ChatGPT

PEFINDO New Report API Documentation

PEFINDO's **New Report** API is a RESTful web service that returns credit report data in JSON format (lighter and smaller than the previous XML-based "Old Report" service) [1] [2]. The New Report API consists of several endpoints for authentication, searching debtor data, generating reports, and retrieving the results (including options for large reports and PDF files). All responses are JSON formatted with a standard structure: a **code** (string), **status** (`"success"` or `"failed"`), and **message** describing the outcome. A **token** is used for authentication after login. Below are the complete specifications for each endpoint in the New Report API (skip Old Report):

Authentication and Token Management

GET `/api/v1/getToken` – Get Token

- **Function:** Obtain an access token using member credentials (username/password) [3].
- **Endpoint:** `https://{domain}/api/v1/getToken` [4] (replace `{domain}` with the provided domain).
- **Method:** GET [4].
- **Authentication:** Basic Auth – provide credentials as `Authorization: Basic <base64(username:password)>` [5]. (Precondition: valid username/password and whitelisted IP address) [5].
- **Headers:** `Content-Type: application/json` and `Authorization: Basic <credentials>` [6]. (No request body for this GET request.)

Example Request:

```
curl -X GET "https://{domain}/api/v1/getToken" \
  -H "Content-Type: application/json" \
  -H "Authorization: Basic cGJrX3VzZXI6UEBzc3cwcmQxMjM0"
```

(In the above example, `Authorization: Basic cGJrX3VzZXI6UEBzc3cwcmQxMjM0` *is the Base64 encoding of* `pbk_user:P@ssw0rd1234`*.)* [7]

- **Success Response (200):** Returns a JSON object with a new JWT token. Fields:
- `code`: `"01"` (indicates request was successful) [8]
- `status`: `"success"` [8]
- `message`: `"token aktif"` (means "token is active") [9]
- `token`: the issued JWT token string [10].

Example Success Response:

```
{
  "code": "01",
  "status": "success",
  "message": "token aktif",
  "token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIi... (JWT token
string)"
}
```

The `token` *value is a long JWT string (omitted here for brevity) that the client must use in subsequent requests.* [8] [10]

- **Error Responses:**
- **Invalid Credentials (403):** If username or password is incorrect, returns HTTP 403 with `code`: **"13"**, `status`: **"failed"**, `message`: **"username atau password salah"** (incorrect username or password) [11].
- **Unauthorized IP (403):** If the caller's IP is not whitelisted, returns HTTP 403 with `code`: **"17"**, `status`: **"failed"**, `message`: **"akses ditolak"** ("access denied") [12].
- **Other Errors (500):** Any other internal server error returns HTTP 500 with `code`: **"99"**, `status`: **"failed"**, and `message` containing the system error message [13]. *(Code "99" is a generic error code used for unspecified internal errors in all endpoints [14].)*

Example Error Response (invalid login):

```
json
  {
    "code": "13",
    "status": "failed",
    "message": "username atau password salah"
  }
```
[15]

GET `/api/v1/validateToken` – Validate Token

- **Function:** Validate the token's validity and get its expiration info (checks if a token is active) [16].
- **Endpoint:** `http://{domain}/api/v1/validateToken` [17].
- **Method:** GET [17].
- **Authentication:** Bearer Token – supply the JWT obtained from `/getToken` as `Authorization: Bearer <token>` [18].
- **Headers:** `Content-Type: application/json`, `Authorization: Bearer <token>` [19].
- **Request Body:** *None.* (Token is provided in header.)

Example Request:

```
curl -X GET "http://{domain}/api/v1/validateToken" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer <your_jwt_token>"
```
[19]

- **Success Response (200):** If the token is valid, returns:
  - `code`: **"01"** (success) [20]
  - `status`: **"success"** [20]
  - `message`: **"authorized"** (the token is active/authorized) [21].

  Example Success Response:
  ```json
  {
    "code": "01",
    "status": "success",
    "message": "authorized"
  }
  ``` [22] [23]

- **Error Response (403):** If the token is invalid or expired, returns HTTP 403 with:
  - `code`: **"06"**, `status`: **"failed"**, `message`: **"Invalid Token"** [24] [25].

  Example Error Response:
  ```json
  {
    "code": "06",
    "status": "failed",
    "message": "Invalid Token"
  }
  ``` [25]

(Note: Ensure to call `/getToken` to obtain a token before calling this, and include the token in the Bearer header. Each token's validity and expiration can be checked via this endpoint.)

Debtor Search

POST `/api/v1/product/search` – **Search Debtor Data**

- **Function:** Search for debtor information based on provided personal or corporate data and product type [26]. This initiates a credit inquiry for a debtor.
- **Endpoint:** `http://{domain}/api/v1/product/search` [27].
- **Method:** POST [28].
- **Authentication:** Bearer Token (requires a valid token from `/getToken`) [29].
- **Headers:** `Content-Type: application/json`, `Authorization: Bearer <token>` [29]. (IP must be whitelisted as well, same precondition as above endpoints.)
- **Request Body:** JSON object with the following fields:
  - `type` – **(String, required)**: Debtor type, either `"PERSONAL"` or

`"CORPORATE"` [30].
  - `product_id` – **(Number, required)**: Product identifier code for the type of report or service requested [31] (see product ID reference table in documentation).
  - `inquiry_reason` – **(Number, required)**: Inquiry reason code (purpose of inquiry) [32] (see inquiry reason reference table in documentation).
  - `reference_code` – **(String, optional)**: A custom reference code from the client (if needed for tracking) [33].
  - `params` – **(Array of Object, required)**: An array containing one or more parameter objects, each with identifying information about the debtor. The exact parameters required inside each object depend on the `product_id` chosen [34] [35]. For example, for a credit score product, a parameter object may include:
    - `id_type` (e.g. `"KTP"`, `"NPWP"`, `"PHONE"`, `"PASSPORT"`, etc. – type of identifier) [36].
    - `id_no` (the identification number corresponding to the `id_type`) [37].
    - `name` (full name of the individual, if personal) [38].
    - `date_of_birth` (YYYY-MM-DD format, if personal) [39].
    - `report_date` (YYYY-MM-DD, optional, used for backdated reports – if provided, the report will include data up to that date in the past) [40] [41].
    (For a corporate search, the parameter object may differ, e.g., company name, establishment date, etc., as required by the specific product.)

Example Request Body: Searching for a personal debtor by national ID (KTP) for a certain product, with a backdate:

```json
{
  "type": "PERSONAL",
  "product_id": 1,
  "inquiry_reason": 1,
  "reference_code": "ABC123",
  "params": [
    {
      "id_type": "KTP",
      "id_no": "3101110967000498",
      "name": "JOHN DOE",
      "date_of_birth": "1967-09-11",
      "report_date": "2024-03-06"
    }
  ]
}
```

Example Request (cURL):

```
curl -X POST "http://{domain}/api/v1/product/search" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer <your_token>" \
  -d '{
        "type": "PERSONAL",
        "product_id": 1,
        "inquiry_reason": 1,
        "reference_code": "ABC123",
        "params": [ { "id_type": "KTP", "id_no": "3101110967000498", "name":
"JOHN DOE", "date_of_birth": "1967-09-11" } ]
      }'
```

- **Success Response (200):** If data is found matching the query, returns a JSON with the search results. Fields include:
- `code`: **"01"** (success) [42]
- `status`: **"success"** [43]
- `message`: **"Data ditemukan"** (data found) [44]
- `inquiry_id`: *(Number)* An inquiry identifier for this search request [45]. This ID will be needed for generating the report.
- `data`: *(Array)* List of debtor records that matched the search criteria [45]. Each element in the array is an object containing debtor details and a similarity score. Common fields in each data object include:
  - `similarity_score` (Number): A score (percentage) indicating match confidence [46] [47].
  - `id_pefindo` (Number): Pefindo's internal unique ID for the debtor [48].
  - `id_type` (String): Identifier type (e.g., KTP) used in the match [48].
  - `id_no` (String): The identifier number that was matched [49].
  - `id_tipe_debitur` (String): Debtor type, e.g. `"PERSONAL"` or `"CORPORATE"` [50].
  - Other personal or company details: e.g. `nama_debitur` (debtor name), `tanggal_lahir` (DOB), `npwp` (tax ID), `alamat` (address), `nama_gadis_ibu_kandung` (mother's maiden name), etc., if available [51] [52]. Corporate fields such as `tgl_pendirian` (establishment date) or `tempat_pendirian` (place of establishment) may appear for corporate searches (in the example below these are `null` for personal debtor) [53] [54]. There is also a `response_status` field indicating data quality (e.g., `"ALL_CORRECT"`) [55].

Example Success Response:

```
{
  "code": "01",
  "status": "success",
  "message": "Data ditemukan",
  "inquiry_id": 318,
  "data": [
    {
      "similarity_score": 100.00,
      "id_pefindo": 101110967000498,
```

```
      "id_type": "KTP",
      "id_no": "3101110967000498",
      "id_tipe_debitur": "PERSONAL",
      "nama_debitur": "INDIVIDU 101110967000498",
      "tanggal_lahir": "1967-09-11",
      "npwp": "101110967000498",
      "tgl_pendirian": null,
      "tempat_pendirian": null,
      "alamat": "ALAMAT 101110967000498",
      "nama_gadis_ibu_kandung": "IBU 101110967000498",
      "tempat_lahir": null,
      "jenis_kelamin": -1385635274,
      "kode_kota": "89",
      "kode_pos": null,
      "is_current": 1,
      "response_status": "ALL_CORRECT"
    },
    { ... possible additional matches ... }
  ],
  "response_status": "ALL_CORRECT"
}
```

(In this example, two matching records are returned with identical personal details except for `id_pefindo` *and slight variations in name, illustrating multiple debtor records found for the given criteria.)* [56] [57]

- **Error Response (404):** If no data is found for the given query, returns HTTP 404 with:
- `code`: **"31"**, `status`: **"failed"**, `message`: **"Data tidak ditemukan"** (data not found) [58] [59].

Example Not Found Response:

```json
{
  "code": "31",
  "status": "failed",
  "message": "Data tidak ditemukan"
}
```
[59]

- **Other Errors:** If the request is malformed or an internal error occurs, a `code` **"99"** (HTTP 500) may be returned with a failure status and system message [13].

Note: The `inquiry_id` from a successful search response is required for the next step (report generation). The `data` array provides one or more potential debtor records; the client may choose one or multiple records from this list to generate a report. If multiple records are selected, the system will merge data from those records into a single consolidated report [60] [61].

Report Generation

POST `/api/v1/product/generateReport` – Generate Report

- **Function:** Initiate the creation of a credit report for a debtor (or merge of multiple debtor records) that was found via the search. This request tells the system to start compiling the full credit report for the selected debtor(s) [62] [60]. The response will indicate if the report generation process has started successfully.
- **Endpoint:** `http://{domain}/api/v1/product/generateReport` [63] [64].
- **Method:** POST [65].
- **Authentication:** Bearer Token (the token from `/getToken`) [66].
- **Headers:** `Content-Type: application/json`, `Authorization: Bearer <token>`.
- **Precondition:** Must have performed a `/search` and obtained an `inquiry_id`, and have a valid token and whitelisted IP [66].
- **Request Body:** JSON with the following fields:
- `inquiry_id` – **(Number, required)**: The inquiry ID received from the `/search` response for which the report is to be generated [67].
- `ids` – **(Array<Object>, required)**: An array of one or more objects, each identifying a debtor record to include in the report. (Allows merging multiple records if needed [61].) Each object should contain:
    - `id_type` **(String)** – The ID type of the selected record (e.g., "KTP", "NPWP", etc.), as provided in the `/search` result [68].
    - `id_no` **(String)** – The ID number of the selected debtor record [69].
    - `id_pefindo` **(Number)** – The Pefindo internal ID of the debtor record [70].
    (These values should come directly from the chosen entry/entries in the `/search` response's `data` array.)
- `event_id` – **(String, required)**: A UUID provided by the client to tag this report generation request [71]. This **must be unique**; if an `event_id` is reused, the API will reject the request (see error code 35 below). The same `event_id` will be used to retrieve the report via `/getReport`.
- `generate_pdf` – **(String, optional)**: `"1"` or `"0"` flag to indicate if a PDF version of the report should also be generated on the server [72]. Use `"1"` for yes (generate PDF), or `"0"` (or omit) for no. If `"1"`, the generated PDF can later be downloaded via `/downloadPdfReport`.
- `language` – **(String, optional)**: Language code for the PDF report, if `generate_pdf = "1"`. `"01"` for Bahasa Indonesia, or `"02"` for English [73]. If not provided, a default may be used (likely Indonesian).

Example Request Body: Generating a report for one PERSONAL debtor record (with a PDF in Indonesian):

```
{
  "inquiry_id": 243,
  "ids": [
    {
      "id_type": "KTP",
      "id_no": "3101110967000498",
      "id_pefindo": 101110967000498
    }
  ],
```

```
  "event_id": "451bd8bd-19dd-4605-bd05-a92971892a47",
  "generate_pdf": "1",
  "language": "01"
}
```

Example Request (cURL):

```
curl -X POST "http://{domain}/api/v1/product/generateReport" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer <your_token>" \
  -d '{
        "inquiry_id": 243,
        "ids": [ { "id_type": "KTP", "id_no": "3101110967000498", "id_pefindo":
101110967000498 } ],
        "event_id": "451bd8bd-19dd-4605-bd05-a92971892a47",
        "generate_pdf": "1",
        "language": "01"
      }'
```

- **Success Response (200):** A successful request means the report generation has begun. The response confirms acceptance and echoes the `event_id`. Fields:
- `code`: **"01"** (success) [74]
- `status`: **"success"** [74]
- `event_id`: *(String)* The UUID provided by the client, returned back for reference [75].
- `message`: *(String)* A message indicating the report is being processed, e.g., **"Proses membuat report sedang dikerjakan"** (which means "The process of creating the report is being carried out") [76]. This implies the report compilation is in progress.

Example Success Response:

```
json
  {
    "code": "01",
    "status": "success",
    "event_id": "451bd8bd-19dd-4605-bd05-a92971892a47",
    "message": "Proses membuat report sedang dikerjakan"
  }
```
[77]

Note: A code `"01"` here indicates the request was accepted. The report itself is not yet available at this stage (just processing). The client should proceed to call `/getReport` using the same `event_id` to check if the report is ready (or to retrieve it once ready).

- **Error Responses:**
- **Event ID Conflict (409):** If the provided `event_id` has already been used in a previous request, the API returns HTTP 409 Conflict with `code`: **"35"**, `status`: **"failed"**, and message **"event_id sudah ada, gunakan yang lain"** ("event_id already exists, use another") [78] [79]. This means the client must use a new unique UUID.

- **Data/Inquiry Not Found (404):** If the `inquiry_id` is invalid or the referenced search results are not found (or perhaps if the selected IDs do not match the inquiry), the API may return a 404 with `code "31" (Data tidak ditemukan)`. (This case is implied; the documentation explicitly covers code 31 for search and getReport, but any missing data could yield code 31.)
- **Other Errors (500):** Internal errors yield `code "99"` as mentioned before.

Example Error Response (event_id already used):

```
json
  {
    "code": "35",
    "status": "failed",
    "event_id": "451bd8bd-19dd-4605-bd05-a92971892a47",
    "message": "event_id sudah ada, gunakan yang lain"
  }
```
[80]

Report Retrieval

Once a report generation is initiated via `/generateReport`, the next step is to retrieve the report's content. There are two pathways to get the report data, depending on the report size:

1. **Standard retrieval via `/getReport`:** If the report is small enough (under certain internal size limits), the full JSON report will be returned directly by `/getReport`.
2. **Big report retrieval via `/downloadReport`:** If the report is very large (e.g., contains an extremely high number of credit facilities), the `/getReport` response will indicate a "big report" and not include all details, in which case the client must call `/downloadReport` to get the full JSON content.
3. Optionally, if a PDF was requested, it can be downloaded via `/downloadPdfReport` after the report is ready.

Each is detailed below. All these endpoints require the `event_id` that was used in `/generateReport`.

GET `/api/v1/product/getReport/event_id/{eventId}` – Get Report

- **Function:** Retrieve the generated report's status and data by event ID. This endpoint returns the credit report data if ready, or a status if still processing or if the report is too large to return directly [81] [82].
- **Endpoint:** `http://{domain}/api/v1/product/getReport/event_id/{eventId}` [83] (the `{eventId}` path parameter is the UUID used in the generateReport call).
- **Method:** GET [84].
- **Authentication:** Bearer Token [82].
- **Headers:** `Authorization: Bearer <token>` (and `Content-Type: application/json` if needed).
- **Path Parameter:** `event_id` – **(String, required)** – the UUID of the report request (provided by client in `/generateReport` and also returned in that response) [85]. This identifies which report to retrieve.

Example Request:

```
curl -X GET "http://{domain}/api/v1/product/getReport/event_id/
451bd8bd-19dd-4605-bd05-a92971892a47" \
  -H "Authorization: Bearer <your_token>"
```

- **Success Response (200):** There are **three possible success scenarios** indicated by the `code` in the response:
  1. **Report Ready (code "01"):** The report has been generated and is returned in the response.
  2. **Big Report (code "36"):** The report is generated but categorized as “big” – detailed data is not included in this response and must be fetched via `/downloadReport`.
  3. **Report In Progress (code "32"):** The report is still being processed (not yet ready).

  In all cases, HTTP status is 200 and `status` is `"success"` (even for code 32 and 36, which are non-standard success codes indicating special conditions).

  - **(a) Report Ready – `code: "01"`:** If the report is completed and not too large, the full report data is included in the response. Fields:
    - `code`: **"01"**
    - `status`: **"success"`**
    - `message`: e.g. **"Laporan berhasil dibuat"** (“Report successfully created”).
    - `event_id`: the same event ID string.
    - `report`: a JSON object containing the **complete report data**. This object has a complex structure with multiple sections. Key sections include:
      - **`header`**: Metadata about the report request (e.g. username, report ID, inquiry reason code, request time, etc.).
      - **`debitur`**: Debtor information summary (e.g. identity details, totals of facilities, counts of reporters, etc.). Personal or corporate details like name, addresses, phone, etc., and summary statistics such as `jml_fasilitas` (number of credit facilities), `jml_tunggakan` (number of delinquencies), `jml_plafon` (total credit limit), etc., appear here.
      - **Credit Facilities and Other Sections:** The report will include detailed lists of credit facilities (loans, credit cards, etc.), collateral, guarantees, and possibly a credit score or rating. These might be nested under sections like a list of facilities and a summary of overall credit status. (For example, a PEFINDO credit score section might include fields like `risk_grade` and `risk_desc` for the debtor’s risk grading.) In the JSON example from the documentation, a large number of fields related to various facility counts and statuses are present under `debitur` and potentially under separate sections for facilities.

      Example (Partial) Success Response (code 01):
      ```json
      ```

```
      {
        "code": "01",
        "status": "success",
        "message": "Laporan berhasil dibuat",
        "event_id": "451bd8bd-19dd-4605-bd05-a92971892a47",
        "report": {
          "header": { ... },
          "debitur": { ... },
          "facilities": [ ... ],
          "collaterals": [ ... ],
          "pefindo_score": { ... },
          ...
        }
      }
      ```
      *In the above structure, the `report` object contains numerous fields and nested objects/arrays. Refer to **Lampiran 2 – Response Data JSON New Report** in the documentation for a complete list of fields and their meanings [100] [101].* If the report data size is below the threshold defined by the system, all sections will be present here in the response [102].

  - **(b) Big Report – `code: "36"`:** If the report is **too large** to include inline (the number of credit facilities exceeds a certain system-defined limit) [103] [104], the response will have:
    - `code`: **"36"`**, `status`: **"success"** [88]
    - `message`: e.g. **"Kategori big report, detail fasilitas tidak dikirim, silahkan download json object dari report ini menggunakan method downloadReport"** [88]. (This translates to: "Big report category, facility details are not sent; please download the JSON object of this report using the downloadReport method.")
    - `event_id`: the event ID (same as requested) [105]
    - `report`: a JSON object **with partial data** – typically it will contain the `header` and `debitur` sections (and possibly some high-level summary), but **detailed facility lists are omitted** [106] [107]. The client must subsequently call the `/downloadReport` endpoint to fetch the complete report data.

    Example Big Report Response (code 36):
    ```json
    {
      "code": "36",
      "status": "success",
      "message": "Kategori big report, detail fasilitas tidak dikirim, silahkan download json object dari report ini menggunakan method downloadReport",
      "event_id": "451bd8bd-19dd-4605-bd05-a92971892a47",
      "report": {
        "header": { ... },
        "debitur": { ... }
        // detailed facilities not included
```

```
      }
    }
    ``` 88 107

    After receiving this, the client should call `/downloadReport/event_id/{eventId}` to get the full details (see **Download Report** below).

  - **(c) Report In Progress – `code: "32"`:** If the report generation is still underway (not yet finished), the response will indicate this status:
    - `code`: **"32"`**, `status`: **"success"** 87
    - `message`: **"Laporan masih dalam proses scoring"** (meaning "Report is still in the scoring process") 108.
    - No `report` field will be present in this case. This informs the client that the report is not ready yet, and they should retry `getReport` after a short delay.

    **Example In-Progress Response (code 32):**
    ```json
    {
      "code": "32",
      "status": "success",
      "message": "Laporan masih dalam proses scoring"
    }
    ``` 87 109

    The client may poll `/getReport` periodically until a final status (01 or 36) is obtained.

- **Error Responses:**
  - **Invalid Event ID (404):** If the `event_id` is not found (e.g., unknown or does not exist on the server, or possibly if it has expired), the response is HTTP 404 with `code`: **"31"**, `status`: **"failed"**, `message`: **"Data tidak ditemukan"** 110 111. This indicates the report corresponding to that `event_id` does not exist or cannot be retrieved. (Ensure the correct event_id is used and that `/generateReport` was called successfully.)
  - **Authorization Errors (403):** If the token is missing or invalid, a 403 error (e.g., code "06" as in validateToken) would be returned (similar to other endpoints).
  - **Other (500):** Code "99" for unexpected server errors.

### **GET** `/api/v1/product/downloadReport/event_id/{eventId}` – **Download Report**

- **Function:** Download the full report data as a JSON object for **big reports** 112 113. This endpoint is only needed if `/getReport` returned code 36 (indicating the report was too large to include in the `/getReport` response). It will provide the complete report content (which can be a large JSON).
- **Endpoint:** `http://{domain}/api/v1/product/downloadReport/event_id/

```
{eventId}` (plus optional pagination parameters) [114].
- **Method:** GET [115].
- **Authentication:** Bearer Token [113].
- **Headers:** `Authorization: Bearer <token>`.
- **Path Parameter:** `event_id` – The same UUID used in `/generateReport` (and
in `/getReport`) to identify the report [116].
- **Optional Parameters:** This endpoint supports pagination of the report data
if needed. In the example cURL, the URL included `/page/0/max/1` [117]. This
suggests that the client can specify a page index and a max page (or page size)
for retrieving the report in chunks. By default, if not using pagination, the
entire report JSON is returned in one response. (The `page`/`max` segments are
not documented in detail, but likely allow the JSON to be fetched in parts if
extremely large.)

Example Request:

```bash
curl -X GET "http://{domain}/api/v1/product/downloadReport/event_id/
451bd8bd-19dd-4605-bd05-a92971892a47" \
  -H "Authorization: Bearer <your_token>"
```

(Replace `{eventId}` *with the actual UUID. Optionally, add* `/page/{n}/max/{m}` *if the report needs to be fetched in segments.)*

- **Success Response (200):** Returns the full report JSON content in the same structure as described for `/getReport` code 01 success. Essentially, calling this is equivalent to getting the `report` object that would normally be embedded in `/getReport` response. The response format will include the standard wrapper (`code`, `status`, etc.) or it might directly stream the JSON object. According to the documentation, it's described as *"download report dalam bentuk json object"* – meaning the response is a JSON object of the report [112]. Likely, the API wraps it similarly with `code: "01"` and `status: "success"` for consistency, along with the report data. (The documentation does not explicitly show an example of the success JSON for downloadReport, but it can be inferred to mirror the structure of the normal getReport success, containing the `report` object.)

In practice: On success, expect a JSON response with `code: "01"`, `status: "success"`, and `event_id`, and the full `report` content included. This JSON could be very large (potentially megabytes of data if many facilities). If pagination parameters are used, you would get the portion corresponding to the requested page.

- **Error Responses:**
- **Data not found (404):** If the `event_id` is not found or no big-report data is available for it (for example, if the report was not actually categorized as big or is not yet ready), the response will be 404 with `code`: **"31"**, `status`: **"failed"**, `message`: **"Data tidak ditemukan"** [118] [110]. This could happen if this endpoint is called before the report is actually generated, or with a wrong ID.
- **Not Ready (32):** It's uncommon to call downloadReport if the report isn't ready, since you typically call it after receiving code 36. However, if invoked too early, it might also return a code "32" or a 404 until the report is ready.
```

- **Auth Errors (403)** and **Other (500)**: Similar to other endpoints (invalid token -> 403, etc., internal errors -> code 99).

Note: Use this endpoint only after `/getReport` indicates code "36". The `downloadReport` may also support parameters `page` and `max` to manage extremely large data volumes by chunks (e.g., `/downloadReport/event_id/{id}/page/0/max/1` as shown in the example) [117].

GET `/api/v1/product/downloadPdfReport/event_id/{eventId}` – Download PDF Report

- **Function:** Download the report in PDF format, if it was requested during the generate step [119]. This provides the credit report as a PDF file.
- **Endpoint:** `http://{domain}/api/v1/product/downloadPdfReport/event_id/{eventId}` [120].
- **Method:** GET [120].
- **Authentication:** Bearer Token [121].
- **Headers:** `Authorization: Bearer <token>`.
- **Path Parameter:** `event_id` – The UUID of the report request (same one used previously) [122].
- **Precondition:** A PDF must have been generated on the server. This means when calling `/generateReport`, the `generate_pdf` field was set to "1". If no PDF was generated, this call will result in a not-found error.

Example Request:

```
curl -X GET "http://{domain}/api/v1/product/downloadPdfReport/event_id/
451bd8bd-19dd-4605-bd05-a92971892a47" \
  -H "Authorization: Bearer <your_token>"
```[123]

```
- **Success Response (200):** On success, this endpoint returns the PDF file
contents in the HTTP response. The content type is `application/pdf`. Unlike
other endpoints, the response is **binary PDF data** (not JSON). The PDF would
contain the credit report in a human-readable format (in the language specified
if `language` was provided). Typically, the response will include headers like
`Content-Type: application/pdf` and possibly `Content-Disposition` to indicate
a file download. *(The documentation does not show a JSON response for success
because it returns a file.)*

- **Error Responses:**
  - **Data not found (404):** If the PDF is not available (for example,
`generate_pdf` was not requested, or the report generation is not completed, or
an incorrect event_id is used), the response is 404 with a JSON body: `code`:
**"31"**, `status`: **"failed"**, `message`: **"Data tidak ditemukan"`[124][125].
Essentially, if the service cannot find a prepared PDF for that event_id, it
treats it as not found.

  **Example Error Response (PDF not available):**
```

```json
{
  "code": "31",
  "status": "failed",
  "message": "Data tidak ditemukan"
}
``` [124]

- **Auth/Other Errors:** If the token is invalid or other errors occur, similar codes 06 (403) or 99 (500) could be returned in JSON as with other endpoints.

Notes:
- Ensure to call this only after the report generation is complete (i.e., after a successful `/getReport` with code 01 or 36). The PDF generation may slightly lag behind the JSON report generation. If you get a 404 on the PDF but the JSON report was ready, it may mean the PDF is still being generated; you might need to retry after a short delay.
- The PDF will reflect the same data as the JSON report, in the requested language (01 for Indonesian or 02 for English) [73].

General Error Handling: All endpoints can return a generic error with `code: "99"` for internal server errors or unexpected conditions [13]. The `message` in that case may contain a system-generated description of the error. Clients should handle HTTP 500 responses accordingly. Always check the `code` field in JSON responses: `"01"` indicates success (for the final outcome of a process), other codes indicate either interim statuses (e.g., "32", "36") or failures (codes >= "13" for various error conditions as described above).

Reference: For detailed field definitions of the report JSON structure (all possible keys and their meanings), see **Lampiran 2 – Response Data JSON New Report** in the PEFINDO documentation [100]. This includes comprehensive data dictionaries for sections like facilities, payments, collateral, etc., which are beyond the scope of this summary but are available for developers needing to parse and interpret the full report content.

JSON Representation of API Endpoints

Below is a developer-friendly JSON representation of the PEFINDO New Report API endpoints and their specifications (similar to a Postman collection). This JSON outlines each endpoint's method, URL pattern, required authentication, request parameters, and example responses:

```json
```

```
{
  "base_url": "https://{domain}/api/v1",
  "endpoints": [
    {
      "name": "Get Token",
      "method": "GET",
      "path": "/getToken",
      "authentication": {
        "type": "Basic",
        "credentials_format": "Base64(username:password)"
      },
      "headers": {
        "Content-Type": "application/json",
        "Authorization": "Basic <base64-encoded-credentials>"
      },
      "body": null,
      "success_response": {
        "status_code": 200,
        "body": {
          "code": "01",
          "status": "success",
          "message": "token aktif",
          "token": "<JWT access token string>"
        }
      },
      "error_responses": [
        {
          "status_code": 403,
          "body": {
            "code": "13",
            "status": "failed",
            "message": "username atau password salah"
          }
        },
        {
          "status_code": 403,
          "body": {
            "code": "17",
            "status": "failed",
            "message": "akses ditolak"
          }
        },
        {
          "status_code": 500,
          "body": {
            "code": "99",
            "status": "failed",
            "message": "[<system error message>]"
```

```
        }
      }
    ]
  },
  {
    "name": "Validate Token",
    "method": "GET",
    "path": "/validateToken",
    "authentication": {
      "type": "Bearer",
      "token_required": true
    },
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    },
    "body": null,
    "success_response": {
      "status_code": 200,
      "body": {
        "code": "01",
        "status": "success",
        "message": "authorized"
      }
    },
    "error_responses": [
      {
        "status_code": 403,
        "body": {
          "code": "06",
          "status": "failed",
          "message": "Invalid Token"
        }
      },
      {
        "status_code": 500,
        "body": {
          "code": "99",
          "status": "failed",
          "message": "[<system error message>]"
        }
      }
    ]
  },
  {
    "name": "Search Debtor",
    "method": "POST",
    "path": "/product/search",
```

```
      "authentication": {
        "type": "Bearer",
        "token_required": true
      },
      "headers": {
        "Content-Type": "application/json",
        "Authorization": "Bearer <token>"
      },
      "body_params": {
        "type": {
          "type": "String",
          "required": true,
          "description": "Debtor type, 'PERSONAL' or 'CORPORATE'"
        },
        "product_id": {
          "type": "Number",
          "required": true,
          "description": "Product ID code (see documentation reference for
mapping)"
        },
        "inquiry_reason": {
          "type": "Number",
          "required": true,
          "description": "Inquiry reason code (see reference table for meaning)"
        },
        "reference_code": {
          "type": "String",
          "required": false,
          "description": "Client reference code (optional)"
        },
        "params": {
          "type": "Array<Object>",
          "required": true,
          "description": "List of parameter objects containing debtor
identifying info (e.g. id_type, id_no, name, etc. as required by product_id)"
        }
      },
      "success_response": {
        "status_code": 200,
        "body": {
          "code": "01",
          "status": "success",
          "message": "Data ditemukan",
          "inquiry_id": "<number>",
          "data": [
            {
              "similarity_score": 100.0,
              "id_pefindo": "<number>",
```

```
            "id_type": "KTP",
            "id_no": "<ID number>",
            "id_tipe_debitur": "PERSONAL",
            "nama_debitur": "<Name>",
            "tanggal_lahir": "YYYY-MM-DD",
            "npwp": "<Tax ID or NPWP>",
            "alamat": "<Address>",
            "...": "..."
          }
        ],
        "response_status": "ALL_CORRECT"
      }
    },
    "error_responses": [
      {
        "status_code": 404,
        "body": {
          "code": "31",
          "status": "failed",
          "message": "Data tidak ditemukan"
        }
      },
      {
        "status_code": 500,
        "body": {
          "code": "99",
          "status": "failed",
          "message": "[<system error message>]"
        }
      }
    ]
  },
  {
    "name": "Generate Report",
    "method": "POST",
    "path": "/product/generateReport",
    "authentication": {
      "type": "Bearer",
      "token_required": true
    },
    "headers": {
      "Content-Type": "application/json",
      "Authorization": "Bearer <token>"
    },
    "body_params": {
      "inquiry_id": {
        "type": "Number",
        "required": true,
```

```
          "description": "Inquiry ID from /search response"
        },
        "ids": {
          "type": "Array<Object>",
          "required": true,
          "description":
"List of debtor ID objects selected from search results (each with id_type,
id_no, id_pefindo)"
        },
        "event_id": {
          "type": "String (UUID)",
          "required": true,
          "description": "Client-generated unique UUID for this report request"
        },
        "generate_pdf": {
          "type": "String",
          "required": false,
          "description": "'1' to generate a PDF file, '0' or omit for no PDF"
        },
        "language": {
          "type": "String",
          "required": false,
          "description":
"PDF language code, '01' (Bahasa) or '02' (English), if PDF requested"
        }
      },
      "success_response": {
        "status_code": 200,
        "body": {
          "code": "01",
          "status": "success",
          "event_id": "<UUID>",
          "message": "Proses membuat report sedang dikerjakan"
        }
      },
      "error_responses": [
        {
          "status_code": 409,
          "body": {
            "code": "35",
            "status": "failed",
            "event_id": "<UUID>",
            "message": "event_id sudah ada, gunakan yang lain"
          }
        },
        {
          "status_code": 404,
          "body": {
```

```
          "code": "31",
          "status": "failed",
          "message": "Data tidak ditemukan"
        }
      },
      {
        "status_code": 500,
        "body": {
          "code": "99",
          "status": "failed",
          "message": "[<system error message>]"
        }
      }
    ]
  },
  {
    "name": "Get Report",
    "method": "GET",
    "path": "/product/getReport/event_id/{event_id}",
    "authentication": {
      "type": "Bearer",
      "token_required": true
    },
    "headers": {
      "Authorization": "Bearer <token>"
    },
    "path_params": {
      "event_id": "UUID of the report request (from generateReport)"
    },
    "body": null,
    "success_responses": [
      {
        "condition": "Report ready",
        "code": "01",
        "status_code": 200,
        "body": {
          "code": "01",
          "status": "success",
          "message": "Laporan berhasil dibuat",
          "event_id": "<UUID>",
          "report": { "... full report data object ...": "..." }
        }
      },
      {
        "condition": "Big report",
        "code": "36",
        "status_code": 200,
        "body": {
```

```
            "code": "36",
            "status": "success",
            "message": "Kategori big report, detail fasilitas tidak dikirim,
silahkan download...",
            "event_id": "<UUID>",
            "report": { "header": { ... }, "debitur": { ... } }
          }
        },
        {
          "condition": "Still processing",
          "code": "32",
          "status_code": 200,
          "body": {
            "code": "32",
            "status": "success",
            "message": "Laporan masih dalam proses scoring"
          }
        }
      ],
      "error_responses": [
        {
          "status_code": 404,
          "body": {
            "code": "31",
            "status": "failed",
            "message": "Data tidak ditemukan"
          }
        },
        {
          "status_code": 500,
          "body": {
            "code": "99",
            "status": "failed",
            "message": "[<system error message>]"
          }
        }
      ]
    },
    {
      "name": "Download Report (Big Report JSON)",
      "method": "GET",
      "path": "/product/downloadReport/event_id/{event_id}",
      "authentication": {
        "type": "Bearer",
        "token_required": true
      },
      "headers": {
        "Authorization": "Bearer <token>"
```

```
    },
    "path_params": {
      "event_id": "UUID of the report request"
    },
    "query_params": {
      "page": "(optional) page index for pagination",
      "max": "(optional) max page or part count"
    },
    "body": null,
    "success_response": {
      "status_code": 200,
      "body": {
        "code": "01",
        "status": "success",
        "event_id": "<UUID>",
        "report": { "... full report data object ...": "..." }
      }
    },
    "error_responses": [
      {
        "status_code": 404,
        "body": {
          "code": "31",
          "status": "failed",
          "message": "Data tidak ditemukan"
        }
      },
      {
        "status_code": 500,
        "body": {
          "code": "99",
          "status": "failed",
          "message": "[<system error message>]"
        }
      }
    ]
  },
  {
    "name": "Download PDF Report",
    "method": "GET",
    "path": "/product/downloadPdfReport/event_id/{event_id}",
    "authentication": {
      "type": "Bearer",
      "token_required": true
    },
    "headers": {
      "Authorization": "Bearer <token>"
    },
```

```
      "path_params": {
        "event_id": "UUID of the report request"
      },
      "body": null,
      "success_response": {
        "status_code": 200,
        "content_type": "application/pdf",
        "body": "<PDF binary data>"
      },
      "error_responses": [
        {
          "status_code": 404,
          "body": {
            "code": "31",
            "status": "failed",
            "message": "Data tidak ditemukan"
          }
        },
        {
          "status_code": 500,
          "body": {
            "code": "99",
            "status": "failed",
            "message": "[<system error message>]"
          }
        }
      ]
    }
  ]
}
```

---

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118
119 120 121 122 123 124 125 PEFINDO_API_DOCS.pdf

file://file-TDC44fd58p9N3vgNnh618Y