NAV

Clubhouse Outgoing Webhooks

Introduction

You can use webhooks to get notified about events that happen in your Clubhouse organization. If you have any trouble or questions about using our Outgoing Webhooks API, check out our **Help Center** or just **contact us** - we're happy to help!

Status

This is the v1 version of our Outgoing Webhook API. There may be breaking changes, but we will document them in a Change Log section on this page.

Use Cases

An Outgoing Webhook allows you to register a URL that Clubhouse notifies whenever any change happens in your Organization. Webhooks track changes made by users in your account, and by changes made via Token request. Webhooks allow you to use events that occur in Clubhouse to trigger events in other services.

Here are some ideas to get you started...

- If a Story is moved to Ready for QA, send a Slack message to the #ready-for-qa channel.
- If a Story is moved to Ready for Deploy, execute a CI build or integration test script.
- If a Story in a particular Project has a deadline is set, create a reminder in Slack three days before it's due.

NAV

- Log whenever Stories are created, started, or completed, and use that data to compile metrics for a company dashboard.

Webhook Format

Example Response


```
{
  "id": "595285dc-9c43-4b9c-a1e6-0cd9aff5b084",
  "changed_at": "2017-06-27T16:20:44Z",
  "primary_id": 16927,
  "member_id": "56d8a839-1c52-437f-b981-c3a15a11d6d4",
  "version": "v1",
  "actions": [
    {
      "id": 16927,
      "entity_type": "story",
      "action": "update",
      "name": "test story",
      "changes": {
        "started": {
          "new": true,
          "old": false
        },
        "workflow_state_id": {
          "new": 1495,
          "old": 1493
        },
        "owner_ids": {
          "adds": ["56d8a839-1c52-437f-b981-c3a15a11d6d4"]
        }
      }
    }
  ],
  "references": [
    {
      "id": 1495,
      "entity_type": "workflow-state",
      "name": "Ready for Deploy"
    },
    {
      "id": 1493,
      "entity_type": "workflow-state",
      "name": "Ready for Dev"
    }
```

```
  ]
}
```

The Webhook API sends a message about any change to a Story in your Organization. Each event has the following properties:

Name	Type
id	UUID
primary_id	UUID or Integer
member_id	UUID
changed_at	Date
actions	[Action, ...]
references	[Reference, ...]

Also included in the Webhook object is an **actions** array. This array allows you to access changes to objects. It will include changes to the primary object and object effected by the event.

The actions map follows a standard format. The individual action will have the id, entity type and the type of action. Included in the action map is a **changes** map. With the changes map you can access the old and new values of attributes that where changed in the event.

More documentation on this subject is to come. In the meantime, experimenting with an example server will be the fastest and easiest way to both understand what data we send and what data you might be interested in using.

Signature

If you provide a **secret** when you create the Outgoing Webhook, it will include an HTTP header named `Clubhouse-Signature`. The value of this header is a cryptographic hash encoded in hexadecimal.

NAV

The signature is computed by the **HMAC-SHA-256** algorithm. The 'message' is the HTTP request body encoded in **UTF-8**. The 'secret' is the secret string you provided, also encoded in UTF-8.

Example Server

Example Server

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();

app.use(bodyParser());

app.post('/webhook', function (req, res) {
  var event = JSON.parse(req.body);
  console.log(event);
  res.send(200);
})

app.listen(3000, function () {
  console.log('Example webhook app listening on port 3000!');
});
```

This example server written in Node.js will print each event object as it is received. This assumes you've installed the `express` npm library locally using `npm install express --save`.