In-Call Webhook

MyOperator

This document describes webhook in-call API for integrating MyOperator call event with other application.

Overview

Webhooks are HTTP callbacks that receive notification messages for events. MyOperator uses webhooks to notify your application any time an call event happens in your account. For example, if a call has been received by your agent, you will receive a webhook during the call (incall webhook), and after the call has finished (aftercall webhook).

In call webhook is the feature of MyOperator which is used to share the call data during the call through HTTP request. A call contains many events. For example- Call picked up, Call hangup etc. For each event, we invoke a webhook to your webhook listener.

Purpose

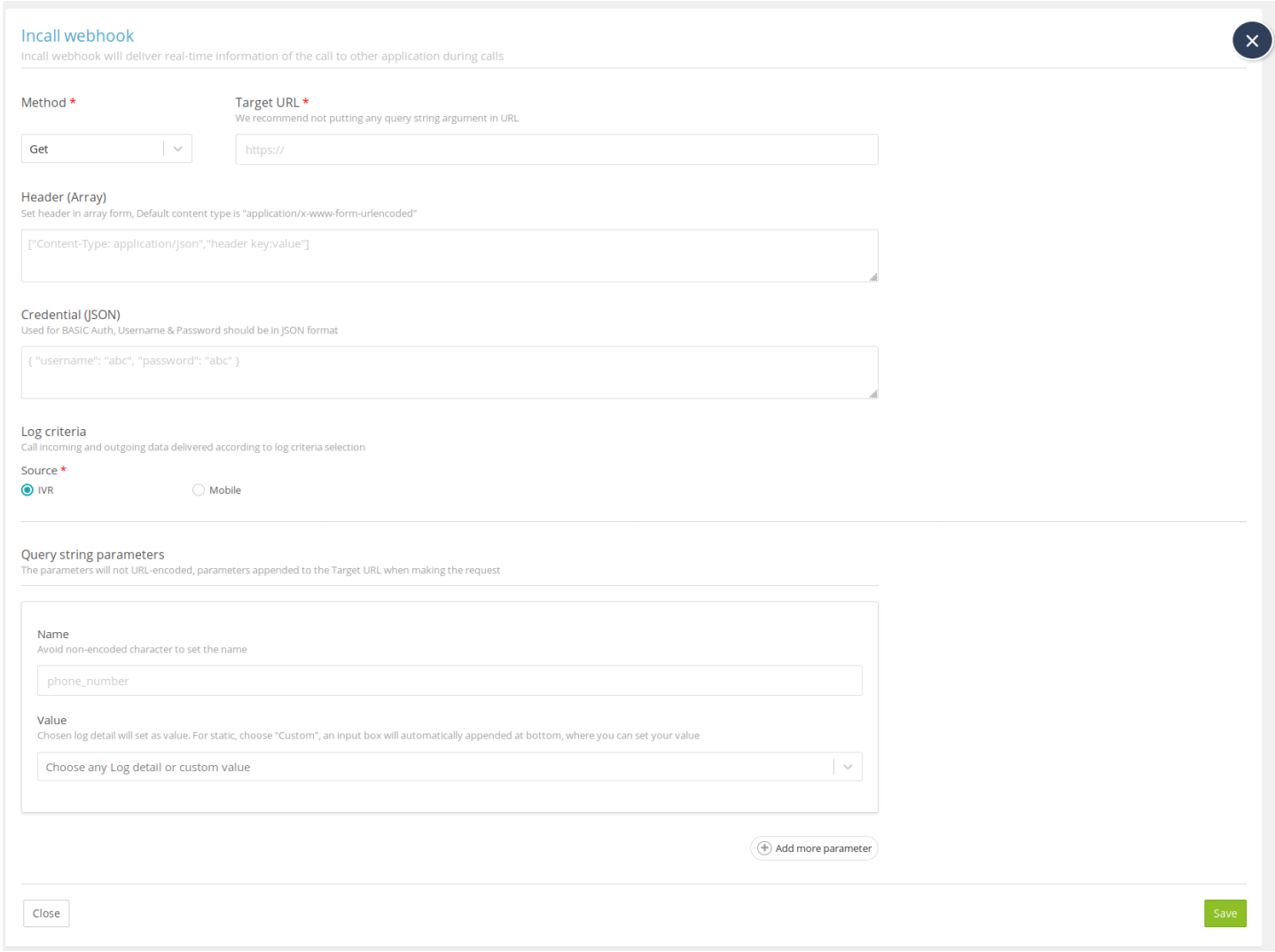
Webhook enables clients to do actions on the data at the end of every call. Webhook data can be used to do below things at client's end:

- Collect call data in datastore at client's end.
- Invoking custom integrations like adding a lead in the CRM.

The call log records all associated information related to given call. During each call, we send the same log via webhooks. How we send the log (in post body or GET params) depends on the type of webhook requested by client.

Components

Incall webhook can be configured inside MyOperator app in api integrations section. The page looks like this:



A webhook is made up of components. Each component allows you to configure certain part of webhook. This gives you flexibility of configuration each part of webhook separately.

Target URL

Target URL allows you to configure your webhook listener url. When you code your system, which will receive the data during each call, you give the url for that code in this input.

Example: `http://your-service-url.com/webhook.php`

Method

There are two type of methods supported by webhooks:

1. GET
2. POST

GET

GET Method, as name suggest, will send webhook using HTTP Get requests. This means, you will need to looks for `query parameters` in your GET requests for data. We use query parameters to send the data that you have set. Hence, the request looks something like this:

```
curl -X GET http://your-service-url.com/webhook.php?parameter1=<call log value>&parameter2=
<another call log value>
```

To get the above data in your code (PHP sample):

```
// If you have set Query string parameters, you can still get them here too.
$some_value = $_GET['parameter1'];
```

In above example, you can see that the `parameter1` and `parameter2` seem important. Thse parameters contains certain call log information. For example, `parameter1` may have caller's number, and `parameter2` might have call duration. To know how to configure these parameters in details, see Query parameter section.

POST

POST method sends a HTTP POST request to your webhook listener. With this method, you get the same data as GET method (query parameters), as well as, you get whole call data in POST body. This allows you to take additional decisions based on the call log. This is why we recommend you to use `POST` method.

We send the following data in `POST` format:

```
myoperator={
  "id": "5f25296c4122c135",
  "uid": "n2.1596270949.2175190",
  "reference_id": "fd771ed0-d3d1-11ea-935a-9a5b7644c4f0",
  "company_id": "5cd40f6554442586",
  "clid_raw": "919711636496",
  "clid": "9711636496",
  "rdnis": "unknown",
  "call_state": "1",
  "event": "2",
  "status": null,
  "users": [
    "918586848544"
  ],
  "created": "2020-08-01 08:35:56",
  "call_time": "2020-08-01 08:35:56",
  "public_ivr_id": "5ee9a795b318a786",
  "client_ref_id": "fdfdfdf",
  "job_id": null
}
```

MyOperatos sends the data in POST body inside `myoperator` key. A basic cURL request for it looks like this:

```
curl -X POST http://your-service-url.com/webhook.php -d myoperator=<call log>
```

NOTE - MyOperator sends the data in JSON format. So, you should json decode the data on your side to ensure correctness of it. See the example below.

You can get the call log data from a POST webhook like this (PHP code sample):

```
$call_data = $_POST['myoperator'];

/*
 The data is in this string format:
 {"_ai": "xyz", "_ci": "123abc", ...}
 Hence you should decode it to array to get particular call data
*/
$call_data_array = json_decode($call_data, true);

// get caller's number
$caller_number = $call_data_array['_cr'];

/*
 Remember, POST request doesn't mean you won't get query parameters.
 If you have set Query string parameters, you can still get them here too.
*/

$caller_numer = $_GET['caller_number'];
```

Header

This component allows you to send any custom headers with the webhook request. Headers is a way to pass any key value pair from our side.

Suppose, in your code, you want to verify if the request is actually coming from our side, and not from someone outside or hackers, you can easily check this using headers.

You can set any secret token in this component, which we will append in the webhook request before sending it to your service. Once you receive a request from us, you can check if the header has the secret token. If it does, you can confirm we are the one sending the request.

Example:

Suppose, you set the header:

```
["my-super-secret-token: abc123def"]
```

Then we will send this cURL request to your webhook listener:

```
curl -X POST http://your-service-url.com/webhook.php -H 'my-super-secret-token: abc123def'
```

which you can check like this (PHP code sample):

```
$my_secret_header = $_SERVER['HTTP_my-super-secret-token'];

if( $my_secret_header == 'abc123def' ) {
  // Request come from MyOperator webhook
```

```
  echo "Good";
} else {
  // Request didn't came from MyOperator webhook
  echo "Bad";
}
```

Credentials

Sometimes, you may have a service which requires authentication. There are several types of possible authentications, such as, OAuth, Basic, API Key based etc.

However, we only support BASIC auth for now. If your service has BASIC Auth enabled, You can configure the username and password in this component. The format we use is
`{"username": <your username>, "password": <password>}`

For example, if my service has "admin" as username and "123456" as password, I will enter following in the panel:

```
{"username": "admin", "password": "123456"}
```

When we detect that you have configured your credentials, we will invoke the webhook with credentials.

```
curl -u admin:123456 http://your-service-url.com/webhook.php
```

Log Criteria

This component allows you to choose which type of data you want to receive from webhook. Sometimes, you want the call log (from panel), while sometimes, you want to receive mobile logs only. In this component, you can only pick one out of two.

If you want both the call logs and mobile logs, you can create two webhooks - one for call log and one for mobile logs.

Query string parameters

This section is the most important of all. In this component, you can pick which part of call log you are interested in receiving.

This component allows you to mention a name, and map it to a particular call record value you are interested in. Once you choose a name and map it to a value, we will send the webhook to you with that name in the query parameters.

For example, lets say you are interested in receiving the caller's number. You can choose the name to be "caller_number" (do not put spaces and url encodeable values in the name). From the dropdown, pick "Caller number with country code". Once you save, we will send the webhook to your webhook listener like this:

```
curl -XGET http://your-service-url.com/webhook.php?caller_number=+919999999999
```

So, you can get the `caller_number` in your code (PHP code sample) like:

```
$caller_number = $_GET['caller_number'];
echo $caller_number; //+919999999999
```

The dropdown, which you have selected the value from, are called query parameters fields. The description of each field is documented in the Query parameter fields section.

NOTE: If you have set the Method as `POST`, we will have sent the data like this:

```
curl -XPOST http://your-service-url.com/webhook.php?caller_number=+919876543210 -d myoperator=
<call log>
```

Query parameter fields

Query parameters is a set of key value pairs. You configure which call log should be mapped to the key of your choosing. This section focus on explaining what each call log (from the dropdown) means, with an example.

Caller number with country code

This field gives the caller's mobile number, along with country code. For example: +919876543210

This field is same as `_cl` in Call log (the data that you receive in POST body)

Caller number without country code

This field gives the caller's mobile number, without country code. This basically means the raw call data we got in call records. For example: 9876543210

This field is same as `_cr` in Call log (the data that you receive in POST body)

Creation date of log (epoch)

This field gives you the time (in UNIX Epoch format), in UTC timezone, when the log was created. For example: 1577797873 (Tuesday, 31 December 2019 13:11:13)

Event of log

This gives you the type of call record. Possible values are:

- incoming
- outgoing

Status of log

This gives you the overall call status of record. Possible values are:

- Connected (call was picked by atleast one agent and for some non zero duration)
- Missed (call was dialed by never picked by any of the agents)
- Voicemail (call landed on voicemail)

Call state

This field gives you the state at which this webhook event has been triggered to you.

A call is a linear combination of several stages. Stage represents an event through which call progresses. First stage is call initialization. In case of outgoing call, we initiate a call from our system to your customer. In call of incoming call, we receive a call from customer to one of your agent or departments. Another stage is when call has been picked by one of your agents.

Below is a list of call events we support:

call state value	Description	Type of call
1	Incoming call on server	OBD + incoming
2	call finished	OBD + incoming
3	call initiated by user to customer	Click to call
4	first party no answer	OBD
5	dialing users	OBD + incoming
6	user answered	OBD + incoming
7	client rejected the call	Click to call
9	no user picked the call (no answer)	Click to call

UID

We use UID as a unique identifier for each call. Each call log has its own UID. It helps us trace the call and routes. An example UID looks like: s5.347638.78643857

This field is same as `_pm.ui` in Call log

Company ID

Company ID gives you the company in which call took place. It can help you get the company details and in which country the agent was in. It is alphanumeric in nature. Example: abc123.

This field is same as `_ci` in Call log

Service number

This gives you the company's service number whose agent picked up the call(in incoming cases) or dialed the number (in outgoing cases). For example- 919999999999

User's phone number

This gives you the agent's contact number who picked up the call(in incoming cases) or dialed the number (in outgoing cases). An agent represents a user within a company. Agent contact number contains 10 digits contact number, without country code. For example- 9999999999

This field is same as `_ld._rr._ct` in Call log

Custom

You can also set any type of custome key value pair to be included in the webhook request. Selecting this option will allow you to set a custom key to the name of your choosing.

Reference ID

The reference ID will be generated once you initiate any OBD campaign.

This field is same as `_ri` in Call log

Client reference ID

The client will pass a custom ID and MyOperator will return this as it.

This field is same as `_cri` in Call log

IVR ID

This is ID of IVR on which the call has been landed

This field is same as `_ivid` in Call log

Call Log

A call log is a hashmap (key-value pair) composed of several short field names with their values, related to call. The call log is in JSON format. We use short field name to save bandwidth and storage requirements. However, it may become difficult to understand the field meaning, hence this section explores the fields, with their meanings, as well as example value inside it.

Log field name	Meaning	Example
clid_raw	Caller's number (raw)	919999999999
clid	Caller's number (formatted)	+919999999999
created	Call log creation time	2016-11-23 13:26:15
event	Call event type	1=incoming, 2=outgoing
status	Call status	1
call_state	Call state	1
company_id	Company ID	abcdef123
rdnis	Service number	9999999999
uid	Call unique ID	sn.1572424727.994

Log field name	Meaning	Example
users	comma separated agent ids	abc123,def456
reference_id	reference ID generated in OBD process	fd771ed0-d3d1-11ea-935a-9a5b7644c4f0
public_ivr_id	IVR ID of campaign	5ee9a795b318a786
client_ref_id	reference ID given by the client	fdfdfdf

Need Help?

In case of any query please contact the support team at support@myoperator.co. Or generate a support ticket from MyOperator Panel.

Revision #16
★ Created Tue, Dec 31, 2019 11:39 AM by Ashutosh
Updated Thu, Feb 13, 2020 2:29 PM by Ashutosh