Avira | OEM

16 November 2023

File Reputation API v. 1.0 (APC)

Table of Contents

1. Introduction

The **File Reputation API (APC)** is a global, cloud-based system that classifies files in real time. Behind it there's an online database containing hundreds of millions of file metadata and a set of innovative algorithms that help decide if a file is potentially dangerous. APC can be leveraged to enhance local antivirus detection and provide real-time protection against unknown threats.

1.1. Terminology

Term	Description
APC	Avira Protection Cloud
category	The security classification of a file (e.g. Clean, Trojan, etc.)
file_sha256	The SHA256 of a file, e.g. `bf004766bdc5996cd1766c09f9eb2022789487fd2266c405c5a080d13d128802`
customer	Integrator of our service
product	The customer's product
PII	Personally Identifiable Information

2. API Reference

2.1. Entry point

The API is available at `https://query-api.eu1.apc.avira.com`

2.2. Authorization

Authorization is based on API KEYs, sent via HTTP Authorization header

```
curl -H 'Content-Type: application/json' -H 'X-AVIRA-APIKEY: your_api_key_here' https://query-
api.eu1.apc.avira.com/files/v1/query -d 'payload'
```

2.3. Usage

In order to use the API, a client must send properly formatted `JSON` messages using HTTP’s `POST` method; the `Content-Type` must be set to `application/json`.

The URL for accessing the API looks like
`https://query-api.eu1.apc.avira.com/<component>/<version>/<action>`

where `component` is currently

- `files` (APC)

`version` is detailed for each component and `action` is also component specific.

Current version is **v1**.

2.4. API methods

2.4.1. /files/v1/query

Query for the security status of a file by its SHA256 hash.

- **Endpoint**: `/files/v1/query`
- **HTTP Method**: `POST`
- **Payload**: `JSON`

Request (POST)

The `JSON` sent as body for a `/files/v1/query` request has to contain the following members (described with JSON Pointers):

Member	Type	Required	Description
/sha256	object	yes	A dictionary with the SHA256 hashes of the file(s).
/sha256/file_sha256	object	yes	Data associated with the file; it can be empty.
/sha256/file_sha256/size	numeric	yes	The size of the file, in bytes. If the request is done with the `FLAG_WOULD_UPLOAD` set (value 1), then the `size` is mandatory, else it can be omitted.
/sha256/file_sha256/gen_data	object	no	Generic data associated with the file (e.g. file path, sensors, etc.) (see Section 2.6)
/flags	numeric	no	Bitwise flags indicating how the server should handle this request. (see Section 2.7).
/metadata	object	no	The client metadata, e.g. the unique client identifier, language, country, etc. (see Section 2.5).

Response

In case of success, a `HTTP 200` status code is returned and the body of the response contains a `JSON` with the following members:

Member	Type	Description
/sha256	object	A dictionary with the SHA256 hashes of the file(s) and their attributes
/sha256/file_sha256	object	The data associated with the file.
/sha256/file_sha256/cat	numeric	The security category returned by the cloud; 0 = Unknown, 1 = Clean, etc. (see Section 2.12).
/sha256/file_sha256/status	string	The status of the operation in the cloud (see Section 2.9); if missing, the status is OK.
/sha256/file_sha256/extra_info	object	The extra information sent by the cloud, if requested (associated with `request_extra_info` flag [not yet implemented]).
/sha256/file_sha256/det_name	string	The detection name if there's a detection; null if the file is Clean.
/sha256/file_sha256/known	boolean	If true, we are very certain about the category of the file.

Member	Type	Description
/sha256/file_sha256/ttl	numeric	The time in seconds for how long the client can cache the result; after this time period, it’s recommended that the client repeats the request to the cloud, if needed. See Time To Live).
/sha256/file_sha256/first_seen	numeric	Special metadata indicating when the file was added to Avira Cloud (**needs special license to be visible**).
/sha256/file_sha256/times_requested	numeric	Special metadata indicating how often the file was requested from Avira Cloud since it was added (**needs special license to be visible**).
/sha256/file_sha256/prevalence_band	numeric	Special metadata indicating the number of unique users which have seen this hash (**needs special license to be visible**). See Section 2.11

If the backend has no information about the file and the upload conditions to upload it to the cloud are not met (client is not willing to upload the file, or the file size is too large), the response will contain cat 0 (`Unknown`) and the status will be `OK`.

The `query` method can be used in a scenario where only hash checks are done in the cloud; in this case the `FLAG_WOULD_UPLOAD` will be set to `0` indicating that no uploads will follow the request and the file size is not mandatory to be present in the request.

If the client is willing to upload files to the cloud to be analysed, then the request must have `FLAG_WOULD_UPLOAD` set to `1` and contain the size of the file.

Example

Request:

```
{
  "sha256": {
    "63a12df4cf36a5983ec22ea5027a609720ca74b11ccef7493f18c096c11533d1": {
      "size": 1893827,
      "gen_data": {
        "file_path": "C:\\Program Files (x86)\\ASUS\\Giftbox\\Asusgiftbox.exe"
      }
    },
    "bf004766bdc5996cd1766c09f9eb2022789487fd2266c405c5a080d13d128802": {
      "size": 9186
    }
  },
  "flags": 1,
  "metadata": {
    "user_randid": "53bb23d62d0f42d9581a74ba08631ca52a783fd7"
  }
}
```

Response:

```
{
  "sha256": {
    "63a12df4cf36a5983ec22ea5027a609720ca74b11ccef7493f18c096c11533d1": {
      "cat": 3,
      "det_name": "TR/Gen",
      "ttl": 3600,
      "known": false,
      "first_seen": 1491886865,
      "times_requested": 664,
      "prevalence_band": 4
    },
    "01df91e80a51cabd10e9ae5ec0a9f697cbcfafd7fc0db0aaaeeaf11dd4c5ffab": {
      "cat": 1,
      "ttl": 2592000,
      "known": false,
      "first_seen": 1491886859,
      "times_requested": 281,
      "prevalence_band": 3
    },
    "bf004766bdc5996cd1766c09f9eb2022789487fd2266c405c5a080d13d128802": {
      "cat": 1,
      "ttl": 7183073,
      "known": true,
      "first_seen": 1491897655,
      "times_requested": 279,
      "prevalence_band": 3
    }
  }
}
```

2.4.2. /files/v1/upload

Files can be uploaded to the back-end as part of a `multipart/form-data` payload. If the status from a query request is `UPLOAD`, it indicates that APC wants to receive this file for analysis; however, the upload is possible only if the client has set the corresponding flag when doing the query(`"flags": 1`). In this case, the reply will contain a field `upload_to`, which is the URL where the file must be uploaded.

- **Endpoint**: `/files/v1/upload`
- **HTTP Method**: `POST`
- **Payload**: `multipart/form-data`

The uploaded file can be `bzip2` archived (with the extension `.bz2`) or uncompressed and placed in a form field called `file`; additional metadata is added in a similar way (see example). All text must be `utf-8` encoded.

Example

Request (initial query):

```
curl -H 'Content-Type: application/json' -H 'X-AVIRA-APIKEY: your_api_key_here' https://query-
api.eu1.apc.avira.com/files/v1/query -d '{"sha256":
{"569751a54e58cdfd936cbbfc537f406c76f1e49675908d91c508fe3ddebbba40": {"size": 537600}},"flags":
1, "metadata": {"user_randid": "53bb23d62d0f42d9581a74ba08631ca52a783fd7"}}'
```

Response:

```
{
  "sha256": {
    "569751a54e58cdfd936cbbfc537f406c76f1e49675908d91c508fe3ddebbba40": {
      "status": "UPLOAD",
      "upload_to": "https://query-
api.eu1.apc.avira.com/files/v1/upload/569751a54e58cdfd936cbbfc537f406c76f1e49675908d91c508fe3dde
bbba40",
      "det_name": null,
      "ttl": 2,
      "known": false,
      "cat": 0
    }
  }
}
```

The ttl in response indicates the time in seconds after which the client should query the back-end for a response, after the file was uploaded.

There are file size restriction implemented, currently the back-end would accept files with maximum size of **4 MB**.

With CURL command, an upload would look like this:

```
curl -H 'Content-Type: multipart/form-data' -H 'X-AVIRA-APIKEY: your_api_key_here'
https://query-
api.eu1.apc.avira.com/files/v1/upload/569751a54e58cdfd936cbbfc537f406c76f1e49675908d91c508fe3dde
bbba40 -F "file=@PE.Explorer_setup.exe.bz2" -F "params=@params.json"
```

In the `params` file, we have a `JSON` object containing the `metadata` and `gen_data` for the file, same as in the query request.

Example `params.json`:

```
{
  "gen_data": {
    "file_path": "C:\\Program Files (x86)\\ASUS\\Giftbox\\Asusgiftbox.exe",
    "file_name": "Asusgiftbox.exe"
  },
  "metadata": {
    "user_randid": "53bb23d62d0f42d9581a74ba08631ca52a783fd7",
    "os_type": 1,
    "os_vmajor": 6,
    "os_vminor": 1,
    "os_vbuild": 7601
  }
}
```

Although the `metadata` / `gen_data` fields are not mandatory, is recommended to attach them as parameters (`params`) to the upload requests since some of them are used to deliver a better detection.

While querying the back-end for a response, after the file was uploaded, the server can respond with `"status": "IN_PROGRESS"`, indicating that the file is still being analyzed. It will also indicate an estimate for when the results will be ready, with the `ttl` field.
The client must repeat the query (for a limited number of times) until it will receive a `"status": "OK"` response with the classification for the file; the number of retries can be decided on each client implementation, depending on how long it can wait for a response.

If a client tries to upload a file (using the upload entry point) without being asked by the server, the uploaded file will be rejected.

2.4.3. /files/v1/non-pe-exts

Returns the list of supported non-PE extensions (only those non-PE file types should be requested / uploaded to APC if the license allows non-PE).

- **Endpoint**: `/files/v1/non-pe-exts`
- **HTTP Method**: `GET`
- **Payload**: `None`

Example

Request:

```
curl -H 'X-AVIRA-APIKEY: your_api_key_here' https://query-api.eu1.apc.avira.com/files/v1/non-pe-
exts
```

Response:

```
{
    "exts": ["apk", "swf", "dex", "do*", "?ht*", "asp", "ht*", "php", "shtm*", "jar", "class",
"lnk", "bin", "pdf", "ppt*", "pot*", "pps*", "rtf", "xl*", "bat", "cmd", "csh", "css", "eml",
"inf", "ini", "ins", "isp", "js*", "osd", "pl*", "ps1", "psh", "scf", "script", "sh", "vb*",
"wsc", "wsf", "wsh", "xml", "vb?", "mpp", "mpt", "ms?", "pkg"]
}
```

2.4.4. /stats/v1/requests

This endpoint will deliver a daily statistic for the queries / uploads done by the customer in any time interval in the last 30 days.

- **Endpoint**: `/stats/v1/requests`
- **HTTP Method**: `POST`
- **Payload**: `JSON`

Request (POST)

The `JSON` sent as body for a `/files/v1/query` request has to contain the following members (described with JSON Pointers):

Member	Type	Required	Description
start_day	string	yes	The start of the time interval in `YYYY-MM-DD` format
end_day	string	yes	The end of the time interval in `YYYY-MM-DD` format

Response

In case of success, a `HTTP 200` status code is returned and the body of the response contains a `JSON` with the following members (described with `JSON POINTERS`):

Member	Type	Description
/data	object	A dictionary containing daily statistics
/data/YYYY-MM-DD	object	A dictionary with the statistics for the day `YYYY-MM-DD`
/data/YYYY-MM-DD/queries	numeric	The number of `query` requests done in the `YYYY-MM-DD` day
/data/YYYY-MM-DD/uploads	numeric	The number of `upload` requests done in the `YYYY-MM-DD` day

Example

Request:

```
curl -D- -H 'Content-Type: application/json' -H 'X-AVIRA-APIKEY: your_api_key_here'
https://query-api.eu1.apc.avira.com/stats/v1/requests -d '{"start_day": "2018-05-01", "end_day":
"2018-05-05"}'
```

Response:

```
{
  "data": {
    "2018-05-01": {
      "queries": 3758,
      "uploads": 1
    },
    "2018-05-02": {
      "queries": 3977,
      "uploads": 45
    },
    "2018-05-03": {
      "queries": 2545,
      "uploads": 10
    },
    "2018-05-04": {
      "queries": 74353,
      "uploads": 16
    },
    "2018-05-05": {
      "queries": 637,
      "uploads": 16
    }
  }
}
```

2.4.5. /stats/v1/quota

A quota is the number of requests a customer can make to the API within an agreed time interval. A time interval is defined in the contract, and may be any period of time between a minute and one year.

When a quota limit is reached, the back-end may stop responding to requests until the end of the current time interval, or it may continue to respond to requests, depending on contract agreements.

A customer may have one or several quotas, depending on their needs.

- **Endpoint**: `/stats/v1/quota`
- **HTTP Method**: `GET`
- **Payload**: `JSON`

Request (GET)

No additional parameters are needed, a call to `https://query-api.eu1.apc.avira.com/stats/v1/quota` (as shown in the Section 2.2 section) will give you the statistics for your service quota.

Response

In case of success, a `HTTP 200` status code is returned and the body of the response contains a `JSON` with the following members (described with `JSON Pointers`):

Member	Type	Description
/quota	object	A list with all the service's quota attributes.
/quota/[x]/type	string	The service type (`query` | `upload`).
/quota/[x]/interval	string	Time interval (`minute` | `hour` | `day` | `month` | `quarter` | `biannual` | `year`).
/quota/[x]/maximum	numeric	The maximum allowed number of requests for the service (the actual quota).
/quota/[x]/current	numeric	The current usage of the allowed quota.
/quota/[x]/allow_exceed	boolean	Determines what action should be taken if quota is exceeded (deny requests or not).

Example

The customer has a quota of 15,000,000,000 requests / month and 3,000,000 uploads / quarter. The service will continue to deliver even if the quota is reached.

```
{
  "quota": [{
    "current": 8,
    "allow_exceed": true,
    "interval": "month",
    "type": "query",
    "maximum": 15000000000
  }, {
    "current": 0,
    "allow_exceed": true,
    "interval": "quarter",
    "type": "upload",
    "maximum": 3000000
  }]
}
```

The customer has a quota of 15,000,000,000 requests / month but no more than 500,000,000 requests / day, and 3,000,000 uploads / quarter. If query's daily quota is reached, the service will stop delivering until next day. If the monthly quota for queries or quarterly quota for uploads will be reached, the service will continue to deliver.

```
{
  "quota": [{
    "current": 7,
    "allow_exceed": false,
    "interval": "day",
    "type": "query",
    "maximum": 500000000
  }, {
    "current": 64,
    "allow_exceed": true,
    "interval": "month",
    "type": "query",
    "maximum": 15000000000
  }, {
    "current": 20,
    "allow_exceed": true,
    "interval": "quarter",
    "type": "upload",
    "maximum": 3000000
  }]
}
```

2.5. File Metadata

Currently accepted fields in `metadata` are:

user_randid

- **Type**: string
- **Description**: The unique identifier for the end user, e.g. `710451ce65c27b580f5623f437d14c2d605f7495`.

os_type

- **Type**: integer
- **Description**: The Operating System type:

Value	Meaning
1	WINDOWS
2	LINUX
3	MACOS
4	SOLARIS
5	FREEBSD
6	ANDROID
7	OPENBSD
255	OTHER

os_arch

- **Type**: integer
- **Description**: Info about HW architecture:

Value	Meaning
1	x86
2	x86_64
3	SPARC
4	SPARC64
5	PPC

Value	Meaning
6	PPC64
7	ARM
255	OTHER

os_lang

- **Type**: string
- **Description**: Language of the Operating System. e.g. `de`

os_vbuild

- **Type**: integer
- **Description**: Build number for the Operating system, e.g. `7601`

os_vmajor

- **Type**: integer
- **Description**: Major version for the Operating System, e.g. `6`

os_vminor

- **Type**: integer
- **Description**: Minor vwersion for the Operating System, e.g. `1`

2.6. File Generic Data

Currently accepted fields in `gen_data` are:

file_name

- **Type**: string
- **Description**: The file name, e.g. `csrss.exe`

file_path

- **Type**: string
- **Description**: The complete file path, if available, with all the PII removed, e.g. `C:\Users\X\Downloads\ChromeSetup.exe`

url

- **Type**: string
- **Description**: The URL from which the file originated, e.g. `https://cdn.discordapp.com/attachments/273346914890809344/300938577070522369/csrss.exe`

2.7. File Flags

Currently implemented:

Flag	Value	Description
`FLAG_WOULD_UPLOAD`	1	If the client is willing to upload the file to the cloud. Send 0 if not willing to upload (queries only).

2.8. File Status Codes

Status	Description
OK	The cloud has processed the request and has provided the answer.
IN_PROGRESS	The cloud is still processing the request (e.g. scanning a file).
UPLOAD	The cloud is requesting the file to be uploaded to analyze it.

2.9. HTTP Status Codes

Code	Description
200	Operation successful.
400	Bad request (e.g. missing mandatory parameters).
401	Unauthorized (e.g. invalid `API KEY`)
404	Endpoint not found.
405	Method not allowed (e.g. using `GET` when `POST` is needed).
413	Request too large (e.g. too many hashes in a `query` request).
415	Header `Content-Type: application/json` not set for `POST` methods.
429	Too many requests (intended for use with rate-limiting).
500	Internal server error (something is malfunctioning, try again later).

2.10. Time To Live (ttl)

Time to live (`ttl`) is the time, in seconds, for how long a client can cache the response or, in case of upload, when to request the response of a file scan. We have two types of detections, static and dynamic. The static detection is when a detection is assigned to a file directly by a process/researcher, the dynamic one is when the detection is determined dynamically from the results of the scan engines. Possible values for the `ttl` are:

- Interval **[1, 10]** when the status returned is `UPLOAD` or `IN_PROGRESS`;
- **3600** when the category returned is not `CLEAN` (the file is malicious);
- **2592000** when the category returned is dynamic `CLEAN`;
- Interval **[4838400, 7257600]** when the category returned is static `CLEAN`.

2.11. Prevalence Bands

Mapping between the number of times has been seen by unique users:

Unique users count	Prevalence band
0 to 4	1
5 to 49	2
50 to 99	3
100 to 999	4
1,000 to 9,999	5
10,000 to 99,999	6
100,000 to 999,999	7
>1,000,000	8

2.12. File Category to Numeric mapping

Category	Numeric ID	Description	Blocked by Avira	Is malicious
UNKNOWN	0	No information	no	no
CLEAN	1	Clean	no	no
MALWARE	2	Malware	yes	yes
ADSPY	3	Adware with spying functions	yes	yes
AWARE	4	Adware	yes	yes
APPL	5	Application, not necessarily suspicious	no	no
BAT	6	Batch Malware	yes	yes
BDC	7	Backdoor Client	yes	no
BDS	8	Backdoor Server	yes	yes
BOO	9	Malicious Bootsector	yes	yes
DDOS	10	Distributed Denial-Of-Service malware	yes	yes
DIAL	11	Dialer	no	no
DOS	12	DOS-based Malware	yes	yes
DR	13	Malware Dropper	yes	yes
EML	14	E-Mail which contains malicious content	yes	yes
EXP	15	Exploit / Vulnerability	yes	yes
GAME	16	Game Program	no	no
HTML	17	HTML Script Malware	yes	yes
IRC	18	Internet Relay Chat Malware	yes	yes
JAVA	19	JAVA Malware	yes	yes
JOKE	20	Joke Application	yes	yes
JS	21	Javascript Malware	yes	yes
KIT	22	Malware Creation Kit	yes	no
LINUX	23	Linux-based Malware	yes	yes
OSX	24	Apple OSX-based Malware	yes	yes
PERL	25	PERL Language Malware	yes	yes
PFS	26	Potential Fake Software	yes	yes
PHISH	27	Phishing E-Mail or Webpage	yes	no
PHP	28	PHP Language Malware	yes	yes
RKIT	29	Rootkit	yes	yes
SPR	30	Security Privacy Risk Tool	yes	no
SWF	31	Shockwave Flash Malware	yes	yes
SYMBOS	32	SymbianOS-based Malware	yes	yes
TR	33	Trojan Horse / Ransomware	yes	yes
UNIX	34	Linux-based Malware	yes	yes
VBS	35	Visual Basic Script Malware	yes	yes
W32	36	Windows 32bit Virus or File Infector	yes	yes
W64	37	Windows 64bit Virus or File Infector	yes	yes
WORM	38	Worm	yes	yes
W95	39	Windows95-based Malware	yes	yes
W2000	40	Windows2000-based Malware	yes	yes
HEUR	41	Suspicious content / Heuristic-based Malware	yes	yes

Category	Numeric ID	Description	Blocked by Avira	Is malicious
PCK	42	Using unusual packing methods	no	no
ANDROID	43	AndroidOS-based Malware	yes	yes
HIDDENTEXT	44	Misleading file extension	yes	yes
W2000M	45	Macrovirus for Office Word 2000	yes	yes
W97M	46	Macrovirus for Office Word 97	yes	yes
PUA	47	Potential Unwanted Application	yes	no
CLN	48	Internal classification	-	-
VBA	49	Visual Basic Application Malware	yes	yes

3. Contact Information

3.1. Support Services

During evaluation, integration and live use

If you are evaluating or starting to integrate Avira's technology into your solution, or if your integration is finalized and you are going to release your solution to your customers, the Integration Support engineers will answer your technical questions — from planning the architecture of the integration, to detailed code-related routines and live use.

To contact the OEM support team for technical issues, write an email to: oemsupport@avira.com

Partner Portal

For our OEM customers we also provide a login to our Partner Portal which includes all the latest news and information about Avira's technology, SDK downloads, and documentation: OEM Partner Portal

3.2. Contact

Avira Operations GmbH & Co. KG Kaplaneiweg 1 D-88069 Tettnang Germany
You can find further information about us and our products on the Avira OEM website