BluePay CSV Upload Report (bpbureport)

Batch Upload Reporting API

Reference Guide

April 2025

http://www.fiserv.com

This document is classified as Fiserv Public.

Content

About this Document

This documentation provides technical guidance to instruct merchants on interacting with BluePay's Batch Upload Reporting API. The document also details the request and response formats, ensuring the integration team understands how to effectively use the API for secure and efficient batch processing of transactions.

Intended Audience

This document is intended for merchants, partners, and developers who will be responsible for integrating and managing transactions using BluePay's Batch Upload Reporting API.

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	bluepay-integration@fiserv.com

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

Overview

BluePay's Batch Upload Reporting API enables merchants to check the status of and download the results of CSV files uploaded into the BluePay Payment Gateway for batch processing. This API supports standard HTTPS POST requests and responses, providing detailed feedback on the status of each transaction in the batch. The API's cryptographic hash functions create a TAMPER_PROOF_SEAL that ensures data security and verifies the integrity and authenticity of transaction data.

Additionally, merchants can integrate, automate, and monitor transaction processing, enhancing efficiency and security using the structured request and response formats.

URL

The API endpoint is as follows:

Sample: https://secure.bluepay.com/interfaces/bpbureport

Input Parameters

The BluePay system requires the following parameters as a POST to the indicated URL.

ACCOUNT_ID	
Required:	Yes
Description:	A 12-digit BluePay Account ID associated with the merchant profile.

ACCOUNT_ID	
Required:	Yes
Description:	A 12-digit ID of the batch that the merchant wants to view.

TAMPER_PROOF_SEAL	
Required:	Yes
Description:	Hash for security, using selected algorithm (either TPS_HASH_TYPE or account's 'Hash Type in APIs' value).
Refer the TAMPER_PROOF_SEAL section for more details	

TPS_HASH_TYPE	
Required:	Optional
Description:	The algorithm used to compute the TAMPER_PROOF_SEAL. 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.

Refer the TAMPER_PROOF_SEAL section for more details

TPS_HASH_TYPE	
Required:	Optional
Description:	• Space-separated list of input field names in the order they are to be used in the calculation of the TAMPER_PROOF_SEAL. • If set as blank or not sent, it will default to: "ACCOUNT_ID BATCH_ID". • The merchant's Secret Key is always used in the calculation of the TAMPER_PROOF_SEAL, but should NOT be included in the TPS_DEF.

- The use of this field can possibly weaken your security. Please ensure to understand how the TAMPER_PROOF_SEAL works before you consider using this option.
- Refer the TAMPER_PROOF_SEAL section for more details.

Request And Response Formats

To effectively interact with BluePay's Batch Upload Reporting API,use the correct requests and responses formats.

Request Format

The expected input format uses a standard HTTPS POST. The request body contains all parameters as URI-encoded.

Example

```
### BEGIN REQUEST EXAMPLE ###

      POST https://secure.bluepay.com/interfaces/bpbureport

      Content-Type: application/x-www-form-urlencoded

      BATCH_ID=00000000001&ACCOUNT_ID=123412341234&TAMPER_PROOF_
      SEAL=fb075373242bb78d2b806811bdd7dac4

### END REQUEST EXAMPLE ###
```

Response Format

The expected output format t is a standard HTTPS response. The HTTP status is a follows:

- For a successfully processed request- '200'
- For an error - '400'

The response will include the following HTTP headers indicating how many transactions are in that state (if the number is 0, the header will be omitted)

Header	Description
X-Tx-New	Number of transactions in a queue to be processed
X-Tx-Running	Number of transactions currently being processed
X-Tx-Done	Number of transactions that were processed
X-Tx-Error	Number of transactions that returned an error during processing

- If there are still transactions waiting to be processed (for example, 'X-Tx-New' header is included), the body of the response will be the message "BATCH PROCESSING".
- Otherwise, the body of the response will contain CSV-style data, one line per transaction, in the same order as the uploaded file, with a header row containing the columns.

Columns may be added/ removed/ re-ordered/ etc., so always pay attention to the column headers.

Example

```
### BEGIN RESPONSE EXAMPLE - BATCH PROCESSING ###

      HTTP/1.1 200 OK
      Connection: close
      Date: Thu, 07 Nov 2019 20:26:00 GMT
      Server: Apache
      Content-Type: text/csv; charset=ISO-8859-1
      X-Tx-New: 5
      X-Tx-Running: 3
      X-Tx-Done: 2

BATCH PROCESSING
### END RESPONSE EXAMPLE - BATCH PROCESSING ###
```

```
### BEGIN RESPONSE EXAMPLE - BATCH DONE PROCESSING ###

      HTTP/1.1 200 OK

      Connection: close

      Date: Thu, 07 Nov 2019 20:26:00 GMT

      Server: Apache

      Content-Type: text/csv; charset=ISO-8859-1

      X-Tx-Done: 9

      X-Tx-Error: 1

      "line_num","id","payment_type","trans_type","amount","card_type","payment_
      account","order_id","invoice_id","custom_id","custom_id2","master_
      id","status","f_void","message","origin","issue_date","rebilling_
      id","card_expire","bank_name","addr1","addr2","city","state","zip",
      "phone","email","auth_code","name1","name2","company_name",
      "memo","backend_id","doc_type","avs_result","cvv_result","card_present",
      "merchdata","f_transarmor"

      ...

### END RESPONSE EXAMPLE - BATCH DONE PROCESSING ###
```

TAMPER_PROOF_SEAL

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 Payment Gateway (https://secure.bluepay.com) under "Hash Type in APIs".

Merchants may override their default by including the TPS_HASH_TYPE field in the transaction request.

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

HASH Type	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.
    - For SHA256 blocksize = 64 and for 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 TAMPER_PROOF_SEAL, 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 TAMPER_PROOF_SEAL

STEP ONE

Concatenate the values of the fields that make up the 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. When no TPS_DEF is sent ('+' represents string concatenation, and the field names represent the contents of the respective fields).

Message	= ACCOUNT_ID + BATCH_ID

STEP TWO

- If TPS_HASH_TYPE is "" or is not sent, the merchant's 'Hash Type in APIs' value will determine which hash function to use.
- If 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 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.

Example

The following parameters provide the Merchant A's account information:

Secret Key	= "abcdabcdabcdabcd"
ACCOUNT_ID	= "123412341234"
Hash Type in APIs (default hash type)	= "MD5"

If Merchant A set their TPS_DEF to "BATCH_ID ACCOUNT_ID" and wanted to view the batch with ID 100000000001, the request would include the following parameters:

TPS_DEF	= "BATCH_ID ACCOUNT_ID"
ACCOUNT_ID	= "123412341234"
BATCH_ID	= "100000000001"

To calculate the TAMPER_PROOF_SEAL, Merchant A would need to

STEP ONE

Concatenate the values in the 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	= BATCH_ID + ACCOUNT_ID = "100000000001" + "123412341234" = "10000000000112341234 1234"

STEP TWO

This step will vary depending on which TPS_HASH_TYPE is sent (if any).

- If TPS_HASH_TYPE = "" or was not sent, the merchant's default hash type must be used.

TAMPER_PROOF_SEAL	= md5sum(Secret Key + message) in hex format = md5sum("abcdabcdabcdabcd" + "10000000000112341234 1234") in hex format = "5e2e96f6d794b1d4311d73dff5162805"

- If TPS_HASH_TYPE = "SHA256"

TAMPER_PROOF_SEAL	= sha256sum(Secret Key + message) in hex format = sha256sum("abcdabcdabcdabcd" + "10000000000112341234 1234") in hex format = "b0c5c887b91632734872a59463f947890031a313f9f961 bb5121d0bafce0d693"

- If TPS_HASH_TYPE = "HMAC_SHA256"

TAMPER_PROOF_SEAL	= HMAC_SHA256(Secret Key, message) in hex format = HMAC_SHA256("abcdabcdabcdabcd", "10000000000112341234 1234") in hex format = "3824cd4e1903d12f2e08b70cac61a242d43ec0c5641052 c1a365da4bdae0514a"

Revision History

Version	Revision Date	Reason for Change
1.1	April 2025	• Updated the layout format of the document