16Ibnx7M4kx4gy4ETKyv8hkBD6ydHgIaps3JBiftw%2BvorICvLjROt41FAdChcI5AofhzZ1HKMSEivkQVTvPLtzxSWCpidYMEGfBrFOp90Yqwr54tWe8X8TMYwRfdhk7M6wjcLiHYicd3yeurO%2Bhai8gfrB9anzoHUulaXNrc05RdxEaQ%3D

ResponseCipher for Sample Request

Element Name	Value
ResponseCipher	irvN4mdV6jA0wmsSVq8yHv3F+2frLchvQpTuJj1r8lBMmYhP8ZJ5TmVGbdm1vPm91UEW3m89IgfgM5R+HjF8jwzskTX4sRExSbDv3szDdwRyUnAT9neieDJDXzdCmG6/+FhxIN/Lai0Dg5s7lWfGsl+xNmsTr/4yQ//btMl1u6AM+Jyi2+tBwGPgjMgrG5hjnqpbKwsd5k7yELDdCQHLzkjFagKjYMyfXgaRlHX7rNBpiQSKqmhESZm/XrktySfOrf80jFQtUZq0iSHxVr83W55/QGhH0TFs+avcP4yVJPuwCju2bH0Kkd6m3QtclygPM29mc2xIyIxl/8SB3bgJ8ESwWJCImIvqXJa1rcSO8y5UYkn+QL5cI9iuAaEhGjlwoC3q/q6F9kQrSlnV4ExymTio0VLFUSN4Sx6GjV5E2ITT25ypspkp05FvQPZnp+8S

- Return URL web page must be publically exposed and contain a field with name of 'ResponseCipher'

3.2.5 SPI Transitional Response html

The following html is returned if there were no errors in submitting the payment method details to the SPI. The clients browser will read interpret the HTML and execute the Script if the request was form POSTed. If the request was submitted via AJAX instead of a form POST the request the response is identical without the script being executed.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!—(c)2016-ProPay -->
<html xmlns="http://www.w3.org/1999/xhtml" >
        <head>
        <title></title>
    </head>
        <body>
            <form method="post" action="spr.aspx" id="form1">
                <div class="aspNetHidden">
                        <!--The SPI supports View States -->
                        <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="" />
                </div>
                <div class="aspNetHidden">
                        <!--The SPI supports Event Validation -->
                        <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION" value="" />
                </div>
                <div></div>
                    <!--The ResponseCipher prior to be Form POSTed to the ReturnURL -->
                    <input name="ResponseCipher" type="hidden" id="ResponseCipher" value="" />
        </form>
        </body>
</html>
<script type="text/javascript">
    //The action will be the returnURL in the SettingsCipher
        document.forms[0].action='https://il01addproc.propay.com/Return.aspx';
        document.forms[0].submit();
</script>
```

Sample Transitional Response html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
        <head>
        <title></title>
    </head>
        <body>
            <form method="post" action="spr.aspx" id="form1">
                <div class="aspNetHidden">
                        <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="" />
                </div>
                <div class="aspNetHidden">
                        <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION" value="
/wEdAAKvVXD1oYELeveMr0vHCmYPexh68czOUCZr7Ag7DJmz7Z+a1vpqAoNCsL3bMp63kqR0V0mUiq3TIIVw+e5c39X2" />
                </div>
                <div></div>
                    <input name="ResponseCipher" type="hidden" id="ResponseCipher"
value="irvN4mdV6jA0wmsSVq8yHv3F+2frLchvQpTuJj1r8lBMmYhP8ZJ5TmVGbdm1vPm91UEW3m89IgfgM5R+HjF8jwzskTX4sRExSbDv3szDdwRyUnAT9neieDJDXzd
CmG6/+FhxIN/Lai0Dg5s7lWfGsl+xNmsTr/4yQ//btMl1u6AM+Jyi2+tBwGPgjMgrG5hjnqpbKwsd5k7yELDdCQHLzkjFagKjYMyfXgaRlHX7rNBpiQSKqmhESZm/Xrkty
SfOrf80jFQtUZq0iSHxVr83W55/QGhH0TFs+avcP4yVJPuwCju2bH0Kkd6m3QtclygPM29mc2xIyIxl/8SB3bgJ8ESwWJCImIvqXJa1rcSO8y5UYkn+QL5cI9iuAaEhGjl
woC3q/q6F9kQrSlnV4ExymTio0VLFUSN4Sx6GjV5E2ITT25ypspkp05FvQPZnp+8S" />
        </form>
        </body>
</html>
<script type="text/javascript">
        document.forms[0].action='https://il01addproc.propay.com/Return.aspx';
        document.forms[0].submit();
</script>
```

Possible Error Responses

The following Error Responses will be returned by the SPI if the SPI is unable to process the request.

- ErrCode=301&ErrMsg= Invalid CID
  - The TempToken has expired
  - The CID is an invalid CID
  - The SPI did not get the CID value from the request

- ErrCode=301&ErrMsg= Invalid SettingsCipher
  - The SPI was able to acquire the CID and the SettingsCipher is improperly encrypted
  - The SPI was able to acquire the CID and the SettingsCipher is improperly encoded in the request
  - The SPI was able to acquire the CID However the SPI is unable to get the SettingsCipher value from the request

- ErrCode=348&ErrMsg= Invalid SettingsCipherLength
  - Query string exceeded max length of characters allowed
    - Developer should reduce the number of characters submitted
    - Developer should use a Form POST
    - Developer should consider using the ProtectPay Payer Management Interface: Hosted Payment Page

3.2.6 ResponseCipher Decrypted Values

The following table displays the response values that are returned by the SPI after decryption. The ResponseCipher is returned in the ResponseCipher element of the transitional HTML and will be form POSTed to the ReturnURL as indicated in the transitional HTML script.

*Not all values are returned. See the individual notes for each response value.

Seamless Payment Interface – Redirect Response Values

Name	Type	Notes
Action	string	"Complete" indicates the request was completed "Err" indicates one or multiple errors with the transaction request
ErrCode	String	Numeric Code returned only when Action=Err. This indicates a problem with the SPI Request *Multiple ErrCode#s may be returned. Example: ErrCode1 = , ErrCode2 = , …
ErrMsg	String	Text detail returned only when there Action=Err. This indicates a problem with the SPI Request *Multiple ErrCode#s may be returned. Example: ErrMsg1 = , ErrMsg2 = , …
echo	String	*Returned only if submitted
ProcessResult	String	*Result of transaction processing request
ProcessResultResultCode	String	Result Code of transaction request *Not returned if there was an error processing the transaction request
ProcessResultResultMessage	String	Result text description of transaction request *Not returned if there was an error processing the transaction request or if the ProcessResultResultCode=00
ProcErrCode	string	Processing Error Code for transaction Processing *Only returned if there was an error processing the transaction request
ProcErrMsg	string	Processing Error text description for transaction *Only returned if there was an error processing the transaction request
StoreErrCode	string	Storage Error Code for transaction Processing *Only returned if there was an error storing the payment method
StoreErrMsg	string	Storage Error text description for transaction *Only returned if there was an error storing the payment method
ProcessResultAuthorizationCode	string	The auth code supplied by the issuing bank *Only returned on a successful transaction
ProcessResultCvvCode	string	The issuer CVV response *Only returned if supplied *ProtectPay WILL NOT store the CVV code of a Credit Card Payment Method
ProcessResultAVSCode	string	AVS response produced by gateway *Only returned if AVS information is supplied and AVS is supported by your gateway
PayerId	string	Id of the payer that is the owner of the payment method
PaymentMethodId	string	*Only returned if a payment method was stored as defined by PaymentMethodStorageOption
CardholderName	string	*Returned only for credit card transactions
ObfuscatedAccountNumber	string	*Returned only for credit card transactions obfuscated for security
ExpireDate	string	*Returned only for credit card transactions
Address1	string	*Returned only for credit card transaction.

Address2	string	*Returned only for credit card transactions
Address3	string	*Returned only for credit card transactions
City	string	*Returned only for credit card transactions
State	string	*Returned only for credit card transactions
PostalCode	string	*Returned only for credit card transactions
Country	string	*Returned only for credit card transactions
BankName	string	*Returned only for ACH transactions
ProcessResultTransactionHistoryID	string	Unique transaction number assigned by ProtectPay
ProcessResultTransactionId	string	Transaction number assigned by processor (Gateway)
GrossAmt	string	Gross amount of transaction repeated back to you
NetAmt	string	Net amount of transaction after fees charged; *ProPay Gateway Only
PerTransFee	string	Per transaction fee ; *ProPay Gateway Only
Rate	string	Percentage fee ; *ProPay Gateway Only
GrossAmtLessNetAmt	string	Total of fees; *ProPay Gateway Only

Example of a decrypted response for successful credit card transaction:

Action=Complete
Echo=echotest
PayerID=6192936083671743
ObfuscatedAccountNumber=474747******4747
ExpireDate=1215
CardholderName=John Q Test
Address1=123 A St.,
Address2=
Address3=
City=Orem
State=UT
PostalCode=84058
Country=USA
PaymentMethodId=bb466e8d-0cdb-44db-8ef4-939207c204b3
ProcessResult=Success
ProcessResultAuthorizationCode=A11111
ProcessResultAVSCode=T
ProcessResultResultCode=00
ProcessResultResultMessage=
ProcessResultTransactionHistoryID=7909962
ProcessResultTransactionId=524

Amount=10.00
GrossAmt=10.00
NetAmt=9.32
PerTransFee=0.35
Rate=3.25
GrossAmtLessNetAmt=0.68

Example of a decrypted response for successful tokenization, but declined card transaction:

Action=Complete
&Echo=echotest
&PayerID=2833955147881261
&ObfuscatedAccountNumber=474747******4747
&ExpireDate=1212
&CardholderName=John Q Test
&Address1=123 A St.,
&Address2=
&Address3=
&City=Orem
&State=UT
&PostalCode=84058
&Country=USA
&PaymentMethodId=58bff1ed-e8a7-44e2-bce3-71a389e86eec
&ProcErrCode=51
&ProcErrMsg=Insufficient funds/over credit limit

Example of a decrypted response of a storage error:

Action=Complete
&Echo=echotest
&PayerID=7588958043622683
&ExpireDate=1212
&CardholderName=John Q Test
&Address1=123 A St.,
&Address2=
&Address3=
&City=Orem
&State=UT
&PostalCode=84058
&Country=USA
&StoreErrCode=308

&StoreErrMsg=CreditCard number is invalid for specified type

Example of a decrypted response with an error with the SPI request:

ErrCode=301
&ErrMsg=Invalid Settings Cipher

Example of a decrypted response with multiple SPI request error codes:

Action=Err
&ErrCode0=301
&ErrMsg0=Invalid Bank AccountNumber
&ErrCode1=301
&ErrMsg1=Invalid RoutingNumber
&ErrCode2=301
&ErrMsg2=Invalid BankAccountType
&ErrCode3=301
&ErrMsg3=Invalid StandardEntryClassCode

Example of a decrypted response with successful tokenization and gateway refusal to process the transaction:

Action=Complete
&Echo=echotest
&PayerID=3664332127321408
&ObfuscatedAccountNumber=474747******4747
&ExpireDate=1212
&CardholderName=John Q Test
&Address1=123 A St.,
&Address2=
&Address3=
&City=Orem
&State=UT
&PostalCode=84058
&Country=USA
&PaymentMethodId=ddc3310b-a737-4714-bf10-e52ab6cbb1b5
&ProcErrCode=204
&ProcErrMsg=Account Expired

3.2.7 SampleCheckoutPage.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
        <!–(c)2016-ProPay -->
        <!--
        *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.
    -->
    <head id="Head1">
        <title>SPI Submit Payment</title>
        <style>
            .PageContent{ width: 620px;color: #000000;margin-left: auto;margin-right: auto;font-family: Verdana,Arial;font-size:
14px; }
            .FieldSet{ border: 1px solid #D3D3D3;padding: 10px 10px 20px 10px;margin-bottom: 25px;border-radius: 4px;-moz-border-
radius: 4px;-webkit-border-radius: 4px; }
            .Legend{ padding-left: 7px;padding-right: 7px;color: #A3A3A3; }
            .FormLabel{ font-family: monospace,Verdana,Arial; font-size: 14px; }
            .FormLeftColumn{ width: 240px;display: inline-block;white-space: nowrap;text-align: right;vertical-align: top;margin-
right: 8px;padding-top: 4px; }
            .FormRightColumn{ display: inline-block;vertical-align: middle; }
            .FormRow{ margin: 5px; }
            .LabelColon{ padding-left: 3px; }
            .TextBox{ width: 289px;height: 28px;border: 1px solid #9B9B9B;padding: 3px 3px 3px 6px;font-family:
monospace,Verdana,Arial;font-size: 14px;border-radius: 2px;-moz-border-radius: 2px;-webkit-border-radius: 2px; }
            .DropDown{ width: 300px;height: 28px;border: 1px solid #9B9B9B;padding: 3px;font-family: monospace,Verdana,Arial;font-
size: 14px;border-radius: 2px;-moz-border-radius: 2px;-webkit-border-radius: 2px; }
            .FormCardNumber{width: 150px;}
            .FormCardType{ width: 150px; }
            .FormExpDate{ width: 75px; }
            .FormCvv{ width: 95px; }
            .FormZip{ width: 95px; }
            .SubmitButton{float: right;}
            .RequiredField{ color: red;font-size: 11px; }
        </style>
    </head>
    <body>
        <div class="PageContent">
            <h4>Payment Information</h4>
```

```
<form id="PaymentMethodForm" method="post" action="https://protectpaytest.propay.com/pmi/spr.aspx">
    <fieldset class="FieldSet">
        <legend class="Legend LegendIe">Credit Card Payment Information</legend>
        <div id="CCType" class="">
            <div class="FormRow">
                <span class="FormLeftColumn">
                    <span class="FormLabel">Card Holder Name </span>
                    <span class="LabelColon">:</span>
                </span>
                <span class="FormRightColumn">
                    <input id="CardHolderName"class="TextBox" maxlength="50" name="CardHolderName"/>
                </span>
            </div>
            <div class="FormRow">
                <span class="FormLeftColumn">
                    <span class="RequiredField">*</span>
                    <span class="FormLabel">Card Number </span>
                    <span class="LabelColon">:</span>
                </span>
                <span class="FormRightColumn">
                    <input id="CardNumber" class="TextBox FormCardNumber" maxlength="16" autocomplete="off"
name="CardNumber" />
                </span>
            </div>
            <div class="FormRow">
                <span class="FormLeftColumn">
                    <span class="RequiredField">*</span>
                    <span class="FormLabel">Card Type </span>
                    <span class="LabelColon">:</span>
                </span>
                <span>
                    <select id="PaymentTypeId" class="DropDown FormCardType" name="PaymentTypeId">
                        <option value="Visa">Visa</option>
                        <option value="MasterCard">MasterCard</option>
                        <option value="AMEX">American Express</option>
                        <option value="Discover">Discover</option>
                        <option value="DinersClub">Diners Club</option>
                        <option value="JCB">JCB</option>
                    </select>
                </span>
            </div>
            <div class="FormRow">
```

```
        <span class="FormLeftColumn">
            <span class="RequiredField">*</span>
            <span class="FormLabel">Expiration Date </span>
            <span class="LabelColon">:</span>
        </span>
        <span class="FormRightColumn">
            <select id="ExpMonth" class="DropDown FormExpDate" name="ExpMonth">
                <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>
             / 
            <select id="ExpYear" class="DropDown FormExpDate" name="ExpYear">
                <option value="16">2016</option>
                <option value="17">2017</option>
                <option value="18">2018</option>
                <option value="19">2019</option>
                <option value="20">2020</option>
                <option value="21">2021</option>
                <option value="22">2022</option>
                <option value="23">2023</option>
                <option value="24">2024</option>
                <option value="25">2025</option>
                <option value="26">2026</option>
            </select>
        </span>
    </div>
    <div class="FormRow">
        <span class="FormLeftColumn">
            <span class="FormLabel">CVV</span>
            <span class="LabelColon">:</span>
        </span>
        <span class="FormRightColumn">
```

```
                <input id="CVV" class="TextBox FormCvv" maxlength="4" autocomplete="off" name="CVV" />
            </span>
        </div>
    </div>
</fieldset>
<fieldset class="FieldSet">
    <legend class="Legend LegendIe">ACH Payment Information</legend>
    <div id="ACHType" class="">
        <div class="FormRow">
        <span class="FormLeftColumn">
            <span class="FormLabel">Name on Account </span>
            <span class="LabelColon">:</span>
        </span>
        <span class="FormRightColumn">
            <input id="NameOnBankAccount" class="TextBox" maxlength="50" name="NameOnBankAccount"/>
        </span>
        </div>
        <div class="FormRow">
            <span class="FormLeftColumn">
                <span class="FormLabel">Financial Institution: </span>
                <span class="LabelColon">:</span>
            </span>
            <span class="FormRightColumn">
                <input id="BankName" class="TextBox" maxlength="25" name="BankName"/>
            </span>
        </div>
        <div class="FormRow">
            <span class="FormLeftColumn">
                <span class="RequiredField">*</span>
                <span class="FormLabel">Routing Number: </span>
                <span class="LabelColon">:</span>
            </span>
            <span class="FormRightColumn">
                <input id="RoutingNumber" class="TextBox" maxlength="10" name="RoutingNumber"/>
            </span>
        </div>
        <div class="FormRow">
            <span class="FormLeftColumn">
                <span class="RequiredField">*</span>
                <span class="FormLabel">Account Number: </span>
                <span class="LabelColon">:</span>
            </span>
```

```
                        <span class="FormRightColumn">
                            <input id="AccountNumber" class="TextBox" maxlength="25" name="Bank AccountNumber"
autocomplete="off"/>
                        </span>
                    </div>
                     <div class="FormRow">
                         <span class="FormLeftColumn">
                            <span class="RequiredField">*</span>
                            <span class="FormLabel">Account Type </span>
                            <span class="LabelColon">:</span>
                        </span>
                        <span>
                            <select id="BankAccountType" class="DropDown" name="BankAccountType">
                                <option value="Checking">Checking</option>
                                <option value="Savings">Savings</option>
                            </select>
                        </span>
                    </div>
                    <div class="FormRow">
                        <span class="FormLeftColumn">
                            <span class="RequiredField">*</span>
                            <span class="FormLabel">Account Country </span>
                            <span class="LabelColon">:</span>
                        </span>
                        <span>
                            <select id="BankCountryCode" class="DropDown" name="Bank CountryCode">
                                <option value="USA">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>
</fieldset>
<fieldset class="FieldSet">
    <legend class="Legend LegendIe">Billing Information</legend>
    <div class="FormRow">
        <span class="FormLeftColumn">
            <span class="FormLabel">Address Line 1 </span>
            <span class="LabelColon">:</span>
        </span>
        <span class="FormRightColumn">
            <input id="Address1" class="TextBox" maxlength="50" name="Address1" />
        </span>
    </div>
    <div class="FormRow">
        <span class="FormLeftColumn">
            <span class="FormLabel">Address Line 2 </span>
            <span class="LabelColon">:</span>
        </span>
        <span class="FormRightColumn">
            <input id="Address2" class="TextBox" maxlength="50" name="Address2"/>
        </span>
    </div>
    <div class="FormRow">
        <span class="FormLeftColumn">
            <span class="FormLabel">Address Line 3 </span>
            <span class="LabelColon">:</span>
        </span>
        <span class="FormRightColumn">
            <input id="Address3" class="TextBox" maxlength="50" name="Address3" />
        </span>
    </div>
    <div class="FormRow">
        <span class="FormLeftColumn">
            <span class="FormLabel">City </span>
            <span class="LabelColon">:</span>
        </span>
        <span class="FormRightColumn">
```

```
            <input id="City" class="TextBox" maxlength="25" name="City"/>
        </span>
    </div>
    <div class="FormRow">
        <span class="FormLeftColumn">
            <span class="FormLabel">State </span>
            <span class="LabelColon">:</span>
        </span>
        <span class="FormRightColumn">
            <select id="State" class="DropDown" style="display: inline-block;" name="State">
                <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>
        </span>
    </div>
    <div class="FormRow">
        <span class="FormLeftColumn">
            <span class="FormLabel">Postal Code </span>
            <span class="LabelColon">:</span>
        </span>
        <span class="FormRightColumn">
            <input id="PostalCode" class="TextBox FormZip" maxlength="10" name="PostalCode" />
        </span>
    </div>
    <div class="FormRow">
        <span class="FormLeftColumn">
            <span class="FormLabel">Country </span>
            <span class="LabelColon">:</span>
```

```
</span>
<span class="FormRightColumn">
    <select id="Country" class="DropDown" name="Country">
        <option value="USA">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>
                </fieldset>
                <input style="display: none;" id="CID" name="CID" value=""/>
                <input style="display: none;" id="SettingsCipher" name="SettingsCipher" value=""/>
            </form>
            <input type="button" id="btnSubmit" name="btnSubmit" value="Submit Form" class="" onclick="formPost()" />
            <input type="button" id="btnSubmit" name="btnSubmit" value="Submit AJAX" class="SubmitButton" onclick="AJAXRequest()"
/>
        </div>
        <script>
            function formPost() {
                 //This function should be modified to meet the merchants needs and serves as a demonstration of pre-validation
prior to submission
                //The Type of Payment Method to process (CreditCard or ACH) is determined by the value set in the SettingsCipher
prior to loading this page
                //This sample validation function is designed specifically for the SampleCheckout.html
                //A Developer should only use the method that the SPI is set to process.
                var PaymentForm = document.getElementById("PaymentMethodForm");
                PaymentForm.action='https://il01addproc.propay.com/Return.aspx';
```

```
            if (validateCC()) { PaymentForm.submit(); return; }
            if (validateACH()) { PaymentForm.submit(); return; }

            //All Payment Method Details failed validation
            alert("Payment Details Validation Failed Please Review and Re-Submit");
        }


        function AJAXRequest() {
        //This function should be modified to meet the merchants needs and serves as a demonstration of pre-validation prior
to submission
        //The Type of Payment Method to process (CreditCard or ACH) is determined by the value set in the SettingsCipher prior
to loading this page
            var payload = 'CID=' + document.getElementById("CID").value
                        + '&SettingsCipher=' + encodeURIComponent(document.getElementById("SettingsCipher").value);

            var ccPayload = '&CardHolderName=' + document.getElementById("CardHolderName").value
                          + '&CardNumber=' + document.getElementById("CardNumber").value
                          + '&PaymentTypeId=' + document.getElementById("PaymentTypeId").value
                          + '&ExpMonth=' + document.getElementById("ExpMonth").value
                          + '&ExpYear=' + document.getElementById("ExpYear").value
                          + '&CVV=' + document.getElementById("CVV").value

            var achPayload = '&NameOnBankAccount=' + document.getElementById("NameOnBankAccount").value
                           + '&BankName=' + document.getElementById("BankName").value
                           + '&RoutingNumber=' + document.getElementById("RoutingNumber").value
                           + '&Bank AccountNumber=' + document.getElementById("AccountNumber").value
                           + '&ExBankAccountTypepYear=' + document.getElementById("BankAccountType").value
                           + '&Bank CountryCode=' + document.getElementById("BankCountryCode").value

            var avsPayload = '&Address1=' + document.getElementById("Address1").value
                           + '&Address2=' + document.getElementById("Address2").value
                           + '&Address3=' + document.getElementById("Address3").value
                           + '&City=' + document.getElementById("City").value
                           + '&State=' + document.getElementById("State").value
                           + '&PostalCode=' + document.getElementById("PostalCode").value
                           + '&Country=' + document.getElementById("Country").value

            //This sample validation function is designed specifically for the SampleCheckout.html
            //A Developer should only use the method that the SPI is set to process.
            if (validateCC()) {
                payload += ccPayload;
```

```
            payload += avsPayload;
            submitAJAX(payload);
            return;
        }
        if (validateACH()) {
            payload += achPayload;
            payload += avsPayload;
            submitAJAX(payload);
            return;
        }

        //All Payment Method Details failed validation
        alert("Payment Details Validation Failed Please Review and Re-Submit");
    }

    function submitAJAX(payload) {
        var xhttp = new XMLHttpRequest();
        if (xhttp != null) {
            xhttp.open("POST", "https://protectpaytest.propay.com/pmi/spr.aspx", true);
            xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
            xhttp.onreadystatechange = function () {
                if (xhttp.readyState === 4) {
                    var resp = xhttp.responseText;
                    alert(resp);
                }
            }
            xhttp.send(payload);
        }
    }

    function validateCC() {
        //Credit Card Payment Method Type Required
        var CardBrand = document.getElementById("PaymentTypeId").value;
        var CCNum = document.getElementById("CardNumber").value;
        var ExpMonth = document.getElementById("ExpMonth").value;
        var ExpYear = document.getElementById("ExpYear").value;

        //Only optional field for Card Validation
        var CVV = document.getElementById("CVV").value;

        if (CCNum == "") { alert("Card number not present"); return false; }
```

```
//Validate Card Number using Luhn Check
var nCheck = 0
var nDigit = 0
var bEven = false;
CCNum = CCNum.replace(/\D/g, "");

for (var n = CCNum.length - 1; n >= 0; n--) {
    var cDigit = CCNum.charAt(n),
                              nDigit = parseInt(cDigit, 10);

    if (bEven) {
        if ((nDigit *= 2) > 9) nDigit -= 9;
    }

    nCheck += nDigit;
    bEven = !bEven;
}

if ((nCheck % 10) != 0) { alert("Card number is not a valid card number"); return false; }

//Validate Card Expiration Date
var currentDate = new Date();
var cardExpirationDate = new Date((2000 + parseInt(ExpYear)), parseInt(ExpMonth), 0, 23, 59, 59);

if (cardExpirationDate < currentDate) { alert("Card expiration date is invalid"); return false; }

//Validate Card Brand IIN and CVV length
var cardValidation = {
    Visa: /^4[0-9]{12}(?:[0-9]{3})?$/,
    MasterCard: /^5[1-5][0-9]{14}$/,
    AMEX: /^3[47][0-9]{13}$/,
    Discover: /^6(?:011|5[0-9]{2})[0-9]{12}$/,
    DinersClub: /^3(?:0[0-5]|[68][0-9])[0-9]{11}$/,
    JCB: /^(?:2131|1800|35\d{3})\d{11}$/
};

//Validate IIN and CVV Length
var validBrandNumber;
switch (CardBrand) {
    case "Visa":
        if (!(cardValidation.Visa.test(CCNum))) { alert("Card number does not match Visa IIN"); return false; }
        if (CVV != "" && CVV.length != 3) { alert("CVV does not match VISA CVV requirements"); return false; }
```

```
                    break;

                case "MasterCard":
                    if (!(cardValidation.MasterCard.test(CCNum))) { alert("Card number does not match MasterCard IIN"); return
false; }
                    if (CVV != "" && CVV.length != 3) { alert("CVV does not match MasterCard CVV requirements"); return false;
}
                    break;

                case "AMEX":
                    if (!(cardValidation.AMEX.test(CCNum))) { alert("Card number does not match AMEX IIN"); return false; }
                    if (CVV != "" && CVV.length != 4) { alert("CVV does not match AMEX CVV requirements"); return false; }
                    break;

                case "Discover":
                    if (!(cardValidation.Discover.test(CCNum))) { alert("Card number does not match Discover IIN"); return
false; }
                    if (CVV != "" && CVV.length != 3) { alert("CVV does not match Discover CVV requirements"); return false; }
                    break;

                case "DinersClub":
                    if (!(cardValidation.DinersClub.test(CCNum))) { alert("Card number does not match DinersClub IIN"); return
false; }
                    if (CVV != "" && CVV.length != 3) { alert("CVV does not match DinersClub CVV requirements"); return false;
}
                    break;

                case "JCB":
                    if (!(cardValidation.JCB.test(CCNum))) { alert("Card number does not match JCB IIN"); return false; }
                    if (CVV != "" && CVV.length != 3) { alert("CVV does not match JCB CVV requirements"); return false; }
                    break;

            }

            return true;
        }

        function validateACH() {
            //ACH Payment Method Type Required
            var RoutingNumber = document.getElementById("RoutingNumber").value;
            var AccountNumber = document.getElementById("AccountNumber").value;
            var AccountCountryCode = document.getElementById("BankCountryCode").value;
```

```
            //Must have both Routing and Account Number
            if (RoutingNumber == "") { alert("ACH routing number not present"); return false; }
            if (AccountNumber == "") { alert("ACH account number not present"); return false; }

            //Routing Number Validation
            switch (AccountCountryCode) {
                case "USA":
                    //Routing Number is a Number
                    try {
                        parseInt(RoutingNumber);
                    }
                    catch (err) {
                        alert("ACH routing number is not Valid"); return false;
                    }
                    if (RoutingNumber.length != 9 || RoutingNumber.charAt(0) == 5) { alert("ACH routing number is not Valid");
return false; }

                    //First two digits are between 01-12, 21-32, 61-72, 80
                    var validStart = RoutingNumber.substring(0, 2);
                    var regexp = /0(?=[1-9])|1(?=[0-2])|2(?=[0-9])|3(?=[0-2])|6(?=[0-9])|7(?=[0-2])|80/
                    if (!validStart.match(regexp)) { alert("ACH routing number is not Valid"); return false; }

                    //ABA Routing Number Checksum
                    var n = 0;
                    for (i = 0; i < RoutingNumber.length; i += 3) {
                        n += parseInt(RoutingNumber.charAt(i), 10) * 3
                          + parseInt(RoutingNumber.charAt(i + 1), 10) * 7
                          + parseInt(RoutingNumber.charAt(i + 2), 10);
                    }
                    if (n == 0 || n % 10 != 0) { alert("ACH routing number is not Valid"); return false; }
                    break;
            }

            return true;
        }


    </script>
</body>
</html>
```