archiving-sample-php

A sample showing how to use the OpenTok 2.0 archiving API, written in PHP.

Prerequisites

- PHP 5.2 or higher. (Tested against 5.4.17).
- PHP must not have been compiled with `--disable-json=true`.
- PHP must be configured to allow `file_get_contents` to fetch remote URLs.
- An OpenTok API key, secret and session (from dashboard.tokbox.com).

To run this app for yourself

1. Clone this repo.
2. Copy config.php.sample to config.php (in the same folder) and set the following:
    - `config_api_key` -- Your OpenTok API key
    - `config_api_secret` -- Your OpenTok secret
    - `config_session_id` -- An OpenTok Session ID
    - `date_default_timezone_set` -- Set timezone to use for dates. See the PHP website for the list of supported timezones.
3. Copy this repo to the location where your web server can serve it. (The entire folder should be reachable, not just the public folder.)

Then time to test!

1. Open http://yourserver/path-to-this-folder/index.php in Chrome or Firefox 22+.
2. Click the "Host view" button. The Host View page publishes an audio-video stream to an OpenTok session. It also includes controls that cause the web server to start and stop archiving the session, by calling the OpenTok 2.0 archiving REST API.
3. Click the Allow button to grant access to the camera and microphone.
4. Click the "Start archiving" button. The session starts recording to an archive. Note that the red archiving indicator is displayed in the video view.
5. Open http://localhost:3052/ in a new browser tab. (You may want to mute your computer speaker to prevent feedback. You are about to publish two audio-video streams from the same computer.)
6. Click the "Participant view" button. The page connects to the OpenTok session, displays the existing stream (from the Host View page)
7. Click the Allow button in the page to grant access to the camera and microphone. The page publishes a new stream to the session. Both streams are now being recorded to an archive.
8. On the Host View page, click the "Stop archiving" button.
9. Click the "past archives" link in the page. This page lists the archives that have been recorded. Note that it may take up to 2 minutes for the video file to become available (for a 90-minute recording).
10. Click a listing for an available archive to download the MP4 file of the recording.

Known issues

- Recordings of streams from mobile devices do not adjust video orientation when the device orientation changes.
- Archive recordings are MP4 files with H.264 video and AAC audio. Some browsers, such as Firefox on Mac OS, do not support playback of this format in an HTML video tag. As a workaround, you can use Adobe Flash Player to load the video.
- The URLs for recorded archives are not secure, and they may change during or after the beta period.

Documentation

- Archiving REST API documentation
- Archiving JavaScript API documentation

More information

The OpenTok 2.0 archiving feature is currently in beta testing.

If you have questions or to provide feedback, please write denis@tokbox.com.

OpenTok 2.0 Archiving REST API

The OpenTok 2.0 archiving feature lets you record OpenTok 2.0 sessions.

This API only works with OpenTok 2.0 sessions. OpenTok 1.0 sessions are not supported.

Note: This documentation is a preview of the OpenTok 2.0 beta archiving REST API.

When you record an archive of a session, the resulting video is an H.264-encoded MP4 file, available for download.

The REST APIs include functions for the following:

Starting an archive recording

To start recording an archive of an OpenTok 2.0 session, submit an HTTP POST request.

Clients must be actively connected to the OpenTok session for you to successfully start recording an archive.

You can only record one archive at a time for a given session. You can only record archives of OpenTok server-enabled sessions; you cannot archive peer-to-peer sessions.

HTTP POST to archive

Submit an HTTP POST request to the following URL:

```
https://api.opentok.com/v2/partner/<api_key>/archive/
```

Replace `<api_key>` with your OpenTok API key. See https://dashboard.tokbox.com.

POST header properties

API calls must be authenticated using a custom HTTP header: X-TB-PARTNER-AUTH. Send your API key and API secret concatenated with a colon:

```
X-TB-PARTNER-AUTH: <api_key>:<api_secret>
```

(See https://dashboard.tokbox.com.)

Set the Content-type header to application/json:

```
Content-Type:application/json
```

POST data

Include a JSON object of the following form as the POST data:

```
{
    "sessionId" : "session_id",
    "action" : "start",
    "name" : "archive_name"
}
```

The JSON object includes the following properties:

- `sessionId` -- The session ID of the OpenTok session you want to start archiving
- `action` -- Set this to "start". (This is case-sensitive.)
- `name` -- (Optional) The name of the archive (for your own identification)

Response

The raw data of the HTTP response, with status code 200, is a JSON-encoded message of the following form:

```
{
  "createdAt" : 1384221730555,
  "duration" : 0,
  "id" : "b40ef09b-3811-4726-b508-e41a0f96c68f",
  "name" : "The archive name you supplied",
  "partnerId" : 234567,
  "reason" : "",
  "sessionId" : "flR1ZSBPY3QgMjkgMTI6MTM6MjMgUERUIDIwMTN",
  "size" : 0,
  "status" : "started",
  "url" : null
}
```

The JSON object includes the following properties:

- createdAt -- The timestamp for when the archive started recording, expressed in milliseconds since the Unix epoch (January 1, 1970, 00:00:00 UTC).
- partnerId -- Your OpenTok API key.
- sessionId -- The session ID of the OpenTok session being archived.
- id -- The unique archive ID. Store this value for later use (for example, to stop the recording).
- name -- The name of the archive you supplied (this is optional)

The HTTP response has a 400 status code in the following cases:

- You do not pass in a session ID or you pass in an invalid session ID.
- You do not pass in an action or pass in an invalid action type. To start recording an archive, set the action to "start". (This is case-sensitive.)
- No clients are actively connected to the OpenTok session.

The HTTP response has a 403 status code if you pass in an invalid OpenTok API key or partner secret.

The HTTP response has a 404 status code if the session does not exist.

The HTTP response has a 409 status code if you attempt to start an archive for a peer-to-peer session or if the session is already being recorded.

The HTTP response has a 500 status code for an OpenTok server error.

Example

The following command line example starts recording an archive for an OpenTok session:

```
api_key=12345
api_secret=234567890
session_id=2_MX40NzIwMzJ-flR1ZSBPY3QgMjkgMTI6MTM6MjMgUERUIDIwMTN-MC45NDQ2MzE2NH4
name="Foo"
data='{"sessionId" : "'$session_id'", "action" : "start", "name" : "'$name'"}'
curl \
     -i \
     -H "Content-Type: application/json" \
     -X POST -H "X-TB-PARTNER-AUTH:$api_key:$api_secret" -d "$data" \
     https://api.opentok.com/v2/partner/$api_key/archive/
```

- Set the values for `api_key` and `api_secret` to your OpenTok API key and partner secret.
- Set the `session_id` value to the session ID of the OpenTok 2.0 session you want to archive.
- Set the `name` value to the archive name (this is optional).

Stopping an archive recording

To stop recording an archive, submit an HTTP POST request.

Archives automatically stop recording after 90 minutes or when all clients have disconnected from the session being archived.

HTTP POST to archive

Submit an HTTP POST request to the following URL:

```
https://api.opentok.com/v2/partner/<api_key>/archive/<archive_id>
```

Replace `<api_key>` with your OpenTok API key. See https://dashboard.tokbox.com.

Replace `<archive_id>` with the archive ID. You can obtain the archive ID from the response to the API call to start recording the archive.

POST header properties

API calls must be authenticated using a custom HTTP header: X-TB-PARTNER-AUTH. Send your API key and API secret concatenated with a colon:

```
X-TB-PARTNER-AUTH: <api_key>:<api_secret>
```

(See https://dashboard.tokbox.com.)

Set the Content-type header to application/json:

```
Content-Type:application/json
```

POST data

Include a JSON object of the following form as the POST data:

```
{
    "action" : "stop"
}
```

Response

The raw data of the HTTP response, with status code 200, is a JSON-encoded message of the following form:

```
{
  "createdAt" : 1384221730555,
  "duration" : 60,
  "id" : "b40ef09b-3811-4726-b508-e41a0f96c68f",
  "name" : "The archive name you supplied",
  "partnerId" : 234567,
  "reason" : "",
  "sessionId" : "flR1ZSBPY3QgMjkgMTI6MTM6MjMgUERUIDIwMTN",
  "size" : 0,
  "status" : "stopped",
  "url" : null
}
```

The JSON object includes the following properties:

- createdAt -- The timestamp for when the archive started recording, expressed in milliseconds since the Unix epoch (January 1, 1970, 00:00:00 UTC).
- partnerId -- Your OpenTok API key.
- sessionId -- The session ID of the OpenTok session that was archived.
- id -- The unique archive ID.
- name -- The name of the archive you supplied (this is optional)

The HTTP response has a 400 status code in the following cases:

- You do not pass in a session ID or you pass in an invalid session ID.
- You do not pass in an action or pass in an invalid action type. To stop recording an archive, set the action to "stop". (This is case-sensitive.)

The HTTP response has a 403 status code if you pass in an invalid OpenTok API key or partner secret.

The HTTP response has a 404 status code if you pass in an invalid archive ID.

The HTTP response has a 409 status code if you attempt to stop an archive that is not being recorded.

The HTTP response has a 500 status code for an OpenTok server error.

Example

The following command line example stops recording an archive for an OpenTok session:

```
api_key=123456
api_secret=2345678
id=b40ef09b-3811-4726-b508-e41a0f96c68f
data='{"action" : "stop"}'
curl \
     -i \
     -H "Content-Type: application/json" \
     -X POST -H "X-TB-PARTNER-AUTH:$api_key:$api_secret" \
     -d "$data" \
     https://api.opentok.com/v2/partner/$api_key/archive/$id
```

- Set the values for `api_key` and `api_secret` to your OpenTok API key and partner secret.
- Set the `id` value to the archive ID. You can obtain the archive ID from the response to the API call to start recording the archive.

Listing archives

To list archives for your API key, both completed and in-progress, submit an HTTP GET request.

HTTP GET to archive

Submit an HTTP GET request to the following URL:

```
https://api.opentok.com/v2/partner/<api_key>/archive?offset=<offset>&count=<count>
```

Replace `<api_key>` with your OpenTok API key. See https://dashboard.tokbox.com.

Replace `<offset>` with the index offset of the first archive. 0 is offset of the most recently started archive. 1 is the offset of the archive that started prior to the most recent archive. Setting `<offset>` is optional; the default is 0.

Replace `<count>` with the number of archives to be returned. Setting `<count>` is optional. The default number of archives returned is 50 (or fewer, if there are fewer than 50 archives). The maximum number of archives the call will return is 1000.

Replace `<api_key>` with your OpenTok API key. See https://dashboard.tokbox.com.

GET header properties

API calls must be authenticated using a custom HTTP header: X-TB-PARTNER-AUTH. Send your API key and API secret concatenated with a colon:

```
X-TB-PARTNER-AUTH: <api_key>:<api_secret>
```

(See https://dashboard.tokbox.com.)

Response

The raw data of the HTTP response, with status code 200, is a JSON-encoded message of the following form:

```
{
  "count" : 2,
  "items" : [ {
    "createdAt" : 1384221730000,
    "duration" : 5049,
    "id" : "09141e29-8770-439b-b180-337d7e637545",
    "name" : "Foo",
    "partnerId" : 123456,
    "reason" : "",
    "sessionId" : "2_MX40NzIwMzJ-flR1ZSBPY3QgMjkgMTI6MTM6MjMgUERUIDIwMTN-
MC45NDQ2MzE2NH4",
    "size" : 247748791,
    "status" : "available",
    "url" : "http://tokbox.com.archive2.s3.amazonaws.com/123456/09141e29-8770-439b-
b180-337d7e637545/archive.mp4"
  }, {
    "createdAt" : 1384221380000,
    "duration" : 328,
    "id" : "b40ef09b-3811-4726-b508-e41a0f96c68f",
    "name" : "Foo",
    "partnerId" : 123456,
    "reason" : "",
    "sessionId" : "2_MX40NzIwMzJ-flR1ZSBPY3QgMjkgMTI6MTM6MjMgUERUIDIwMTN-
MC45NDQ2MzE2NH4",
    "size" : 18023312,
    "status" : "available",
    "url" : "http://tokbox.com.archive2.s3.amazonaws.com/123456/b40ef09b-3811-4726-
b508-e41a0f96c68f/archive.mp4"
  } ]
```

The JSON object includes the following properties:

- count -- The total number of archives for the API key.
- items -- An array of objects defining each archive retrieved. Archives are listed from the newest to the oldest in the return set.

Each archive object (item) has the following properties:

- createdAt -- The timestamp for when the archive started recording, expressed in milliseconds since the Unix epoch (January 1, 1970, 00:00:00 UTC).
- duration -- The duration of the archive in seconds. For archives that have are being recorded (with the status property set "started"), this value is set to 0.
- id -- The unique archive ID.
- name -- The name of the archive you supplied (this is optional)
- partnerId -- Your OpenTok API key.
- reason -- For archives with the status "stopped", this can be set to "90 mins exceeded", "failure", "session ended", "user initiated". For archives with the status "failed", this can be set to "system failure".
- sessionId -- The session ID of the OpenTok session that was archived.
- status -- The status of the archive. This can be "started", "stopped", "available", or "failed".
- size -- The size of the MP4 file. For archives that have not been generated, this value is set to 0.
- url -- The URL of the available MP4 file. This is only set for archives with the status set to "available"; for other archives, this property is set to null.

The HTTP response has a 403 status code if you pass in an invalid OpenTok API key or partner secret.

The HTTP response has a 500 status code for an OpenTok server error.

Example

The following command line example retrieves information for all archives:

```
api_key=12345
api_secret=234567890
curl \
    -i \
    -X GET \
    -H "X-TB-PARTNER-AUTH:$api_key:$api_secret" \
    https://api.opentok.com/v2/partner/$api_key/archive
```

Set the values for `api_key` and `api_secret` to your OpenTok API key and partner secret.

Retrieving archive information

To retrieve information about a specific archive, submit an HTTP GET request.

You can also retrieve information about multiple archives. See Listing archives.

HTTP GET to archive

Submit an HTTP GET request to the following URL:

```
https://api.opentok.com/v2/partner/<api_key>/archive/<archive_id>
```

- Replace `<api_key>` with your OpenTok API key. See https://dashboard.tokbox.com.
- Replace `<archive_id` with the archive ID.

GET header properties

API calls must be authenticated using a custom HTTP header: X-TB-PARTNER-AUTH. Send your API key and API secret concatenated with a colon:

```
X-TB-PARTNER-AUTH: <api_key>:<api_secret>
```

(See https://dashboard.tokbox.com.)

Response

The raw data of the HTTP response, with status code 200, is a JSON-encoded message of the following form:

```
{
 "createdAt" : 1384221730000,
    "duration" : 5049,
    "id" : "09141e29-8770-439b-b180-337d7e637545",
    "name" : "Foo",
    "partnerId" : 123456,
    "reason" : "",
    "sessionId" : "2_MX40NzIwMzJ-flR1ZSBPY3QgMjkgMTI6MTM6MjMgUERUIDIwMTN-
MC45NDQ2MzE2NH4",
```

```
    "size" : 247748791,
    "status" : "available",
    "url" : "http://tokbox.com.archive2.s3.amazonaws.com/123456/09141e29-8770-439b-
b180-337d7e637545/archive.mp4"
}
```

The JSON object includes the following properties:

- createdAt -- The timestamp for when the archive started recording, expressed in milliseconds since the Unix epoch (January 1, 1970, 00:00:00 UTC).
- duration -- The duration of the archive in seconds. For archives that have are being recorded (with the status property set "started"), this value is set to 0.
- id -- The unique archive ID.
- name -- The name of the archive you supplied (this is optional)
- partnerId -- Your OpenTok API key.
- reason -- For archives with the status "stopped", this can be set to "90 mins exceeded", "failure", "session ended", "user initiated". For archives with the status "failed", this can be set to "system failure".
- sessionId -- The session ID of the OpenTok session that was archived.
- status -- The status of the archive. This can be "started", "stopped", "available", or "failed".
- size -- The size of the MP4 file. For archives that have not been generated, this value is set to 0.
- url -- The URL of the available MP4 file. This is only set for archives with the status set to "available"; for other archives, this property is set to null.

The HTTP response has a 400 status code if you do not pass in a session ID or you pass in an invalid archive ID.

The HTTP response has a 403 status code if you pass in an invalid OpenTok API key or partner secret.

The HTTP response has a 500 status code for an OpenTok server error.

Example

The following command line example retrieves information for an archive:

```
api_key=12345
api_secret=234567890
id=23435236235235235235
curl \
    -i \
    -X GET \
    -H "X-TB-PARTNER-AUTH:$api_key:$api_secret" \
    https://api.opentok.com/v2/partner/$api_key/archive/$archive
```

- Set the values for `api_key` and `api_secret` to your OpenTok API key and partner secret.
- Set the `id` value to the archive ID.

Deleting an archive

To delete an archive, submit an HTTP DELETE request.

You can only delete an archive which has a status of "available".

HTTP DELETE to archive

Submit an HTTP DELETE request to the following URL:

```
https://api.opentok.com/v2/partner/<api_key>/archive/<archive_id>
```

Replace `<api_key>` with your OpenTok API key. See https://dashboard.tokbox.com.

Replace `<archive_id>` with the archive ID.

DELETE header properties

API calls must be authenticated using a custom HTTP header: X-TB-PARTNER-AUTH. Send your API key and API secret concatenated with a colon:

```
X-TB-PARTNER-AUTH: <api_key>:<api_secret>
```

(See https://dashboard.tokbox.com.)

Response

An HTTP response with a status code of 204 indicates that the archive has been deleted.

The HTTP response has a 403 status code if you pass in an invalid OpenTok API key, invalid partner secret, or invalid archive ID.

The HTTP response has a 500 status code for an OpenTok server error.

Example

The following command line example deletes an archive:

```
api_key=12345
api_secret=234567890
id=b40ef09b-3811-4726-b508-e41a0f96c68f
curl \
     -i \
     -X DELETE \
     -H "X-TB-PARTNER-AUTH:$api_key:$api_secret" \
     https://api.opentok.com/v2/partner/$api_key/archive/$id
```

- Set the values for `api_key` and `api_secret` to your OpenTok API key and partner secret.
- Set the `id` value to the ID of the archive to delete.

Notification of when an archive status changes

You can register for notifications when archives' statuses change by submitting an HTTP POST request.

Each archive has one of the following statuses:

"started" -- The archive recording has started, but it has not completed.
- "stopped" -- The archive recording has completed, but the download file is not available.
- "failed" -- The archive recording failed.
- "available" -- The archive is completed and available for download.
- "deleted" -- The archive has been deleted.

HTTP POST to callback

Submit an HTTP POST request to the following URL:

```
https://api.opentok.com/v2/partner/<api_key>/callback/
```

Replace `<api_key>` with your OpenTok API key. See https://dashboard.tokbox.com.

POST header properties

API calls must be authenticated using a custom HTTP header: X-TB-PARTNER-AUTH. Send your API key and API secret concatenated with a colon:

```
X-TB-PARTNER-AUTH: <api_key>:<api_secret>
```

(See https://dashboard.tokbox.com.)

Set the Content-type header to application/json:

```
Content-Type:application/json
```

POST data

Include a JSON object of the following form as the POST data:

```
{
    "url":'http://example.com/archive-callback',
    "group": "archive",
    "event": "status"
}
```

The JSON object includes the following properties:

- `url` -- The URL on your server, which the OpenTok server will call when an archive status changes. Be sure to include the complete URL.
- `group` -- Set this to "archive".
- `event` -- Set this to "status".

The length of the JSON data cannot exceed 8096.

Response

The OpenTok server returns a 200 status code upon successfully setting the notification URL. The raw data of the HTTP response is a JSON-encoded message of the following form:

{ "id" : "95771", "uri" : "http://www.example.com"}

The JSON object includes the following properties:

- id -- The unique ID for the registered callback. Use this ID to delete a callback, using the DELETE method. (See Deleting a notification callback.)
- uri -- The callback URL.

The OpenTok server returns a 400 status code in the following cases:

- The JSON data you send is invalid.
- The JSON data does not include group, url, or event properties.
- The JSON data is longer than 8096 characters.

The OpenTok server returns a 403 status code if you send an invalid API key or API secret.

Callback HTTP requests

When an archive's status changes, the server sends HTTP POST requests to the URL you supply. The Content-Type for the request is application/json.

The data of the request is a JSON object of the following form:

```
{
    "id" : "b40ef09b-3811-4726-b508-e41a0f96c68f",
    "event": "archive",
    "createdAt" : 1384221380000,
    "duration" : 328,
    "name" : "Foo",
    "partnerId" : 123456,
    "reason" : "",
    "sessionId" : "2_MX40NzIwMzJ-flR1ZSBPY3QgMjkgMTI6MTM6MjMgUERUIDIwMTN-
MC45NDQ2MzE2NH4",
    "size" : 18023312,
    "status" : "available",
    "url" : "http://tokbox.com.archive2.s3.amazonaws.com/123456/b40ef09b-3811-4726-
b508-e41a0f96c68f/archive.mp4"
}
```

The JSON object includes the following properties:

- createdAt -- The timestamp for when the archive started recording, expressed in milliseconds since the Unix epoch (January 1, 1970, 00:00:00 UTC).
- duration -- The duration of the archive in seconds. For archives that have are being recorded (with the status property set "started"), this value is set to 0.
- id -- The unique archive ID.
- name -- The name of the archive you supplied (this is optional)
- partnerId -- Your OpenTok API key.
- reason -- For archives with the status "stopped", this can be set to "90 mins exceeded", "failure", "session ended", "user initiated". For archives with the status "failed", this can be set to "system failure".
- sessionId -- The session ID of the OpenTok session that was archived.
- size -- The size of the MP4 file. For archives that have not been generated, this value is set to 0.
- status -- The status of the archive. This can be "started", "stopped", "available", or "failed".
- url -- The URL of the available MP4 file. This is only set for archives with the status set to "available"; for other archives, this property is set to null.

Example

The following command line example sets a callback URL for archive status changes:

```
api_key=12345
api_secret=234567890
callback_url=http://www.example.com
curl \
    -i \
    -X POST \
    -H "Content-Type: application/json" \
    -H "X-TB-PARTNER-AUTH:$api_key:$api_secret" \
    -d "{'url':'"$callback_url"', 'group': 'archive','event': 'status'}" \
    https://api.opentok.com/v2/partner/$api_key/callback
```

Set the values for `api_key` and `api_secret` to the API key and partner secret provided to you when you signed up for an OpenTok account. Set the value for `callback_url` to the callback URL for status change notifications.

Listing notification callbacks

You can list the notification callback URLs you've registered by submitting an HTTP GET request.

HTTP GET to callback

Submit an HTTP GET request to the following URL:

```
https://api.opentok.com/v2/partner/<api_key>/callback/
```

Replace `<api_key>` with your OpenTok API key. See https://dashboard.tokbox.com.

GET header properties

API calls must be authenticated using a custom HTTP header: X-TB-PARTNER-AUTH. Send your API key and API secret concatenated with a colon:

```
X-TB-PARTNER-AUTH: <api_key>:<api_secret>
```

(See https://dashboard.tokbox.com.)

Response

The OpenTok server returns a 200 status code upon successfully setting the notification URL. The raw data of the HTTP response is a JSON-encoded message of the following form:

```
[
  {
    "id" : "85771",
    "url" : "http://www.example.com/foo",
    "event" :"status",
    "group" :"archive",
    "createdAt" :"2013-12-08 17:14:52.0",
    "updatedAt" :"2013-12-08 17:14:52.0"
  },
  {
    "id" : "95643",
```

```
    "url" : "http://www.example.com/bar",
    "event" :"status",
    "group" :"archive",
    "createdAt" :"2013-12-09 12:39:22.0",
    "updatedAt" :"2013-12-09 12:39:22.0"
  }
]
```

The JSON object includes an array of objects defining each callback. Each callback object contains the following properties:

- id -- The unique ID for the registered callback. Use this ID to delete a callback, using the DELETE method. (See Deleting a notification callback.)
- url -- The callback URL.
- event -- This is set to "status" for archiving callbacks. In addition to archiving callbacks, this JSON object lists any other OpenTok callbacks you have registered (for example, for OpenTok Cloud Raptor.)
- group -- This is set to "archive" for archiving callbacks.
- createdAt -- The time when the callback registration was created.
- updatedAt -- The time when the callback registration was last modified.

The OpenTok server returns a 403 status code if you send an invalid API key or API secret.

Example

The following command line example retrieves information about registered callbacks:

```
api_key=12345
api_secret=234567890
curl \
    -i \
    -X GET \
    -H "X-TB-PARTNER-AUTH:$api_key:$api_secret" \
    https://api.opentok.com/v2/partner/$api_key/callback
```

Set the values for `api_key` and `api_secret` to the API key and partner secret provided to you when you signed up for an OpenTok account.

Deleting a notification callback

You can delete a notification callback URLs you've registered by submitting an HTTP DELETE request.

HTTP DELETE to callback

Submit an HTTP DELETE request to the following URL:

```
https://api.opentok.com/v2/partner/<api_key>/callback/<callback_id>
```

Replace `<callback_id>` with the ID of the registered callback. See Notification of archive status changes and Listing notification callback URLs.

Replace `<api_key>` with your OpenTok API key. See https://dashboard.tokbox.com.

DELETE header properties

API calls must be authenticated using a custom HTTP header: X-TB-PARTNER-AUTH. Send your API key and API secret concatenated with a colon:

```
X-TB-PARTNER-AUTH: <api_key>:<api_secret>
```

(See https://dashboard.tokbox.com.)

Response

The OpenTok server returns a 204 status code upon successfully deleting the callback.

The OpenTok server returns a 204 status code upon successfully deleting the callback.

The OpenTok server returns a 403 status code if you send an invalid API key or API secret.

Example

The following command line example deletes a registered callbacks:

```
api_key=12345
api_secret=234567890
id=95771
curl \
    -i \
    -X DELETE \
    -H "X-TB-PARTNER-AUTH:$api_key:$api_secret" \
   https://api.opentok.com/v2/partner/$api_key/callback/$id
```

Set the values for `id` to the callback ID. See Notification of archive status changes and Listing notification callback URLs.

Set the values for `api_key` and `api_secret` to the API key and partner secret provided to you when you signed up for an OpenTok account.

OpenTok 2.0 Archiving JavaScript API

The OpenTok 2.0 archiving feature lets you record OpenTok 2.0 sessions.

The OpenTok 2.0 archiving feature requires that you use the OpenTok 2.2 JavaScript library. See the important notes in OpenTok 2.2 Archiving JavaScript changes.

The OpenTok JavaScript API includes events that are dispatched when the archive recording starts and when the archive recording stops:

- archiveStarted
- archiveStopped

While an archive is being recorded, each OpenTok publisher displays a recording notification indicator.

Use the OpenTok 2.0 Archiving REST API to start, stop, and manage archives.

archiveStarted event

The Session object dispatches an archiveStarted event when an archive of the session starts recording. The Session object also dispatches an archiveStarted event when you connect to a session that has a recording in progress.

The event object includes the following properties:

- id -- The archive ID.
- name -- The name of the ID.

The following code registers event listeners for the archiveStarted event:

```
session.addEventListener("archiveStarted", function (event) {
    console.log("Archive started."
    console.log("- id:" + event.archiveId)
    console.log("- name:" + event.name)
}
```

archiveStopped event

The Session object dispatches an archiveStopped event when an archive of the session stops recording.

The event object includes the following properties:

- id -- The archive ID.
- name -- The name of the ID.

The following code registers event listeners for the archiveStopped event:

```
session.addEventListener("archiveStopped", function (event) {
    console.log("Archive started."
    console.log("- id:" + event.archiveId)
    console.log("- name:" + event.name)
}
```

OpenTok 2.2 JavaScript changes

The OpenTok 2.0 archiving feature requires that you use a pre-release version of the OpenTok 2.2 JavaScript library:

```
<script src="http://static.opentok.com/webrtc/v2.2/js/TB.min.js"></script>
```

When released, the OpenTok 2.2 JavaScript library will include other new features in addition to archiving.

The OpenTok 2.2 JavaScript library includes some change that are not backward compatible with OpenTok 2.0:

- The `sessionConnected` event dispatched by the Session object -- The `streams` and `connections` properties are deprecated (and these are now empty arrays). For streams and connections in the session when you connect, the Session object dispatches individual `streamCreated` and `connectionCreated` events (for each stream and connection). However, the Session object does not dispatch a `connectionCreated` event for the client's own connection. For the client's own connection, the Session object dispatches a `sessionConnected` event only.
- The `streamCreated` event dispatched by the Session object -- The Session object dispatches the `streamCreated` event for streams published by other clients only. The event includes a new property: `stream` (a Stream object corresponding to the new stream). The `streams` property is deprecated.
- The `streamCreated` event dispatched by the Publisher object -- The Publisher dispatches the `streamCreated` event for the stream published by the Publisher object. The `stream` property of the event is a reference to the Stream object.
- The `connectionCreated` event dispatched by the Session object -- The ConnectionCreatedEvent class now has a `connection` property, and the `connections` property is deprecated. The Session object now dispatches the `connectionCreated` event for each connection added to the session, including other clients' connections that exist when you first connect.
- The `connectionDestroyed` event dispatched by the Session object -- The ConnectionDestroyedEvent class now has a `connection` property, and the `connections` property is deprecated. The Session object now dispatches the `connectionDestroyed` event for each connection that leaves the session while your client is connected.
- The `streamDestroyed` event dispatched by the Session object -- The Session object dispatches the `streamDestroyed` event for destroyed streams from other clients only. The event includes a new property: `stream` (a Stream object corresponding to the destroyed stream). The `streams` property is deprecated.

- The `streamDestroyed` event dispatched by the Publisher object -- The Publisher dispatches the `streamDestroyed` event when the stream published by the Publisher object is destroyed. The `stream` property is a reference to the Stream object. The default behavior of this event is to remove the Publisher from the HTML DOM. You can cancel the default behavior by calling the `preventDefault()` method of the event object.
- The `sessionDisconnected` event dispatched by the Session object -- Calling the `preventDefault()` method no longer causes a Publisher to be preserved. The Publisher is destroyed by default when the Publisher dispatches the streamDestroyed event. (You can preserve the Publisher by calling the `preventDefault()` method of the `streamDestroyed` event dispatched by the Publisher object.)
- In the `Session.signal()` method, the optional `to` property of the `signal` parameter takes a single Connection object (defining the connection to which the signal will be sent). The `to` property does not take an *array* of Connection objects, like it did in version 2.0 of the JavaScript library.