BluePay Hosted Payment Form (SHPF)

BluePay Hosted Payment Form

Reference Guide

April 2025

http://www.fiserv.com

This document is classified as Fiserv Public.

Content

About this Document

This document provides technical guidance to on-board and manage a BluePay Payment Gateway merchant account.

Intended Audience

The intended audience of this document is merchant, partners, and developers who are responsible for integrating payment processing functionality with the BluePay Payment Gateway.

Assistance & Feedback

Use the following contact information for help with the BluePay Payment Gateway integration or to provide feedback on this document.

Support Level	Contact Details
BluePay Integration Support Team	bluepayintegration@fiserv.com

Support hours are Monday through Friday 8:00am to 5:00pm (CST UTC-6).

Overview

BluePay Hosted Payment Form allows the merchant to supply BluePay with an HTML payment form and associated static content, to be hosted on the BluePay servers. This may help reduce the merchant's PCI liability. This will generally be used in conjunction with the bp10emu API; the HTML form's action will be a POST to bp10emu.

System Design

1. The merchant will design HTML and associated javascript/css/images/etc. Within the HTML, they can put placeholders which will be replaced later with variables. These placeholders have the form of ${IDENTIFIER} so for example, you might do something like:

   ```
   <INPUT type="hidden" name="TAMPER_PROOF_SEAL" value="${OUR_TPS}">
   ```

2. Once this HTML is provided to BluePay (via secured email), BluePay will put it in an accessible place on secure.bluepay.com.

3. To use the system:

   - Merchant causes customer's browser to make a GET or POST request to https://secure.bluepay.com/interfaces/shpf
   - BluePay reads the posted query parameter named SHPF_FORM_ID and loads the appropriate HTML on our system, validates the SHPF_TPS, if present, then it takes all query parameters that do not begin with SHPF_, and replaces any ${IDENTIFIER} strings in the HTML with the value of the same IDENTIFIER query parameter.
   - The parsed HTML is returned to the customer. Presumably, the parsed HTML contains a form, the action of which is to POST to bp10emu, and as per the bp10emu specs, the response from bp10emu is a redirect -- presumably to the merchant's servers.

Input

Input to BluePay Hosted Payment Form is in the form of an HTTP GET or POST, with certain query parameters included.

Use the following URL for the BluePay Hosted Payment Forms:

URL: https://secure.bluepay.com/interfaces/shpf

Query Parameters

Note all variable names that are recognized by the BluePay Hosted Payment Form are prefixed with "SHPF_". Any variable that begins with SHPF_ except SHPF_ACCOUNT_ID is considered ineligible for variable substitution. All other query parameters are used in substitution.

Parameter	Description
SHPF_FORM_ID	A BluePay assigned identifier, up to a max of 16 chars (alphanumeric - assigned by BluePay)
SHPF_ACCOUNT_ID	The 12-digit BluePay gateway account ID
SHPF_TPS_DEF	A space-separated list of fields in the order they are to be used in the calculation of the SHPF_TPS.
SHPF_TPS_HASH_TYPE	The algorithm used to compute the SHPF_TPS. Accepted values are 'MD5', 'SHA256', 'SHA512', 'HMAC_SHA256', or 'HMAC_SHA512'. Merchant's 'Hash Type in APIs' value is used if this parameter is not present. See NOTES ON SHPF_TPS_HASH_TYPE for more details.
SHPF_TPS	Hash for security, using selected algorithm (either SHPF_TPS_HASH_TYPE or account's 'Hash Type in APIs' value). See CALCULATING THE SHPF_TPS for more details.
MASTER_ID (optional)	The transaction ID of a previous transaction to make that transaction's data available for substitution into a template. MASTER_ID must be listed in SHPF_TPS_DEF to use this feature.

Variable Substitution

Any query parameter not prefixed with SHPF_ is passed into our variable substitution system, which simply parses through the HTML - a single pass, so no nested variables are allowed - and replaces each instance of ${X} with the value of the same-named X query parameter. Any remaining strings in the HTML that are of the form ${X} (i.e. variables for which no substitution was provided) will be replaced with nothing, so no strings of the form ${X} will remain in the output HTML.

In addition, the following substitution variables are in the hosted payment form:

Parameter	Description
SHPF_STATIC_PATH	The correct pathname for a client to use to request static content. See "HANDLING STATIC CONTENT" below.
SHPF_TIMESTAMP	The current time, in ISO format: "YYYY-MM-DD HH:MM:SS"
SHPF_MISSING_PCOUNT	The number of fields that should be included in SHPF_TPS_DEF and are not.
SHPF_VULNERABLE_FIELDS	A comma-separated list of fields that are missing from the SHPF_TPS_DEF.

SHPF_YEARBLOCK	Substitution variable that will display a list of <option> tags for the current year and the next 15 years.
	Usage Sample:
	`<select name="CC_EXPIRES_YEAR">`
	`${SHPF_YEARBLOCK}`
	`</select>`

If a MASTER_ID is provided, the following field names are available for substitution to insert information from the previous transaction into the hosted payment form.

Parameter	Description
card_expire	Date in MMYY format.
amount	Amount value that is used in the current transaction.
order_id	Alphanumeric merchant-assigned order ID that is associated with the merchant account processing the current transaction orders.
addr1	Customer address1 details that are associated with bank account used for the current transaction.
addr2	Customer address2 details that are associated with bank account used for the current transaction.
state	State of the customer that is associated with bank account used for the current

Parameter	Description
	transaction.
zip	Zip code of the customer that is associated with bank account used for the current transaction.
memo	Memo associated with the current transaction.
phone	Phone number associated with the current transaction.
email	Email address of the customer that is associated with bank account used for the current transaction.
issue_date	Date and time of the previous transaction in YYYY-MM-DD HH:MM:SS format.
city	City name associated with the current transaction.
name1	Name1 associated with the current transaction.
name2	Name2 associated with the current transaction.
payment_type	Supportive value that describes the type of payment used in the current transaction.
payment_account	Masked payment account.
card_type	Card type associated with the current transaction.
invoice_id	Invoice number associated with the current transaction.
country	Country name associated with the current transaction.
custom_id	Custom ID1 that is associated with bank account used for the current transaction.
custom_id2	Custom ID2 that is associated with bank account used for the current transaction.
bank_name	Bank name that is associated with the account number used in the current transaction.
merchdata	Merchant data associated with the current transaction.

Output

The only output from the BluePay Hosted Payment Form is the parsed HTML (and appropriate headers) returned to the customer's browser as a response to the input request, with the exception of Error Conditions.

Error Conditions

If there is an unrecoverable error, BluePay Hosted Payment Form will display a default error HTML created by BluePay.

The following are the current possible error conditions that will result in this occurring:

- "Missing SHPF_FORM_ID" = Was not provided in query parameters
- "Invalid SHPF_FORM_ID" = Was provided, but no such form exists
- "Unable to locate account key." = Invalid (or unsent) SHPF_ACCOUNT_ID
- "Security Error" = SHPF_TPS is incorrect

Handle Static Content

All static content must be prefixed with ${SHPF_STATIC_PATH} if it is to be loaded off the BluePay server, as illustrated below:

"Original" HTML on merchant's server:	`<img src="images/foo.jpg" alt="A foo.">`
HTML to load foo.jpg from images subdir on BluePay:	`<img src="${SHPF_STATIC_PATH}images/foo.jpg" alt="A foo.">`

Notes on SHPF_TPS_HASH_TYPE

BluePay uses cryptographic hash (or "digest") functions as a means of both protecting transaction data from being altered and ensuring that the transaction is genuine. A cryptographic hash function is an algorithm that maps data of any size to a bit string of a fixed size that cannot be deciphered.

All merchants have a default hash type assigned to their account. This can be viewed and updated on the merchant's Account Admin page of BluePay's Gateway (https://secure.bluepay.com) under "Hash Type in APIs". Merchants may override their default by including the SHPF_TPS_HASH_TYPE query parameter.

The default hash type and the SHPF_TPS_HASH_TYPE may be any of the following algorithms (in hexadecimal form):

Hexadecimal Value	Description
MD5	Use md5sum or a similar program to calculate a 128-bit hash, then convert it into hexadecimal form; result is 32 hexadecimal characters
SHA256	Use sha256sum or a similar program to calculate a 256-bit hash, then convert it into hexadecimal form; result is 64 hexadecimal characters
SHA512	Use sha512sum or a similar program to calculate a 512-bit hash, then convert it into hexadecimal form; result is 128 hexadecimal characters
HMAC_SHA256	A 128-bit hash, resulting in a sequence of 64 hexadecimal characters
HMAC_SHA512	A 128-bit hash, resulting in a sequence of 128 hexadecimal characters

Steps to find the HMAC of either SHA256 (HMAC_SHA256) or SHA512 (HMAC_SHA512):

1. Compare the length of the key (the merchant's Secret Key) to the hash's input blocksize. SHA256 blocksize = 64, SHA512 blocksize = 128.
    - If length of key is > blocksize, set the key's value to the hash of the original key.
    - If length of key is < blocksize, pad the key to the right with zeros until its length equals the blocksize.
2. Create the inner key (inner_key):
    - Create an inner padding value of 0x36 repeated the blocksize number of times.
    - Perform a bitwise exclusive-OR (XOR) on the key and the inner padding to create the inner key.
3. Create the outer key (outer_key):
    - Create an outer padding value of 0x5c repeated the blocksize number of times.
    - Perform a bitwise exclusive-OR (XOR) on the key and the outer padding to create the outer key.

4. Calculate the hash of the inner key concatenated with the text string, then calculate the hash of the outer key concatenated with the previous hash result:
    - hash (outer_key + hash(inner_key + string))
5. Convert the result into a hex string.

When using a program or function to calculate the SHPF_TPS, make sure that it will accept a text string (or "message") argument and will return the hashed string (or "message digest") in hexadecimal form.

Calculating the SHPH_TPS

STEP ONE

Concatenate the values of the fields that make up the SHPF_TPS_DEF in same order that they are listed. Use "" (empty string - no space) as the value for any fields that are empty or unsent.

For example:
If SHPF_TPS_DEF = "SHPF_FORM_ID SHPF_ACCOUNT_ID SHPF_TPS_DEF SHPF_TPS_HASH_TYPE TPS_DEF TPS_HASH_TYPE TAMPER_PROOF_SEAL AMOUNT REBILLING REB_CYCLES REB_AMOUNT REB_EXPR REB_FIRST_DATE CUSTOM_ID CUSTOM_ID2" ('+' represents string concatenation, and the field names represent the contents of the respective fields):

Message:	= SHPF_FORM_ID + SHPF_ACCOUNT_ID + SHPF_TPS_DEF + SHPF_TPS_HASH_TYPE + TPS_DEF + TPS_HASH_TYPE + TAMPER_PROOF_SEAL + AMOUNT + REBILLING + REB_CYCLES + REB_AMOUNT + REB_EXPR + REB_FIRST_DATE + CUSTOM_ID + CUSTOM_ID2

STEP TWO

- If SHPF_TPS_HASH_TYPE is "" or is not sent, the merchant's 'Hash Type in APIs' value will determine which hash function to use.
- If SHPF_TPS_HASH_TYPE is 'MD5', 'SHA256', or 'SHA512', find the md5sum, sha256sum, or sha512sum of (the merchant's Secret Key + message) in hex format.
- If SHPF_TPS_HASH_TYPE is 'HMAC_SHA256' or 'HMAC_SHA512', find the HMAC_SHA256 or HMAC_SHA512 of (the merchant's Secret Key, message) in hex format.

Note on SHPF_TPS, XSS, IE8

If you do not include all variable substitution fields in the SHPF_TPS_DEF, then there is a possibility of someone using your form in a cross-site-scripting (XSS) attack. We strongly recommend protecting all substitution fields with the tamper proof seal. If you use substitution variables to provide any HTML, IE8 will block the request in an attempt to prevent this.

However, if, and only if, all substitution variables are included in the SHPF_TPS_DEF, then the BluePay Hosted Payment Form will print a header that overrides this behavior in IE8, allowing the page to display.

If you have IE8, you can view an example of this by following the two URLs below. They both provide the same content (some very badly-formed HTML) but the first one will be blocked by IE8, and the second will not. The difference is the inclusion of all substitution variables in the SHPF_TPS_DEF.

XSS-Vulnerable:	https://secure.bluepay.com/interfaces/shpf?SHPF_FORM_ID=shpftest&title=Hello&heading=Heading&text=%3Cform%20action%3D%22foozle.com%22%3E%3Cselect%3EXSS%3C%2Fselect%3E%3C%2Fform%3E&SHPF_ACCOUNT_ID=100001302235&SHPF_TPS_DEF=SHPF_ACCOUNT_ID&SHPF_TPS=eaa59181c84800c6cedca53106395748
XSS-protected:	https://secure.bluepay.com/interfaces/shpf?SHPF_FORM_ID=shpftest&title=Hello&heading=Heading&text=%3Cform%20action%3D%22foozle.com%22%3E%3Cselect%3EXSS%3C%2Fselect%3E%3C%2Fform%3E&SHPF_ACCOUNT_ID=100001302235&SHPF_TPS_DEF=SHPF_ACCOUNT_ID&SHPF_TPS=eaa59181c84800c6cedca53106395748

Examples of SHPF_TPS_DEF, SHPF_TPS_HASH_TYPE, & SHPF_TPS

It is left to the discretion of the merchant to decide what fields to include in the SHPF_TPS_DEF -- if any -- but it is highly recommended to include any fields which you do not wish to allow to be changed by a malicious user. As you are making this decision, please bear in mind the possibility of XSS attacks through the manipulation of the substitution variables.

Merchant A's account information is as follows:

Parameter	Description
Secret Key	"abcdabcdabcdabcd"
Account ID	"123412341234"
Hash Type in APIs (default hash type)	"HMAC_SHA512"

Example 1 (SHPF_TPS_HASH_TYPE = "" or was not sent)

Merchant A wants to send the following fields:

Field	Description
SHPF_FORM_ID	"formid"
SHPF_TPS_DEF	"AMOUNT SHPF_TPS_HASH_TYPE SHPF_FORM_ID SHPF_TPS_DEF"
SHPF_TPS_HASH_TYPE	""
AMOUNT	"9.99"

To calculate the SHPF_TPS, Merchant A would need to:

STEP ONE

Concatenate the values in the SHPF_TPS_DEF to create a message string. Remember, if the field isn't sent or if the value is undefined, use an empty string as that field's value.

Message	= AMOUNT + SHPF_TPS_HASH_TYPE + SHPF_FORM_ID + SHPF_TPS_DEF = "9.99" + "" + "formid" + "AMOUNT SHPF_TPS_HASH_TYPE SHPF_FORM_ID SHPF_TPS_DEF" = "9.99formidAMOUNT SHPF_TPS_HASH_TYPE SHPF_FORM_ID SHPF_TPS_DEF"

STEP TWO

Since Merchant A is sending SHPF_TPS_HASH_TYPE = "", the merchant's default hash type (in this case, 'HMAC_SHA512') must be used.

SHPF_TPS	= HMAC_SHA512(Secret Key, message) in hex format = HMAC_SHA512("abcdabcdabcdabcd", "9.99formidAMOUNT SHPF_TPS_HASH_TYPE SHPF_FORM_ID SHPF_TPS_DEF") in hex format = "38aaa15d31f57ea1a2d25e11a6ecbc6652965a006a8ac64069d936f708044725 6d23ef6ae3c43e425be522f0057ea89d12210c6c31ed168ffb670075f075d0fa"

The resulting URL may look like this:

https://secure.bluepay.com/interfaces/shpf?SHPF_FORM_ID=formid&SHPF_TPS_DEF=AMOUNT%20SHPF%5FTPS%5FHASH%5FTYPE%20SHPF%5FFORM%5FID%20SHPF%5FTPS%5FDEF&SHPF_TPS_HASH_TYPE=&AMOUNT=9%2E99&SHPFTP=38aaa15d31f57ea1a2d25e11a6ecbc6652965a006a8ac64069d936f7080447256d23ef6ae3c43e425be522f0057ea89d12210c6c31ed168ffb670075f075d0fa

Example 2 (SHPF_TPS_HASH_TYPE = "SHA256")

Merchant A wants to send the following fields:

Field	Description
SHPF_FORM_ID	"formid"
SHPF_TPS_DEF	"SHPF_TPS_HASH_TYPE CUSTOM_ID SHPF_FORM_ID SHPF_TPS_DEF"
SHPF_TPS_HASH_TYPE	"SHA256"
CUSTOM_ID	""

To calculate the SHPF_TPS, Merchant A would need to:

STEP ONE

Concatenate the values in the SHPF_TPS_DEF to create a message string. Remember, if the field isn't sent or if the value is undefined, use an empty string as that field's value.

Message	= SHPF_TPS_HASH_TYPE + CUSTOM_ID + SHPF_FORM_ID + SHPF_TPS_DEF = "SHA256" + "" + "formid" + "SHPF_TPS_HASH_TYPE CUSTOM_ID SHPF_FORM_ID SHPF_TPS_DEF" = "SHA256formidSHPF_TPS_HASH_TYPE CUSTOM_ID SHPF_FORM_ID SHPF_TPS_DEF"

STEP TWO

Since Merchant A is sending SHPF_TPS_HASH_TYPE = "SHA256":

SHPF_TPS	= sha256sum(Secret Key + message) in hex format = sha256sum("abcdabcdabcdabcd" +"SHA256formidSHPF_TPS_HASH_TYPE CUSTOM_ID SHPF_FORM_ID SHPF_TPS_DEF") in hex format = "3a01df7d8797b12b32c20566a4a3a1b827e59aa2dbc438c573827e4d4276d9f0"

The resulting URL may look like this:

https://secure.bluepay.com/interfaces/shpf?SHPF_FORM_ID=formid&SHPF_TPS_DEF=SHPF%5FTPS%5FHASH%5FTYPE%20CUSTOM%5FID%20SHPF%5FFORM%5FID%20SHPF%5FTPS%5FDEF&SHPF_TPS_HASH_TYPE=SHA256&CUSTOMID=&SHPFTPS=3a01df7d8797b12b32c20566a4a3a1b827e59aa2dbc438c573827e4d4276d9f0

Example 3 (SHPF_TPS_HASH_TYPE = "HMAC_SHA256")

Merchant A wants to send the following fields:

Field	Description
SHPF_FORM_ID	"formid"
SHPF_TPS_DEF	"SHPF_TPS_HASH_TYPE CUSTOM_ID SHPF_FORM_ID SHPF_TPS_DEF"
SHPF_TPS_HASH_TYPE	"HMAC_SHA256"
CUSTOM_ID	""

To calculate the SHPF_TPS, Merchant A would need to:

STEP ONE

Concatenate the values in the SHPF_TPS_DEF to create a message string. Remember, if the field isn't sent or if the value is undefined, use an empty string as that field's value.

Message	= SHPF_TPS_HASH_TYPE + CUSTOM_ID + SHPF_FORM_ID + SHPF_TPS_DEF = "HMAC_SHA256" + "" + "formid" + "SHPF_TPS_HASH_TYPE CUSTOM_ID SHPF_FORM_ID SHPF_TPS_DEF" = "HMAC_SHA256formidSHPF_TPS_HASH_TYPE CUSTOM_ID SHPF_FORM_ID SHPF_TPS_DEF"

STEP TWO

Since Merchant A is sending SHPF_TPS_HASH_TYPE = SHA256

SHPF_TPS	= HMAC_SHA256(Secret Key, message) in hex format = HMAC_SHA256("abcdabcdabcdabcd", "SHA256formidSHPF_TPS_HASH_TYPE CUSTOM_ID SHPF_FORM_ID SHPF_TPS_DEF") in hex format = "1cb4968eb1ae2d6f31e6aee24ffbe5ea9afc672cb7303db5415d30cb43634dfe"

The resulting URL may look like this:

https://secure.bluepay.com/interfaces/shpf?SHPF_FORM_ID=formid&SHPF_TPS_DEF=SHPF%5FTPS%5FHASH%5FTYPE%20CUSTOM%5FID%20SHPF%5FFORM%5FID%20SHPF%5FTPS%5FDEF&SHPF_TPS_HASH_TYPE=HMAC%5FSHA256&CUSTOMID=&SHPFTPS=1cb4968eb1ae2d6f31e6aee24ffbe5ea9afc672cb7303db5415d30cb43634dfe

Revision History

Version	Revision Date	Reason for Change
V1.0	May 2024	Updated the "HMAC_SHA512" in Example section and the resulting URLs
V1.1	April 2025	Updated the layout format of the document