StopNCII API Reference

Guide for CSP's Integrating the StopNCII API

Date: 08/10/2025

Written By
Will Earp
will.earp@swgfl.org.uk
South West Grid for Learning

Contents

Overview

All methods described in here are based on REST API standards. Access to these REST API is protected through a provider specific key issued by SWGfL.

Required Headers

Every API call must include the two HTTP headers and they are:

Ocp-Apim-Subscription-Key

This key is once again used uniquely to identify the subscriber of the API function, and therefore should not be shared with any other CSPs. Each CSP gets their own Subscription Key for their use and should be used for all the functions called by the CSP.

x-functions-key

This key is equivalent of a password for the CSP to access the specific function.

This key is not only used for authorisation, but also used for identifying the CSP, and therefore should not be shared with other CSPs.

If any of these headers is not present, an HTTP error of `401 Unauthorized` is returned.

Endpoint Specific Headers

Content-Type: application/json

Where the endpoint method is POST, the CSP is required to request operations using a JSON payload. You must send a header indicating that the body contains JSON encoded data.

API Endpoints

FetchHashes

Fetch the (next) list of Hashes from the NCII Database.

Method

GET

Endpoint

https://api.stopncii.org/v1/FetchHashes

Parameters

`startTimestamp`

Specifies the time in Unix Epoch time format (number of seconds since January 1st, 1970, 12:00 Midnight), to select the list of hashes from the database, which has been created or modified since then.

Each CSP must maintain their "call time" to this API, so that when all hashes are returned, newer hashes can be fetched by specifying a new value for `startTimestamp`.

`pageSize`

Specifies the number of records to be returned for the API call. While there are no real restrictions on the total number of records to be returned, it is recommended to use a value of 1,000, to optimize performance.

`nextPageToken`

specifies the continuation of the previous call to this method. When results are returned, a value for this token is also returned as part of the result set. This new token value can be used to fetch the next batch of records. Repeated use of the same token only returns, possibly the same batch of records as the previous call (assuming that there were no changes to that batch of records). This token is updated with each new batch of records, therefore successive calls to this API should contain a newer value for `nextPageToken`. Alternatively, `nextPageToken` value should be set to empty string, with an updated value for `startTimestamp`. Any value for the parameter `startTimestamp` is ignored if a value is provided for the `nextPageToken`.

Return Values

The status of the call is checked through the returned HTTP status code. Following are the possible values:

HTTP Status Code	Description
`200 OK`	The API request was successful
`401 Unauthorized`	The provided `Ocp-Apim-Subscription-Key` or `x-functions-key` are invalid
`400 Bad Request`	Data passed to the endpoint is not valid
`500 Internal Server Error`	The server encountered an error when processing the request that prevented the operation from being completed

Along with the HTTP Status code, a string error message is written as part of the response body.

A Sample output of this API in JSON format, if successful is provided below:


```
{
  "count": 3,
  "nextSetTimestamp": 1625167824,
  "nextPageToken":
  "W3siY29tcG9zaXRlVG9rZW4iOnsidG9rZW4iOiIrUklEOn4xMnRWQUo4SzU2Y0N
  BQUFBQUFBQUFBPT0jUlQ6MSNUUkM6MiNSVEQ6SFBKSm1LVDF1WnFsQkJuR3lOU09
  CTUhaRytHL1FBPT0jSVNWOjIjSUVPOjY1NTY3",
  "hasMoreRecords": true,
  "hashRecords": [
    {
      "lastModtimestamp": 1625167804,
      "hashValue":
      "2afc4a5c09628a7961c14d436493bba66b89b831453baa1d556ba385554
      daa82",
      "hashStatus": "Received",
      "caseNumbers": {
        "27664732-76e1-4a17-8099-455798e67022": "Received",
        "a2696edf-1237-4eb8-a9c1-cd8ee6c055e5": "Received",
        "9fbb73d7-c177-4ed2-b4e8-0050b72decd0": "Received",
        "bd2bff7c-8160-4615-8dce-38c2f066d1ac": "Received"
      },
      "signalType": "ImagePDQ",
      "photoDNAHash":
      "461c929afe876d01e2157600ac25d32e4daa644e6c216ecc7737dee997d
```

```
      2a1afbca537bbc5a4053e1a1c02a00fd38a9d294c643c2fe470639347b68
      d9eb0f4edf2fabc72c81aef58"
    },
    {
      "lastModtimestamp": 1625167844,
      "hashValue":
      "79e07de27d7295339435d63cd31cf35a7bfa29eb2885008500a588a5ea3
      ae75a",
      "hashStatus": "Received",
      "caseNumbers": {
      "bd2bff7c-8160-4615-8dce-38c2f066d1ac": "Received"
    },
      "signalType": " ImagePDQ"
    },
    {
      "lastModtimestamp": 1625175071,
      "hashValue":
      "9def0b7dafa86a1c90f2abd78e79ceb25ec3d1a4b3d4bc7a4354baf7717
      ea038",
      "hashStatus": "Active",
      "caseNumbers": {
        "cc592711-7068-442f-b5d2-24d50d389751": "Active"
      },
      "signalType": " ImagePDQ",
      "CSPFeedbacks": [
        {
          "source": "Facebook",
          "feedbackValue": "Blocked",
          "tags": [
            "Nude",
            "Objectionable"
          ]
        }
      ]
    }
  ]
}
```


The parameters received in the response are as follows:

`nextSetTimestamp`
Indicates the Unix timestamp to be used for next API call, if a fresh API call is made.

`count`
Indicates the number of records returned by this query.

`hasMoreRecords`
If the property value is set to `true`, then there are additional records available, beyond what is returned. This usually occurs when the remaining number of records meeting the specified criteria in the API call exceeds the `pageSize`.

`hashRecords`

An array of Hash records from the NCII database. Properties of a `hashRecord` are:

- `lastModtimestamp`: timestamp in Unix Epoch time, indicating when the record was last modified.
- `hashValue`: Hash value as generated by the specific algorithm, such as PDQ, TMK, MD5 etc.
- `hashStatus`: Indicates the overall status of the `hashRecord` that can be readily used by the caller. Valid values for `hashStatus` are:
    - `Unknown` – Indicates an out-of-sync state from the server
    - `Received` – The hash was received from a user submission
    - `Active` – The hash was submitted by an NGO where the content has been verified by a human
    - `Withdrawn` – The user has indicated the hash has been withdraw and it will move to deleted within 24hours
    - `Deleted` – The hash remains in a deleted state for 24 hours before being removed
- `caseNumbers`: Is an array of Key-Value Pair, with Key indicating the Case ID in GUID/UUID format and value is the `CaseStatus`, same as `hashStatus`, but for the individual case
- `signalType`: Indicates the type of file and the hashing method used. Valid values for are: `Unknown`, `ImagePDQ`, `VideoTMK`, `VideoMD5`, `Text`, `URL`
- `photoDNAHash`: If a PhotoDNA hash has been generated (image hashes only), the 144 byte PhotoDNA hash will be attached

`CSPFeedbacks`

An array of feedbacks received from zero or more CSPs for the Hash. Each item has the following properties:

- `source`: Indicates the source from which the feedback was recorded
- `feedbackValue`: Indicates the feedback from the CSP as to how they handled the match, valid values for `feedbackValue` are: `None`, `QualityUnknown`, `PendingReview`, `Blocked`, `NotBlocked`, `Withdrawn`, `Deleted`, `Unknown`
- `tags`: A List of strings, in free format, provided by the CSP, who recorded the feedback

Example

https://api.stopncii.org/v1/FetchHashes?startTimestamp=1623708077&pageSize=1000&nextPageToken=W3siY29tcG9zaXRlVG9rZW4iOnsidG9rZW4iOiIrUklEOn4xMnRWQUo4SzU2Y3FBQUFBQUFBQUFBPT0jUlQ6MSNUUkM6MzkjUlREOkhQSkptS1QxdVpxbEJCbkd5TlNPQk1IWkdTc3piQT09I0lTVjoyI0lFTzo2NTU2

Usage

Fetches Hashes from the Database, whose timestamps are newer than the `startTimestamp` timestamp value.

A caller can optionally specify the maximum number of hashes that can be returned per call. If no value is provided, a default of `1000` is used. In any case, the value of `pageSize` cannot exceed 1,000. If there are more than the `pageSize` of hashes present in the database for the given "starting time" the results also return a JSON property named `nextSetTimestamp`, which can be used for fetching the next batch of hashes.

Although there is theoretically no limit to when the next page can be requested, it is advised that the next fetch of hashes is performed within 24 hours of the first call. If a Hash is deleted in a subsequent page, between the time of two calls, any deleted hashes are not returned to the caller.

Hashes returned by this API call are sorted in the ascending order of `lastModtimestamp` property. It is recommended that the caller repeatedly call this API, until the value of `hasMoreRecords` property is set to `false` to ensure that all hashes are received by the caller. The `startTimestamp` for next independent call should be set to with the value of `nextSetTimestamp`.

It is also important to note that whenever the `nextPageToken` is not an empty string, the value of `nextSetTimeStamp` is ignored, and therefore, the call is treated as a continuation from the previous call, trying to fetch the next page of data.

It is possible that a single hash may appear more than once in successive calls, indicating change in status of the hash. CSPs must take adequate steps to record change in the hash status. Example of this would a hash in the state of `Active`, going to `Withdrawn` or `Deleted`, if all associated cases are withdrawn and deleted respectively.

It is also worth noting that even if two different hashing algorithms are used on a single content file, if the resulting hashes are different, then they are maintained as separate hashes in the database. This is because the StopNCII System does not have any visibility into the actual content itself, as the hashes are generated by the client-side browser.

Also, it is nearly impossible to identify an image from a hash. One could argue that since we gather the filename, we could make an intelligent guess that two different hashes are for the same content; however, this argument assumes that the filenames are globally unique, which is never the case. Even for the same user, the hashes generated for the same content could be submitted under two different submissions.

SubmitHashes

Submit hashes to the NCII database.

Method

POST

Endpoint

https://api.stopncii.org/v1/SubmitHashes

Parameters

The hashes are submitted to the endpoint by supplying the hashes as a JSON encoded body with the following parameters:

`count`

The number of hashes to be submitted.

`hashRecords`

An array of hash record objects, which have the following properties:

- `lastModtimestamp`: can either contain the time of Hash creation or when it was modified. This entry is preserved as is in the NCII database, with the property of when the hash was created, instead of using system time at the NCII Server
- `hashValue`: The calculated hash
- `signalType`: Indicates the type of file and the hashing method used. Valid values for are: `Unknown`, `ImagePDQ`, `VideoTMK`, `VideoMD5`, `Text`, `URL`
- `photoDNAHash`: If the hash being submitted is an image, and you have the ability to generate a PhotoDNA hash, attach hash here alongside the base PDQ hash sent with `hashValue`
- `CSPFeedbacks`: An array of feedback objects with the following keys (Note that only one entry should be provided per hash record):
    - `feedbackValue`: Indicates the feedback from the CSP as to how they handled the match, valid values for `feedbackValue` are: `None`, `QualityUnknown`, `PendingReview`, `Blocked`, `NotBlocked`, `Withdrawn`, `Deleted`, `Unknown`
    - `tags`: A List of strings, indicating the reasons for the `feedbackValue`

Example

```
{
  "count": 2,
```

```
  "hashRecords": [
    {
      "lastModtimestamp": 1623711941,
      "hashValue":
      "fd91d9317550d66435049bb3e193eaa8226c2eeac0fb08bb1b6d24d4d77
      4d127",
      "signalType": "ImagePDQ",
      "CSPFeedbacks": [
        {
          "feedbackValue": "Blocked",
          "tags": [
            "Nude",
            "Objectionable"
          ]
        }
      ]
    },
    {
      "lastModtimestamp": 1623711941,
      "hashValue":
      "2ecc15d0aea44f464350aea5555013b19cae435254afaeaf537abcaf50b
      b515b",
      "signalType": "ImagePDQ",
      "CSPFeedbacks": [
        {
          "feedbackValue": "NotBlocked",
          "tags": [
            "Political Free Speech"
          ]
        }
      ]
    }
  ]
}
```

Return Value

The output for this API call is a simple JSON structure containing the Unique Internal identifier for the submission. This can be used in future to retrieve the list of hashes submitted in this batch to the platform, or possibly withdraw all the hashes in this batch[1].

```
{
  "batchId": "332ca84c-fdcc-4d4f-a5db-1e77f01a14e0"
}
```

[1] This feature is currently not implemented. Future versions may have this functionality.

The return status of the call is provided through the HTTP Return Status code. Following table contains the list of possible values for the return status code:

HTTP Status Code	Description
`200 OK`	The API request was successful
`401 Unauthorized`	The provided `Ocp-Apim-Subscription-Key` or `x-functions-key` are invalid
`400 Bad Request`	Data passed to the endpoint is not valid
`500 Internal Server Error`	The server encountered an error when processing the request that prevented the operation from being completed

Along with the HTTP Status code, a string error message is written as part of the response body.

Usage

This API is used by CSPs to submit their hashes to the NCII Hash databank, for cross industry hash sharing. The data set sent to NCII is similar to the data received when fetching hashes from NCII Database.

Even though there is no theoretical limit to the number of hashes that can be submitted in a single call, it is highly recommended to submit in a batch size of no more than 1,000.

If the hash is already present in the NCII database, an entry is added to the Hash indicating that this hash was provided the calling CSP as well, so that the reference count for the hash is incremented.

SubmitFeedback

Indicate to the system that you have matched one or more images.

Method

POST

Endpoint

https://api.stopncii.org/v1/SubmitFeedback

Parameters

`count`
The number of hashes to be submitted.

`hashFeedbacks`
An array of hash feedbacks to submit. Each item has the following properties:

- `hashValue`: The primary hash of the image you matched. The hash must be the hash from our database (from the `hashValue` key), not the hash you generated in the match
- `feedbackValue`: Indicates the feedback from the CSP as to how they handled the match, valid values for `feedbackValue` are: `None`, `QualityUnknown`, `PendingReview`, `Blocked`, `NotBlocked`, `Withdrawn`, `Deleted`, `Unknown`
- `tags`: A List of strings, indicating the reasons for the `feedbackValue`

Example


```
{
  "count": 3,
  "hashFeedbacks": [
    {
      "hashValue":
      "e066c399e80570c69f118f1167c639f69b19e0efc267b919384666468f3
      917fa",
      "feedbackValue": "NotBlocked",
      "tags": [
        "Tag-One",
        "Tag-eight"
      ]
    },
    {
      "hashValue":
      "11e6ce9971466699ba210ae647dc3966ee993066ce191be6e81fc699316
      3ce99",
      "feedbackValue": "PendingReview",
      "tags": [
        "Tag-One",
        "Tag-eight"
      ]
    },
    {
      "hashValue":
      "b867ae158f1c21f121f0079007d031f01f90071e703f787ff01ff01ff03
      ff03f",
      "feedbackValue": "Blocked",
```

```
      "tags": [
        "Tag-One",
        "Tag-eight"
      ]
    }
  ]
}
```

Return Value

The return status of the call is provided through the HTTP Return Status code. Following table contains the list of possible values for the return status code:

HTTP Status Code	Description
`200 OK`	The API request was successful
`401 Unauthorized`	The provided `Ocp-Apim-Subscription-Key` or `x-functions-key` are invalid
`400 Bad Request`	Data passed to the endpoint is not valid
`500 Internal Server Error`	The server encountered an error when processing the request that prevented the operation from being completed

No body will be returned for the request.

Usage

This API is used by CSPs to submit feedback on one or more hashes, based on their internal reviews. Feedback on hashes is added to other, feedback already submitted by other CSPs, for those hashes. If a feedback entry is already present for a hash by the same CSP, then, the old feedback is replaced with the new feedback value.

Additionally, any tags added to the CSP feedback is also recorded, and if there are existing tags associated with the old feedback, they are also replaced with the new one provided.

DeleteFeedback

Delete feedback on the specified hashes.

Method

DELETE

Endpoint

https://api.stopncii.org/v1/DeleteFeedback

Parameters

`count`

The number of hashes where your feedback is to be deleted.

`hashes`

An array of hashes to delete the feedback on. Note this must be the primary hash of the record in the StopNCII database (from the `hashValue` key), not the hash generated by you for matching.

Example

```
{
  "count": 3,
  "hashes": [
    "0f641146474647e4c9365b36316b567cb04ad8b5bc3ce2c1fe1ba4a3c2f
    7a68b",
    "acecb213cd84c8885313cdecb2137ded7865b2d3cdec13134d8c5550f65
    24dec",
    "54e9fb675b4829758a2f56a099a5aa4cb6b59fe9428b61db9b4a8818a68
    2e2ed"
  ]
}
```

Return Value

The return status of the call is provided through the HTTP Return Status code. Following table contains the list of possible values for the return status code:

HTTP Status Code	Description
`200 OK`	The API request was successful
`401 Unauthorized`	The provided `Ocp-Apim-Subscription-Key` or `x-functions-key` are invalid
`400 Bad Request`	Data passed to the endpoint is not valid
`500 Internal Server Error`	The server encountered an error when processing the request that prevented the operation from being completed

Along with the HTTP Status code, a string error message is written as part of the response body.

Usage

This API is used to remove the feedback on one or more hashes that the CSP has submitted earlier. This method removes the feedback from the hash completely, whereas only a history of feedback is maintained in the database for future auditing purposes. If there are no feedbacks from the submitting CSP attached to the specified hash, then the system will simply ignore the hash without generating any error code.

Input to this call must be submitted through the body of the HTTP Request in JSON encoded format and query parameters are ignored.

DeleteHashes API

Method

POST

Endpoint

https://api.stopncii.org/v1/DeleteHashes

Parameters

`hashes`

An array of hashes to delete. Note this must be the primary hash of the record in the StopNCII database (from the `hashValue` key) and must also be hashes that were submitted by your organisation.

Example

```
{
  "hashes": [
    "0f641146474647e4c9365b36316b567cb04ad8b5bc3ce2c1fe1ba4a3c2f7a
    68b",
    "acecb213cd84c8885313cdecb2137ded7865b2d3cdec13134d8c5550f6524
    dec",
    "54e9fb675b4829758a2f56a099a5aa4cb6b59fe9428b61db9b4a8818a682e
    2ed"
  ]
}
```

Return Value

The return status of the call is provided through the HTTP Return Status code. Following table contains the list of possible values for the return status code:

HTTP Status Code	Description
`200 OK`	The API request was successful
`401 Unauthorized`	The provided `Ocp-Apim-Subscription-Key` or `x-functions-key` are invalid
`400 Bad Request`	Data passed to the endpoint is not valid
`500 Internal Server Error`	The server encountered an error when processing the request that prevented the operation from being completed

Along with the HTTP Status code, a string error or success message is written as part of the response body:

```
{
  "statusCode": 200,
  "message": "Successfully deleted 3 hashes"
}
```

Usage

This API is used to withdraw hashes that were previously submitted through the `SubmitHashes` API.

Deleted hashes will remain available through the `FetchHashes` API for a minimum of 24 hours, where the hash status will be returned as `Withdrawn`. After this time, the hash will be deleted from the database.

When hashes are submitted through the `SubmitHashes` API, feedbacks are also attached to each hash submitted. Upon calling this endpoint, the feedback that was attached when the hash was created will be deleted immediately, and no longer appear in the `FetchHashes` API. Any other feedbacks attached to the hash will remain, but the hash will still be removed after 24 hours, preventing the retrieval of any other feedback after this time.