`

APWG eCrime eXchange User Guide

Version 2.0, 15 October2021

Table of Contents

Overview: Get to Know the eCrime eXchange

This user guide tells you how to use the APWG's eCrime eXchange (the "eCX"). Here you can learn what kinds of data it contains, and tips for how to get the most out of it.

Your Account, and Your Organization's Account

Log into the eCX's web interface at https://ecrimex.net. There you can view and update your user information, including your password, contact info, and mail and group preferences.

Each user of the eCX receives their own account, with its own username and password. Each user account is associated with an APWG organization membership. The organization's APWG membership level determins the number of indiciual users are allowed to access eCX and what modules they have access too. In most cases an APWG member organization can have multiple individual users.

If your organization needs to create a new user account for one of your staff members, an existing user should send a request to support@ecrimex.net . Please include the new user's name and email address. New users must have an email address on your organization's domain name. If an individual leaves your organization, please let us know so that we can suspend their account.

Billing contacts: each APWG member organization has a billing contact on file with the APWG. APWG sends annual membership invoices to the billing contact. We ask that you keep your billing contact up to date. Note that if your organization does not pay its membership fees, the APWG may eventually suspend access to the eCX. We always send reminders before suspending access.

Technical and Business Support

If you need help, please contact our support team at: support@ecrimex.net We're here to help!

If you're reporting a suspected outage or a potential bug, please write to us with the details, including a step-by-step account of how you encountered the problem, the error message your received, the time of the problem (including the time zone you're referencing), and a screenshot if relevant. These will help us diagnose and reproduce the problem.

The eCX support team occasionally schedules maintenance periods in order to upgrade and service the eCX. When we do so, we will send you advance notice, to minimize disruptions to your work. If there is an unscheduled maintenance, we will send outage notices and updates until the situation is

resolved.

In general, the APWG staff cannot provide detailed evaluations of or fixes to your client-side API code.

Data Origin and Quality

The eCX is a user-supported and user-maintained repository of threat data. The data is submitted by vetted APWG members, who are expected to provide reliable data. The data is there for other members to use as-is, per the eCX terms of use. Neither the APWG nor the data submitters guarantee that the data is fit for a given purpose, and the APWG does not guarantee that the data is 100% accurate. This means that the repository may occasionally contain false-positives and outdated information.

The APWG staff typically do not make changes to existing data in the repository, for several reasons. The APWG vets all members prior to allowing access to the platform, so our user base is 100% full of known good actors. We don't monitor or vet the incoming data. When data needs to be updated we provide our users with the tools to make updates, so members have the same control over the data. And finally, we're a very small team--we don't have the bandwidth to moderate and modify data, and if we were to be making unilateral changes to the data then we'd be taking the chance of stepping on the toes of our valuable members. If the APWG staff is asked to update data, we will usually forward the request to the member who created the entry with a request to examine and update it as needed.

Ways to Interact with the Data: UI and API

There are two methods to interact with the eCX. The eCX has a user-friendly web-based user interface (UI) at https://ecrimex.net . This option works well for low-volume, manual data entry. It's also handy for browsing data. By default you are allowed to view and submit records through the eCX UI the moment you are approved to access the eCX. Both the eCX UI and the eCX API allow you to create new records, and to make changes to existing data.

If you want to automate your interaction with the data, use the eCX API. It's ideal for posting or pulling data in a programmatic way, and can also be used to make queries. The eCX API is a REST API. For an introduction to REST APIs please see
https://en.wikipedia.org/wiki/Representational_state_transfer

All modules within eCX follow the same general submission flow described later in this document, differing mainly in the schema of the data.

We do not allow users to submit data into the eCX production environment via the API until your automated scripts are proven in the eCX sandbox and then approved for automated submissions – see below for more.

API Documentation

This user guide does not attempt to detail all the workings of the eCX's API. To learn more details about the API, please examine the online API documentation. To view that, log in at https://ecrimex.net and visit the "API" section. The API section also contains your API key. There is also additional documentation provided within each module, which details what fields you may submit and manipulate, what fields are mandatory versus optional, allowed values, and so on.

The eCX API documentation is generated using the OpenAPI format. There is an option to dump the API docs in OpenAPI format. There are many tools for OpenAPI documents online that might help you fast track your API integration. On the eCX API documentation page in the eCX UI (https://ecrimex.net/api) you will find ready-to- go eCX API clients in various languages such as Java, Go, PERL, etc., that were generated using OpenAPI tools to help you fast track your eCX API integration.

APWG also maintains code repositories for your use at Github: https://github.com/APWG
These include code and command line examples to interact with the eCX. Among the examples are how to POST data to the eCX /phish endpoint using cURL, how to GET data from the eCX /phish endpoint using wget on the command line, how to perform a CSV export from eCX using date ranges, and more.

The eCX API supports three REST operations – GET, POST, and PATCH.

- GET is a data fetch operation that allows for querying of data. eCX supports multiple parameters.
- POST sends new data to eCX within a JSON payload.
- PATCH is used to update existing eCX data values using a JSON payload.

Every GET, POST, or PATCH operation the eCX API returns data in JSON. There is a GET parameter that will tell the eCX API to return data to you in CSV, however you'll receive a much smaller subset of data per entity that you will with JSON. This is because using JSON allows nested arrays and objects in the JSON schema, and these arrays and objects cannot be represented in CSV, so the data within these JSON objects and arrays are omitted in CSV. Sending data to eCX using POST requires the data payload to be in JSON. CSV is not an option for POST.

APWG staff use the free Postman API client to interact manually with the eCX API to perform quick testing and queries for data. The client can be downloaded at https://www.getpostman.com/apps . The various examples that follow were generated from Postman's option to create the API interaction you are performing in their application into various programming languages. Information on this Postman feature is outlined in the POST section.

eCX API Token Keys

For every eCX API operation, your eCX API Key will need to be sent in the request's Authorization header. You'll see this in the examples for every operation shown here. Your eCX API Key is shown in your eCX user profile, located at https://ecrimex.net/users/update . Your eCX API Key is tied to your user account and allows eCX to determine which access rights are available to you.

Response Codes

Following standardized API interaction protocols, the eCX API returns standard HTTP response codes indicating success or failure of each operation. Users should always check for the expected response code from each operation, and handle situations where the response code returned indicates that the operation failed. The eCX API returns response codes for malformed queries, missing API token keys, and other problems. All of this information, on eCX API use and response codes, is outlined in the eCX API documentation located at https://ecrimex.net/api .

The response codes are:

- 200 – Success
- 201 – Success / Created
- 400 – the JSON data did not validate correctly
- 403 – your account permissions prevent you from submitting data
- 404 – your API token key was missing or invalid
- 409 – this phish has already been entered; use PATCH if you have changes
- 428 – you attempted to POST a phish as a false positive (with a confidence level = 0)

Make sure your submitted data returns 201 for POST and a 200 for PATCH and handle any exceptions that arise.

eCX API result codes in the 4xx range indicate an error. All of the possible result codes specific to each eCX API operation are documented in the eCX API documentation located at https://ecrimex.net/api

A 404 result code indicates that you've either made a mistake in the eCX API URL, or you've omitted sending an eCX API key, or the eCX API key you are using is invalid.
Outside of a 404 result code, any other 4xx result code will return a response body that contains a JSON error object designed to help you understand what went wrong with your operation.

```
"error": {
       "title":
       "invalid
       request",
       "httpStatus":
       400,
```

```
            "messages": [
                "'date_discovered' is not allowed to be changed outside of a
                'status' change"
            ]
        }
```

You can see that eCX returns the result code within the JSON error object as well as a more descriptive error message in a JSON array.

In the case of multiple errors, eCX will return multiple message descriptions within the messages array.

```
"error": {
       "title": "invalid
       request",
       "httpStatus": 400,
       "messages": [
           "'bad_key' is not a valid body
           key", "`confidence_level` must be
           one of 0, 50, 90, 100"
       ]
   }
```

Here are some examples of a few CLI utilities using parameters that return the eCX API result codes to you:

```
curl -w "\ncode: %{http_code}\n\n" --request GET --url
https://api. sandbox.ecrimex.net/phish/87104669-- header 'Authorization:
your eCX API Token Key goes here'

wget --quiet --method GET --header 'Authorization: your eCX API Token
Key goes here' --output-document -
https://api.sandbox.ecrimex.net/phish/87104669--server-response

http GET https://api. sandbox.ecrimex.net/phish/87104669Authorization:your
eCX API Token Key goes here
```

(http returns a result code by default ^, no special parameter necessary)

Approval for Submitting Through the API

First, develop your scripts to work off of the sandbox server located at https://api.sandbox.ecrimex.net . The data you POST or PATCH there will never be distributed and

you are welcome to use the sandbox resource for any edge use cases you can envision with any sort of test data.

You can set the eCX API server URL value as a script parameter so you'll be able to make a quick change later to point to the production server.

The data on the sandbox is reset nightly at 00:00 GMT and is reseeded with live data pulled from the production server to give real and current data to run your scripts against.

We capture usage and performance metrics from all of the servers on the eCX platform. Email support@ecrimex.net when you feel that your submission scripts are well developed. We'll check the sandbox metrics and if we see a problem we'll ask you to make changes. Reapply once you've had 48 hours or so of processing with the new logic and things are looking better. If everything looks good then we'll approve you to submit to production. We'll have you repeat the submit/review process as necessary until your script is well behaved. This approval step is necessary before you begin submitting data to eCX production.

The Modules

The eCX contains five "modules." Think of each as a separate database dedicated to a different type of abuse problem or abuse data. Each module contains some data fields unique to it. However, we're striving to make the modules generally consistent with each other when possible, and in general the API calls work similarly across the modules.

The modules are:

- **Phishing (/phish).** Dedicated to phishing URLs, and associated information such as target.
- **Malicious IP (/mal_ip).** IP addresses that are considered dangerous or the sources of undesirable activity. Due to GDPR, you must sign the code of conduct document to get access to this module.
- **Malicious Domain (/mal_domain).** Suspected or known malicious domains – domains registered by and being used by malefactors.
- **Cryptocurrency (/crypto).** Cryptocurrency addresses that are associated with suspicious or criminal transactions. Supports multiple popular cryptocurrencies.
- **Report Phishing (/report_phishing).** A repository of phishing emails sent via API and also to the APWG's reportphishing@apwg.org mailbox. The records include full Internet headers and the bodies of the emails. These submissions do not feature a Confidence Score, and may sometimes include regular spam emails.

Confidence Level

Most of the modules allow users to submit and manipulate a record's Confidence Level score. This is a way for users to indicate the perceived reliability of that particular record to other users.

There are four possible values for confidence level:

- **100** indicates that the submittor has 100% confidence that the incident submission is accurate. Usually this 100% confidence is achieved by a highly reliable process – such as if the phishing page has been visually confirmed by a staff member.
- **90** indicates that the submittor has high confidence in the submission. A 90 is suitable for phish that are found using an automated or machine learning process – for example if your system spidered and confirmed the presence of a phishing page.
- **50** indicates that you or your machine learning process believes this to be phish, but the phish has not been confirmed by a process that would drive it to a 90 or 100 value, yet you still think the URL should be sent to the eCX.
- **0** indicates that there is no confidence in the entry. A "0" indicates that the submission is probably a false-positive (a domain or URL that should never have been reported as dangerous in the first place). Do not make submissions that begin with a 0 confidence level – these will be rejected. Instead, a 0 confidence level is used to downgrade (via a PATCH) the confidence level of an existing record. Phish that were up at one time but are now down (mitigated) should NOT be marked with a "0" confidence level. Instead, mitigated (down) phish should be moved from "active" to "inactive" status.

Status

In the eCX modules, the Status field indicates whether a threat is "active" (up, a threat, etc.) or "inactive" (down, mitigated, or otherwise no longer functional or a threat). Please keep in mind that once an entry has been submitted, the submitter will likely *not* update the status flag over time. For example, a phishing URL may have been marked as active at the time it was submitted to the eCX, but the phishing page may not be active at the time you download the data about it.

For these reasons, we do *not* recommend that you use the eCX and its status info as an authoritative source of information about whether a malicious event is currently active (up) or inactive (down). Instead, we hope users will take the data and confirm the activity and its current status, if that is important to you.

If you are submitting data to the eCX:

- If the threat is active at the time you submit the data to the eCX, make your submission with the status = active.
- If the threat is not active at the time you submit the data to the eCX, make your submission with the status = inactive. It is useful for you to submit recent but inactive incidents into the eCX – this helps everyone learn about the extent to which phishing is occurring on the Internet.

Making Queries: What's Your Use Case?

When designing your API calls to the eCX, think about your use case, and what data you're interested in. Poorly formed queries can place an unnecessary load on the eCX system, and can burden you with excess data.

In general, phishing attacks are short-lived. If you're using eCX data to protect users, you'll therefore be interested in only fresh, recently submitted data. In this case, mining the eCX for older entries will not help protect users at this time – the old phish are probably down and inactive.

If you're interested in new records, then you'll want to consider how often you query the eCX. Consider: for your purposes, how often is enough? Should you query once an hour? Once every five minutes? Checking for new entries more than once a minute is not recommended.

Also note that it is not usually useful to pull down any given record over and over again. Very few records receive updates (via PATCH), so checking for changes can probably be done less frequently. And changes to older records occur infrequently, for reasons explained below. So if you've crafted a query that retrieves the same record multiple times, you're probably burdening the system unnecessarily, and are not getting new information that will help you.

If you're getting data to train your machine learning process, you may want to capture a set of data and then find the records that are coded at 90 or 100 confidence level.

Phishing Module

Endpoint: *https://ecrimex.net/go/phish*

To submit phish to the eCX there are four values you'll need. For more about these values, please see the online API documentation.

- **date_discovered**: This is the time you discovered the phish. The field is a unix epoch timestamp. The default value is a conversion of the current date and time into an epoch value. If you'd like to report your phish with an earlier date_discovered we'd suggest using http://www.epochconverter.com to work out the correct epoch value then copy & paste the timestamp into the eCX Add window. Please note there are constraints in place to disallow timestamps in the future or older than one year.
- **confidence_level**: see above for the allowed values and definitions.
- **brand**: The brand is the company or entity that the phish is targeting. If you are unsure of the brand (target) then use "other". If there are multiple brands targeted on the phishing page you can use "multiple" or you can specify the most prominent or relevant brand(s).
- **url**: The full phishing URL that you want to report. This is the URL where the phish is located. Feel free to submit more than one URL (in separate POSTs) if there are multiple URLs associated with a phishing attack.

Note that the Phishing Module contains only unique URLs. If a user attempts to POST a URL that is already in the eCX, that POST attempt with be rejected, with a 409 response code. The eCX rejects duplicate URLs in order to keep the data in the repository more usable, and the earliest report of a given URL is also the most useful.

Phishing Module: POST

To POST new data to the eCX API the following pseudo logic should be followed:

POST your phish submission to /phish eCX and check the result code:

- 200 – Success
- 201 – Success / Created
- 400 – the JSON data did not validate correctly
- 403 – your account permissions prevent you from submitting data
- 404 – your API token key was missing or invalid
- 409 – this phish has already been entered; use PATCH if you have changes
- 428 – you attempted to POST a phish as a false positive (with a confidence level = 0)

The phish data for a POST operation goes in the request body as a JSON object:

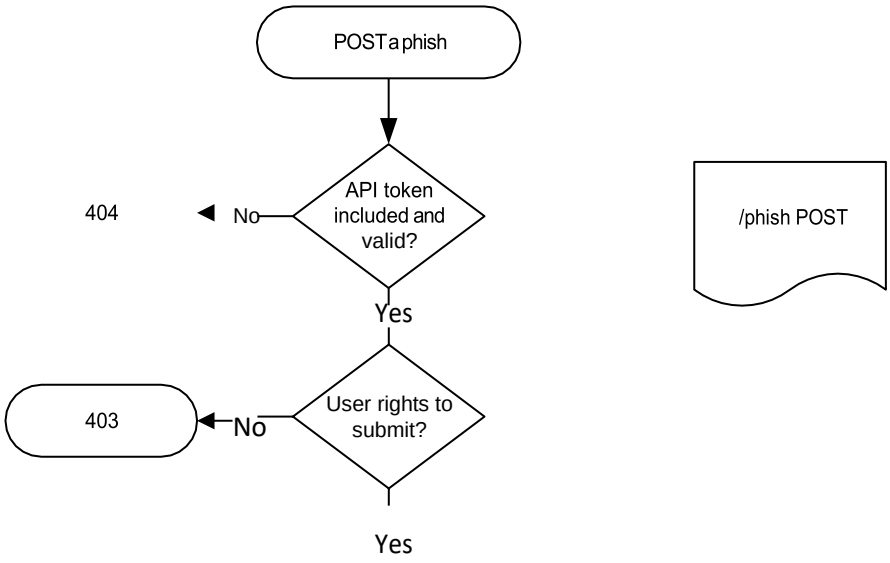
```
{
   "url":  "http://www.dave.com/test1",
   "brand": "dave",
   "date_discovered":  1517439667,
   "confidence_level": 50
}
```

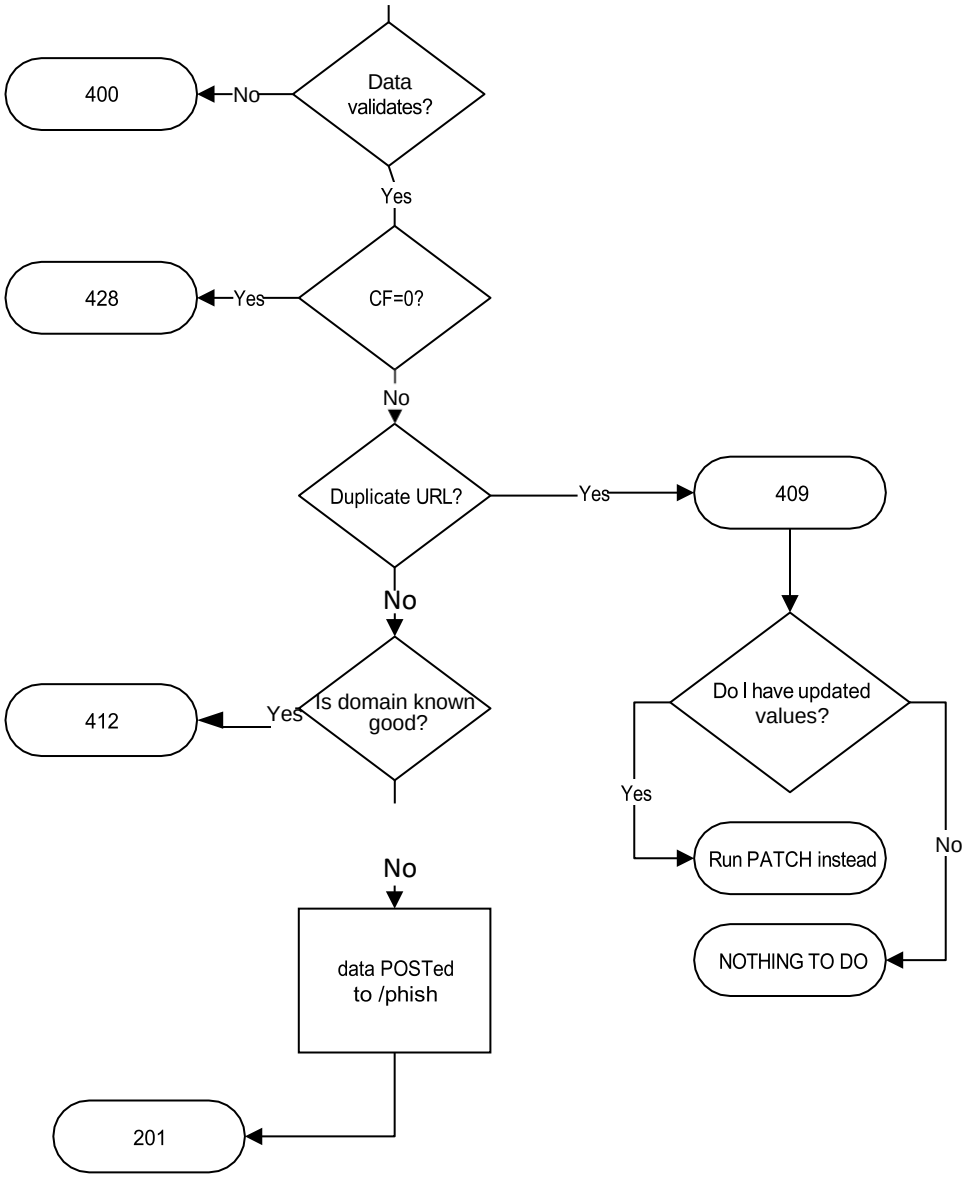
All 4 key properties are required.

Set the request's Content Type header to 'application/json' [to tell the server that the payload it will be dealing with is JSON.]

Upon successful submission, the eCX API will return a response that contains the data saved with the submission along with the ID of the phish entity. If you do not care about the return payload and just want to check the result code of the operation for success/fail then you can use the 'container=empty' parameter to suppress the response body.

The coding examples that follow are PHP and should be easy enough to transcode into another programming language.





POST using PHP using cURL Example:

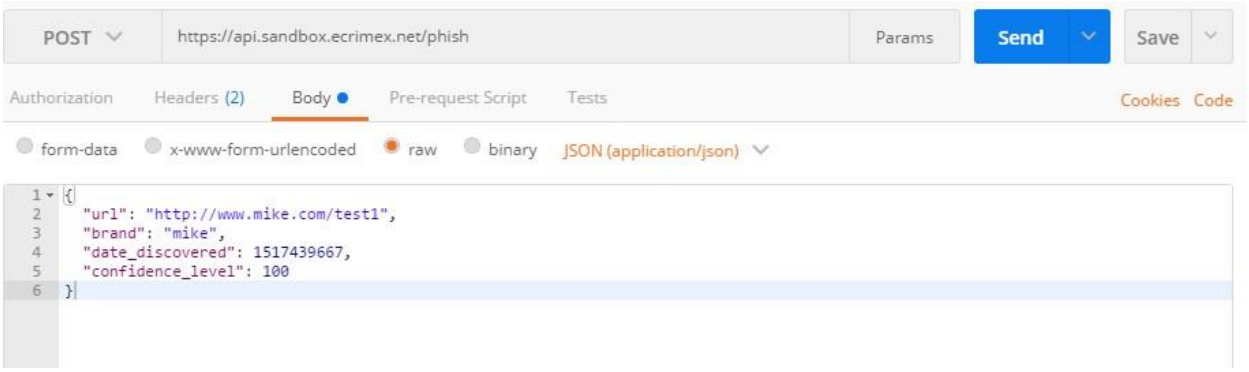
```
<?php
$curl   =
curl_init();
curl_setopt_array($curl
,   array(
  CURLOPT_URL =>
```

```
  "https://api.sandbox.ecrimex.net/phish",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT  => 30,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "POST",
  CURLOPT_POSTFIELDS  =>  "{\r\n    \"url\":
  \"http://www.dave.com/test1\",\r\n    \"brand\":  \"dave\",\r\n
\"date_discovered\":  1517439667,\r\n
  \"confidence_level\":  100\r\n}",
  CURLOPT_HTTPHEADER  =>  array(
    "Authorization: <your eCX API Token Key
    goes here>", "Content-Type:
    application/json"
  ),
));

$response  =  curl_exec($curl);
$err =

curl_error($curl);

curl_close($curl);

if  ($err)  {
  echo "cURL Error #:" . $err;
} else {
  echo  $response;
}
```

POST using cURL CLI Example:

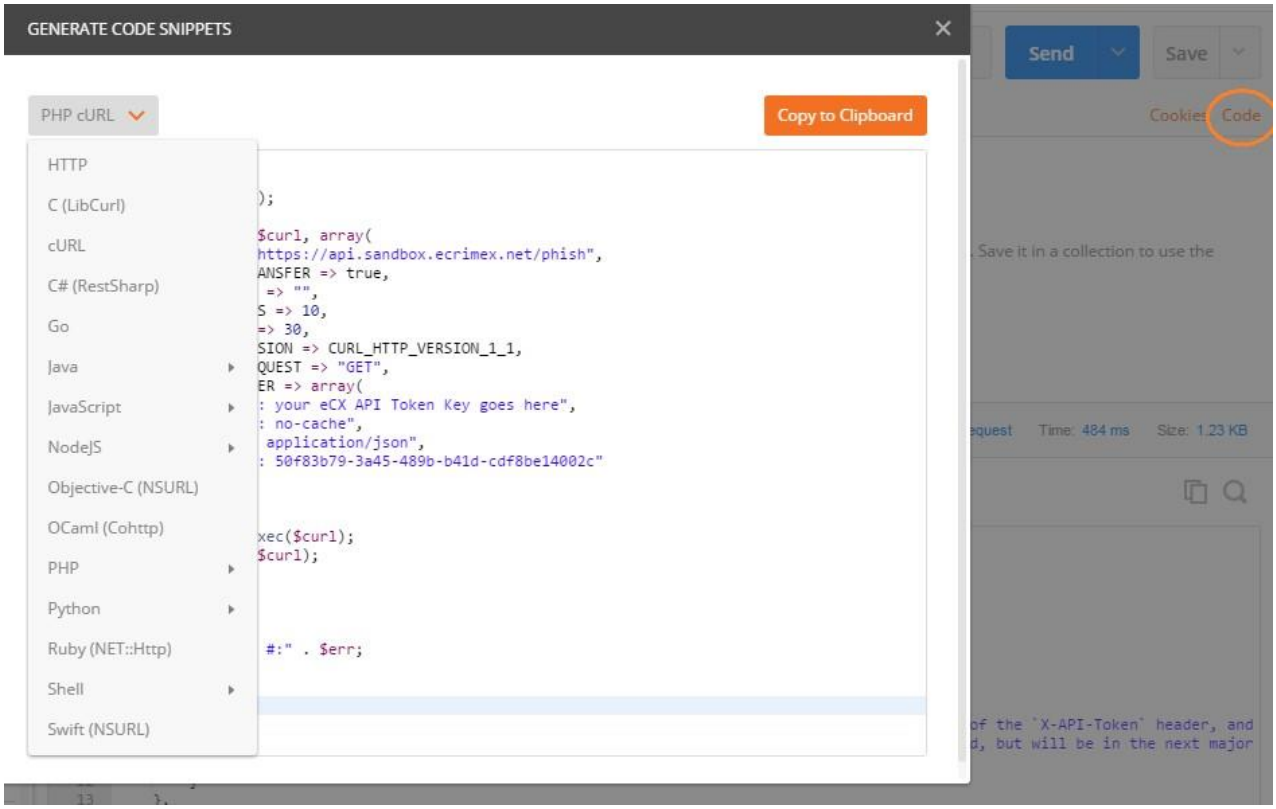
```
curl -X POST \
  https://api.sandbox.ecrimex.net/
  phish  \
  -H 'Authorization: <your eCX API Token Key goes here>'  \
  -H 'Content-Type: application/json' \
  -d '{
  "url":
  "http://www.dave.com/test1",
  "brand": "dave",
  "date_discovered":  1517439667,
  "confidence_level": 100
}'
```

POST using Postman Example:



Press the blue Send button to run the POST operation in Postman.

We used the “Code” link to export the API operation into the PHP and cURL code above. A screenshot of the Code Generation within Postman is below, with the “Code” link circled in orange. You can see the wide range of programming languages Postman will generate code for your query in:



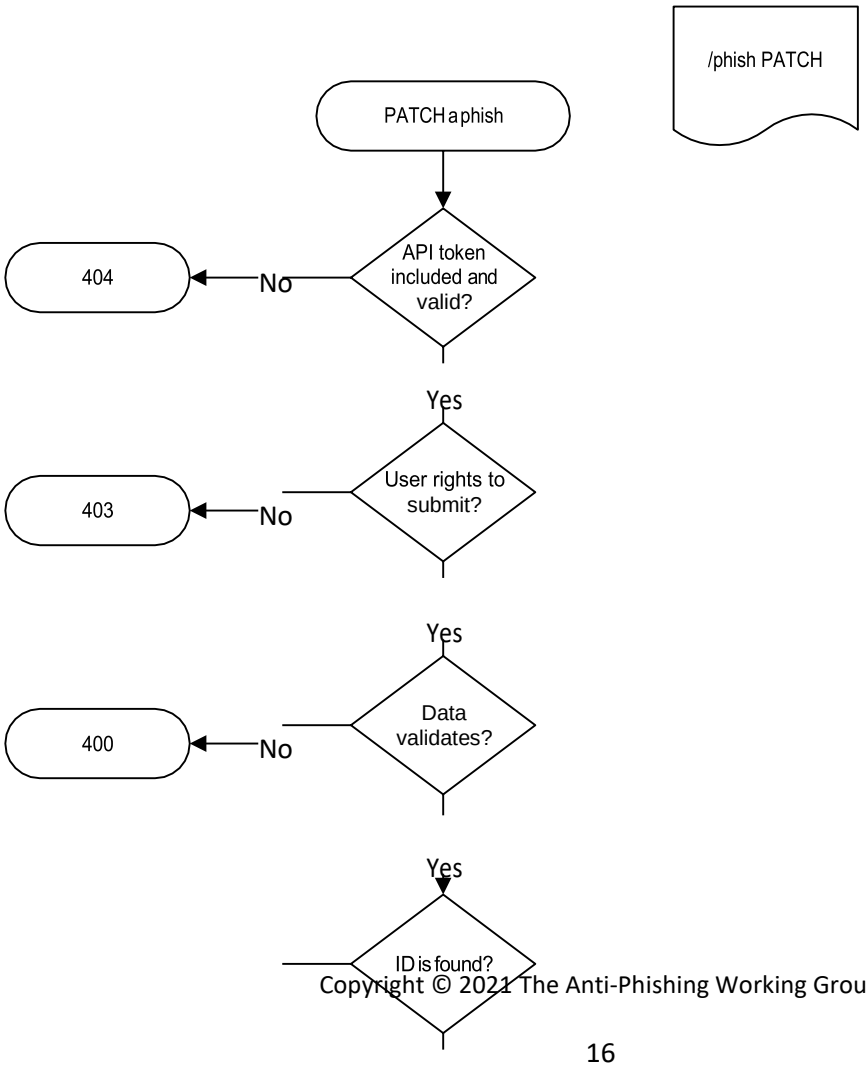
Phishing Module: PATCH

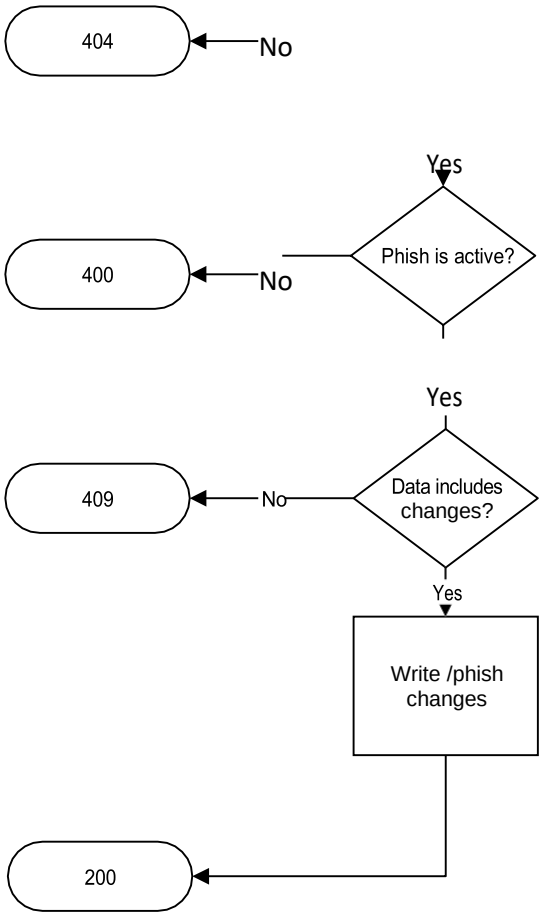
A PATCH requires the ID of the entity. Use PATCH when you find that you have additional or revised data for a specific phishing record within eCX.

You are able to update data via PATCH for brands, confidence_level, and status. In order to PATCH a phish entity the status must be active, and in order to change a status to active you must include a date_discovered value in the PATCH. You cannot PATCH a URL value, once POSTed to eCX the URL value is immutable.

Set the Content-Type header to 'application/json' to tell eCX that you are sending JSON data with the request.

When we POSTed the example below, the eCX returned an ID of 87104669 for the new entity that was created. We'll need this ID for the PATCH operation. You may update any or all of the modifiable fields using a PATCH, in our example we'll update the brand field, looking for a 200 result code being returned to indicate success.





PATCH using PHP Using cURL Example:

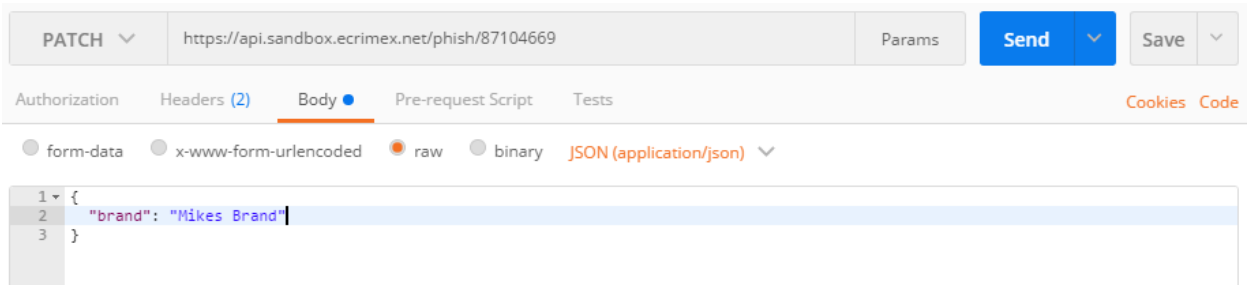
```
<?php

$curl   =

curl_init ( ) ;

curl_setopt_array($

curl,   array (
  CURLOPT_URL =>
  "https://api.sandbox.ecrimex.net/phish/87104669",
  CURLOPT_RETURNTRANSFER   =>  true,
```

```
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT  => 30,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST   =>   "PATCH",
  CURLOPT_POSTFIELDS => "{\r\n \"brand\": \"Daves
  Brand\"\r\n}", CURLOPT_HTTPHEADER   =>   array (
    "Authorization: <your eCX API Token
    Key goes here>", "Content-Type:
    application/json"
    ) ,
) ) ;

$response   =   curl_exec ($curl);
$err =

curl_error($curl)

;

curl_close($curl

);

if   ($err)   {
  echo "cURL Error #:" . $err;
} else {
  echo  $response;
}
```

PATCH Using cURL CLI Example:

```
curl -X PATCH \ https://api.sandbox.ecrimex.net/phish/87104669 \
  -H 'Authorization: <your eCX API Token Key goes here>'  \
  -H 'Content-Type: application/json' \
  -d '{
  "brand":  "Daves Brand"
}'
```

PATCH Using Postman Example:



Phishing Module: GET

GET queries the eCX API for data. You can query for a complete result set, or for as little as one record. Below are different ways to retrieve data -- consider which ones are right for your purposes.

Query for the most recent data since a previous query:

This is a common use case. As a user you want to fetch new data since the last time you fetched data. The correct way to do this is to use the "mod_date_start" parameter. If you are pulling data once per day at the same time of day, for example midnight UTC, use the following query and update the mod_date_start with the current date minus one day:

```
https://api.ecrimex.net/phish?mod_date_start=1626760800
```

GET using PHP cURL Example:

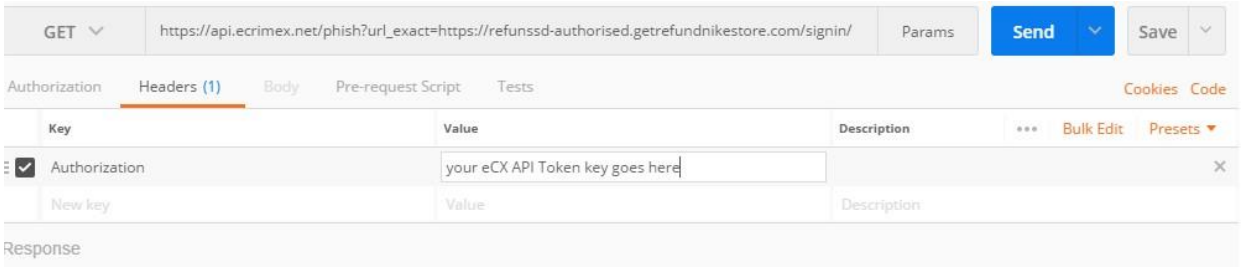
```
<?php

$curl   =

curl_init ( ) ;

curl_setopt_array($c

url,   array (
  CURLOPT_URL   =>
"https://api.sandbox.ecrimex.net/phish?url_exact=https://refun
ssd—  authorised.getrefundnikestore.com/signin/",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
```

```
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT  => 30,
  CURLOPT_HTTP_VERSION =>
  CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER   =>   array(
    "Authorization: <your eCX API Token Key goes here>"
  ),
));

$response   =   curl_exec($curl);
$err = curl_error($curl);
$http_status   =   curl_getinfo($http,

CURLINFO_HTTP_CODE); curl_close($curl);

if   ($err)   {
  echo "cURL Error #:" . $err;
} else {
  echo  $response;
}
```

GET using cURL CLI Example:

```
curl -X GET \
  'https://api.sandbox.ecrimex.net/phish?url_exact=https://refunssd-
  authorised.getrefundnikestore.com/signin/'    \
  -H 'Authorization: <your eCX API Token Key goes here>\
```

GET Using Postman Example:



Cryptocurrency Module

The Cryptocurrency module's API endpoint is: *https://api.ecx2.ecrimex.net/crypto/entry*

The Cryptocurrency Module was upgraded in August 2021. Here are some notes about the attendant changes that all users should be aware of:

- You must change your Authorization header value from just your API key by prefixing "Bearer" to the value. Please note the space after Bearer. This change is necessary because we've fixed the OAuth2 implementation and using a Bearer token is the proper standard.
    - Authorization: Bearer 1234567890abcdef1234567890abcdef
- The **Price** field was changed from a string to a number. Enter values in this field in the native value of the Crypto currency for the record. If you want to add values in USD or EUR then please do that in the Note field, via a PATCH.
- **Crime Category** is an optional field to submit. The field is now limited to the following values. If your submission does not fit into one of the given categories, please use "other".
    - (empty/null)
    - blackmail
    - crypto generator
    - darknet market
    - extortion
    - illicit account
    - other
    - ransomware
    - scam
    - tumbler
- The same **Address** can be used multiple times across entries (records). Because the same address can be used in multiple schemes and transactions, we allow duplicate address entries. This is unlike in the Phish module, which does not allow submission of the same URL more than once.

Posting Notes

The API endpoint to create a note is *https://api.ecx2.ecrimex.net/crypto/note* and the request body should contain two fields:

```
{
  "entry": "1154874",
  "description": "Note Body"
}
```

Querying the Crypto Module

You may use any combination of the following fields to perform a GET query against the Crypto module.

String Fields

- currency: The crypto currency the entry is valued in.
- source: Where the entry was found, e.g. bitcoinabuse.com
- procedure: One of automated or manual
- address: The wallet address of the currency
- crimeCategory: The crime category for the entry
- actorCategory: The actor category for the entry
- siteLink: Any URL used in the execution of the entry
- email: Any email address used in the execution of the entry

Number Fields

- id: The internal identifier of the entry. If you wish to query for records since a specific date use created
- price: The price sought in the native currency of the entry
- confidence: The confidence score of the entry

DateTime Fields: may accept a unix timestamp or an ISO 8601 date e.g. 2004-02-12T15:19:21+00:00

- timestamp: This is the Date Discovered
- created: The date the crypto entry was created
- updated: The date the crypto entry was updated. Aliased as modified.

Range Queries

- Suffix: any Number or DateTime field with _start or _end to create an inclusive range. For example: timestamp_start=123456789&timestamp_end=987654321

Query String

- Query: This special field is used for raw queries. See https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html#query-string-syntax

Sorting: available to only one column at a time

- order: one of asc or desc
- order_by: the field to sort on

Crypto Module Use Cases

Query for the most recent data since your previous query:

This is the most common use case. As a user you may want to fetch new records that have been submitted to the Crypto module since the last time you fetched data. The correct way to do this is to use the "created" field with the suffix _start. In the example below we're pulling data once per day at the same time of day (midnight UTC); use the following query and update the start date with the current date minus one day.

```
https://api.ecx2.ecrimex.net/crypto/entry?created_start=2021-07-
20T00:00:00
```

If you want to fetch everything that has been updated since your last query use the "modified" field:

```
https://api.ecx2.ecrimex.net/crypto/entry?modified_start=2021-07-
20T00:00:00
```

Malicious IP Module

The Malicious IP (/mal_ip) module is a repository of IP addresses that are considered dangerous or the sources of undesirable activity.

Due to GDPR, each member organization must sign the code of conduct document to get access to this module. Please contact support@ecrimex.net to request the paperwork.

The module support both IPv4 and IPv6 addresses. The IP field support only single IP addresses, and not ranges, such as an IPv4 /24 range.

Use the Description field to provide notes about what activity makes the IP address malicious, or other information that may be useful to other APWG members. Examples that our members have inserted include:

- exploited for large-scale scanning attacks
- suricata[84]: [1:2001219:20] ET SCAN Potential SSH Scan [Classification: Attempted Information Leak] [Priority: 2]
- [3:15474:8] SERVER-OTHER Microsoft ISA Server and Forefront Threat Management Gateway invalid RST denial of service attempt [Classification: Attempted Denial of Service]

Malicious Domain Module

The Malicious Domain (/mal_domain) module holds data about suspected or known malicious domains. These are domains registered by and being used by malefactors. In other words, they are domains that are “totally bad” and could be considered for blocking or suspension. If you believe a domain fits that description, it’s suitable for submission to this module. The module contains malware command-and-control domains, domains of fake stores and scam sites, etc.

The Domain field should contain a FQDN, such as *example.com* or *example.co.uk*. Do not include URLs, prefixes such as http://, and do not end the domains with a slash (/).

Report Phishing Module

The Report Phishing (/report_phishing) module is a little different from the others. It is a repository of phishing emails sent to the APWG via API and also by members of the public to the APWG's reportphishing@apwg.org mailbox. The records generally include full Internet headers and the bodies of the emails.

These submissions do not feature a Confidence Score. Because the repository contains emails sent to the reportphishing@apwg.org mailbox, which accepts submissions form the public, the module sometimes contains regular spam emails that advertise various scams and products, and contains some false-positives.


APWG

#