Perfios Insights Statement Upload API

Version 3.0

February 07, 2019

Contents

1 Process Flow

This section describes the high level process flow of the Perfios Insights API for Statement Upload. Perfios Insights API can be used to automate and expedite the processing of application in the financial sector, as listed below

- Income verification as part of loan application processing
- Income eligibility as part of loan application processing
- Income document submission as part of application processing for loan, credit card, Insurance etc.

1.1 Perfios Insights Statement Upload API Listing

- Initiate Transaction
- Upload File
- Process Statement
- Re-Process Statement
- Report Generation
- Retrieve Report

1.2 How Does It Work?

- Step 1: The client starts by initiating a transaction using the `Initiate Transaction` API. This API returns an XML response containing a time-limited `perfiosTransactionId` that the client can use to upload statements. The `perfiosTransactionId` is valid for *60 minutes*.
- Step 2: The client then proceeds to upload a file using the `Upload File` API. One file can be uploaded per API call. This API returns an XML response with a file ID.

- Step 3: The client processes the uploaded file using the `Process Statement` API. This API returns the list of ALL bank accounts processed so far (i.e. cumulative) if everything goes well
- Step 4: After uploading and processing all statements the client invokes the `Report Generation` API to complete the transaction. Perfios tries to generate one or more reports (*XML*, *PDF*, *XLS*, *XLSX*, *JSON*) using the data from the uploaded statements. This API returns an XML response indicating if the process succeeded.
- Step 5: After Step 4 succeeds, the client can use the `Retrieve Report` API to retrieve the report(s) generated for this Transaction.

2 Signing Requests

This API introduces a major change in the way requests are signed.

- Clients must sign **ALL** API requests
- Server expects the signature as a request header: `X-Perfios-Signature`
- Some request headers are mandatory for **ALL** requests

HTTP Request Header	Value
`Host`	Host. E.g. `app.perfios.com`
`X-Perfios-Algorithm`	**PERFIOS-RSA-SHA256**
`X-Perfios-Content-Sha256`	Lowercase, hex encoded SHA256 sum of the XML payload (if there is no XML payload then use an empty string)
`X-Perfios-Date`	Timestamp in `YYYYMMDD'T'HHMMSS'Z'` format
`X-Perfios-Signature`	Calculated Signature
`X-Perfios-Signed-Headers`	**host;x-perfios-content-sha256;x-perfios-date**

Table 1: Mandatory request headers.

Request signature is calculated as follows.

1. Create a canonical request
2. Use the canonical request and additional metadata to create a string for signing
3. Calculate a SHA256 checksum of the string to sign
4. Encrypt the checksum using the client's **RSA Private Key** to create a signature
5. Add the resulting signature to the HTTP request in a header

2.1 Step 1. Create a canonical request

A canonical request can be created as follows.

```
CanonicalRequest = HttpRequestMethod + ”\n”
+ CanonicalUri + ”\n”
+ CanonicalQueryString + ”\n”
+ CanonicalHeaders + ”\n”
+ SignedHeaders + ”\n”
+ Lowercase(Hex(Sha256(PayloadOrEmptyString)))
```

Where:

```
CanonicalUri = UriEncode(path)
```

E.g.

- `https://app.perfios.com/KuberaVault/api/v3/organisations/acme/transactions`
  - Path is `/KuberaVault/api/v3/organisations/acme/transactions`
- `https://app.perfios.com/KuberaVault/api/v3/organisations/acme/transactions/G7BW1549868724968/reports?types=xm`
  - Path is: `/KuberaVault/api/v3/organisations/acme/transactions/G7BW1549868724968/reports`

```
CanonicalQueryString = UriEncode(<QueryParam1>) + ”=” + UriEncode(<Value1>) + ”&”
+ ... + UriEncode(<QueryParamN>) + ”=” + UriEncode(<ValueN>)     // Sorted by query parameter name
```

```
CanonicalHeaders = Lowercase(<HeaderName1>) + ”:” + Trim(<value1>) + ”\n”
+ ... + Lowercase(<HeaderNameN>) + ”:” + Trim(<valueN>)    // Sorted by header name
```

The following headers are deemed ‘canonical’, and must be used in the above calculation:

1. `Host`
2. `X-Perfios-Content-Sha256`
3. `X-Perfios-Date`

```
SignedHeaders = host;x-perfios-content-sha256;x-perfios-date  // Fixed string
```

```
PayloadOrEmptyString = Payload iff request requires XML payload, else empty string ””
```

2.2 Step 2. Use the canonical request and additional metadata to create a string for signing

1. Start with the algorithm designation, followed by a newline character. Algorithm is `PERFIOS-RSA-SHA256` (note that this is the value of the mandatory header, `X-Perfios-Algorithm`)
2. Append the request date value, followed by a newline character. (note that this request date (`RequestDate`) is the value of the mandatory header, `X-Perfios-Date` in the format `YYYYMMDD’T’HHMMSS’Z’`
3. Append the lowercase, Base 16 encoded, Sha256 hash of the canonical request created in the previous step

I.e.

```
StringToSign = ”PERFIOS-RSA-SHA256” + ”\n”
+ RequestDate in ‘YYYYMMDD’T’HHMMSS’Z’‘ format  + ”\n”
+ Lowercase(Hex(Sha256(CanonicalRequest)))
```

For e.g.:

```
StringToSign = ”PERFIOS-RSA-SHA256” + ”\n”
+ ”20190213T114306Z”  + ”\n”
+ ”87d5cd4e84282038505563542974ad5f3eafdad2ada5cfb91f8709d0fd46e497”
```

2.3 Step 3. Calculate a SHA256 checksum of the string to sign

```
Checksum = Lowercase(Hex(Sha256(StringToSign)))
```

2.4 Step 4. Encrypt the checksum using the client’s RSA Private Key to create the signature

```
Signature = Lowercase(Hex(RSA-Private-Key-Encrypted(Checksum)))
```

2.5 Step 5. Add signature to the HTTP request in a header

Add the `X-Perfios-Signature` header to the HTTP request, with the signature as its value.

3 Note on URLs

All URLs will have the following prefix:

- `https://host-name/KuberaVault/api/v3`

The `host-name` will vary depending on the target environment. E.g.

- Production Singapore Data Center (DC): `www.perfios.com`
- Production India DC: `app.perfios.com`
- UAT: `demo.perfios.com`

4 API : Initiate Transaction

4.1 URL

- `/organisations/{organisationName}/transactions`

4.2 URL Parameters

Element Name	Description
`organisationName`	Client's `vendorId` provided by Perfios

Table 2: Initiate Transaction URL Parameters.

4.3 Methods Accepted

`POST`

4.4 Content-Types Accepted

`application/xml`

4.5 Request Body

```
<?xml version="1.0" encoding="UTF-8"?>
<payload>
    <loanAmount>100000</loanAmount>
    <loanDuration>24</loanDuration>
    <loanType>Home</loanType>
    <processingType>STATEMENT</processingType>
    <transactionCompleteCallbackUrl>https://example.com/callback</transactionCompleteCallbackUrl>
    <txnId>PQ1342687YTX</txnId>
    <!-- The following are optional -->
    <acceptancePolicy>atLeastOneTransactionPerMonthInRange</acceptancePolicy>
    <institutionId>2</institutionId>
    <uploadingScannedStatements>true</uploadingScannedStatements>
    <yearMonthFrom>2015-11</yearMonthFrom>
    <yearMonthTo>2016-02</yearMonthTo>
</payload>
```

4.6 Payload Elements

Element Name	Description
`loanAmount`	Loan Amount in INR. Round to the nearest full Rupee. No decimal values.
`loanDuration`	Loan Duration in Months. Integer ranging from 1 to 1188 (99 years)
`loanType`	Loan Type (e.g. Home, Personal, Vehicle etc.). Maximum 64 characters

Element Name	Description
`processingType`	Type of documents processed in this transaction Possible values: `STATEMENT` (for processing bank statements), `FINANCIAL_STATEMENT` (for processing financial statements).
`transactionCompleteCallbackUrl`	Perfios server will send an `HTTP POST` request to this url after the transaction is completed. See the *Notes on Some Topics* section for more details. This element is mandatory if the client is uploading *scanned statements*. See next element.
`txnId`	Client-specified identifier to uniquely identify this transaction. Alpha-numeric, or dash (`-`), maximum 64 characters.
`acceptancePolicy`	(Optional) Constrain the condition under which the user's data should be accepted and report generated. Can be one of two possible values: `atLeastOneTransactionInRange`: proceed if the user has at least one transaction in the expected date range. `atLeastOneTransactionPerMonthInRange`: proceed if the user has at least one transaction per month in the expected date range.
`institutionId`	User selected institution. See `Supported Institutions` API for full list.
`uploadingScannedStatements`	(Optional) Indicates whether the client will upload scanned statements - i.e. images - rather than e-statements - i.e. text only. Possible values are `true`, and `false`. `True` indicates that the client is uploading scanned statements. If this element is omitted, it is assumed that the client will upload e-statements.
`yearMonthFrom` `yearMonthTo`	(Optional) Constrain the date range for which data should be fetched through netbanking, or uploaded by user in case of statement upload. Must be a valid year-month in `YYYY-MM` format. Maximum and minimum values are current month, and current month minus two years respectively. If one of these is given, the other must also be supplied.

Table 3: Initiate Transaction Payload.

4.7 A Note on Scanned Statements

Perfios supports two types of statements: scanned (image), and e-statements (text-only). Of the two, scanned statements are *not* immediately processed. Instead they are *accepted* for processing, and are processed later. The client will receive a call back after processing completes (successfully or otherwise).

4.8 Success Response

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction>
    <perfiosTransactionId>ZZZZ1234567890123</perfiosTransactionId>
</transaction>
```

Note: Client will need to use this `perfiosTransactionId` parameter in subsequent API calls.

4.9 Sample Error Response

See subsequent sections for a complete listing of API and Statement Error Codes.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<error>
  <code>MethodNotAllowed</code>
  <message>This API supports only POST</message>
</error>
```

4.10 Note

See the last section for detailed notes on date range, acceptance policy and transaction complete callback.

5 API : Upload File

After starting the transaction, upload files one by one using this API. Send multi-part `POST` requests with the one file per request.

5.1 URL

- `/organisations/{organisationName}/transactions/{perfiosTransactionId}/files`

5.2 URL Parameters

Element Name	Description
`organisationName`	Client’s `vendorId` provided by Perfios
`perfiosTransactionId`	`perfiosTransactionId` obtained from the *Initiate Transaction API* response

Table 4: Upload File URL Parameters.

5.3 Methods Accepted

`POST`

5.4 Content-Types Accepted

`multipart/form-data`

5.5 Request Structure

This API expects a multi-part request, with the contents of one file per call

5.6 Success Response: E-statements

- HTTP Status code: 200 OK

```
<?xml version="1.0" encoding="UTF-8"?>
<file>
    <fileId>b09051b4ab2859b4ea66cc1a0128fa684e6c49f3</fileId>
</file>
```

5.7 Success Response: Scanned Statements

- HTTP Status code: 202 Accepted

```
<?xml version="1.0" encoding="UTF-8"?>
<file>
    <fileId>b09051b4ab2859b4ea66cc1a0128fa684e6c49f3</fileId>>
</file>
```

5.8 Sample Error Response

```
<?xml version="1.0" encoding="UTF-8"?>
<error>
  <code>MethodNotAllowed</code>
  <message>This API supports only POST</message>
</error>
```

6 API : Process Statement

The Client can try to process an uploaded file as a statement using this API.

6.1 URL

- `/organisations/{organisationName}/transactions/{perfiosTransactionId}/bank-statements`

6.2 URL Parameters

Element Name	Description
`organisationName`	Client's `vendorId` provided by Perfios
`perfiosTransactionId`	`perfiosTransactionId` obtained from the *Initiate Transaction API* response

Table 5: Process Statement URL Parameters.

6.3 Methods Accepted

`POST`

6.4 Content-Types Accepted

`application/xml`

6.5 Payload Elements

Element Name	Description
`fileId`	File identifier obtained on uploading the file.
`institutionId`	(Optional) Identify the institution that issued this bank statement
`password`	(Optional) Password to open the statement, if applicable

Table 6: Process Statement Payload.

6.6 Request Body

```
<?xml version="1.0" encoding="UTF-8"?>
<payload>
    <fileId>b09051b4ab2859b4ea66cc1a0128fa684e6c49f3</fileId>
    <!-- The following are optional -->
    <institutionId>22</institutionId>
    <password>topSecret!</password>
</payload>
```

6.7 Success Response: E-statements

- HTTP Status code: 200 OK

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<bankStatement>
  <bankAccounts>
    <bankAccount>
      <accountId>fec4b3f2fd9fec90e03de0f564a34c37417df2af</accountId>
      <accountNumber>XXXXXXXXXXXX0297</accountNumber>
      <accountType/>
      <complete>true</complete>
      <institutionId>30</institutionId>
      <missingMonths/>
    </bankAccount>
  </bankAccounts>
</bankStatement>
```

6.8 Success Response: Scanned Statements

- HTTP Status code: 202 Accepted

6.9 Sample Error Response

```
<?xml version=”1.0” encoding=”UTF-8”?>
<error>
  <code>MethodNotAllowed</code>
  <message>This API supports only POST</message>
</error>
```

7 API : Re-Process Statement

Sometimes the Client may need to re-process a statement. E.g. if Perfios detects that the statement requires a password, then the (error) response from the Process API will indicate as much. The Client can call the Re-Process API and supply the password to process the statement again.

7.1 URL

- `/organisations/{organisationName}/transactions/{perfiosTransactionId}/bank-statements/{statementFileId}`

7.2 URL Parameters

Element Name	Description
`organisationName`	Client’s `vendorId` provided by Perfios
`perfiosTransactionId`	`perfiosTransactionId` obtained from the *Initiate Transaction API* response
`statementFileId`	`fileId` obtained from the *Upload File API* response

Element Name	Description

Table 7: Re-Process Statement URL Parameters.

7.3 Methods Accepted

```
POST
```

7.4 Content-Types Accepted

```
application/xml
```

7.5 Request Body

```
<?xml version="1.0" encoding="UTF-8"?>
<payload>
    <password>topSecret!</password>
</payload>
```

7.6 Success Response: E-statements

- HTTP Status code: 200 OK

```
<?xml version="1.0" encoding="UTF-8"?>
<bankStatement>
  <bankAccounts>
    <bankAccount>
      <accountId>fec4b3f2fd9fec90e03de0f564a34c37417df2af</accountId>
      <accountNumber>XXXXXXXXXXXX0297</accountNumber>
      <accountType/>
      <complete>true</complete>
      <institutionId>30</institutionId>
      <missingMonths/>
    </bankAccount>
  </bankAccounts>
</bankStatement>
```

7.7 Success Response: Scanned Statements

- HTTP Status code: 202 Accepted

7.8 Sample Error Response

```
<?xml version="1.0" encoding="UTF-8"?>
<error>
  <code>MethodNotAllowed</code>
  <message>This API supports only POST</message>
</error>
```

8 API : Report Generation

After processing sufficient files the client can try generate reports for this transaction.

8.1 URL

- `/organisations/{organisationName}/transactions/{perfiosTransactionId}/reports`

8.2 URL Parameters

Element Name	Description
`organisationName`	Client's `vendorId` provided by Perfios
`perfiosTransactionId`	`perfiosTransactionId` obtained from the *Initiate Transaction API* response

Table 8: Report Generation URL Parameters.

8.3 Methods Accepted

`POST`

8.4 Content-Types Accepted

`application/xml`

8.5 Success Response

```
<?xml version="1.0" encoding="UTF-8"?>
<success/>
```

9 API : Retrieve Reports

After processing sufficient files the client can try generate reports for this transaction.

9.1 URL

- `/organisations/{organisationName}/transactions/{perfiosTransactionId}/reports?types={comma-separated-report-ty`

9.2 URL Parameters

Element Name	Description
`organisationName`	Client's `vendorId` provided by Perfios
`perfiosTransactionId`	`perfiosTransactionId` obtained from the *Initiate Transaction API* response
`types`	Comma separated types of reports to retrieve. • E.g. `?types=xml`: to retrieve XML report • E.g. `?types=xml,pdf`: to retrieve XML, and PDF reports (as a zip file) • E.g. `?types=xls,json`: to retrieve XLS, and JSON reports (as a zip file)

Table 9: Retrieve Reports URL Parameters.

9.3 Methods Accepted

`GET`

9.4 Content-Types Accepted

`application/xml`