Discover the Power of Cloud

Integrate any software to Call Center Studio utilizing our open infrastructure and Public API.

Contact us

Take a look at our Public API

Call Center Studio

# CRM Integrations

Technical Guide

Have you tried WhatsApp Business?

Integrate WhatsApp Business with Call Center Studio to manage all messaging with your cutomers from a single platform. Communication is simplified, freeing your business up to do what it does best.

Moreover, all these services are free for Call Center Studio users!

CCS Integrations Team

March 2020

## Introduction

Call Center Studio is a technology company that aims to offer all the call center functions as a ''service'' to businesses that want to offer their customers an enterprise grade call center.

Many layers of the application have been designed to work integrated with the customer, so Call Center Studio customers go beyond their competitors in shaping the call center infrastructure according to their inquiries.

## Aim

Our target is to make sure that the call center staff leverage many Call Center Studio features such as answering calls, making a one-touch call, automatically accessing the customer record, viewing the call results via CRM etc.

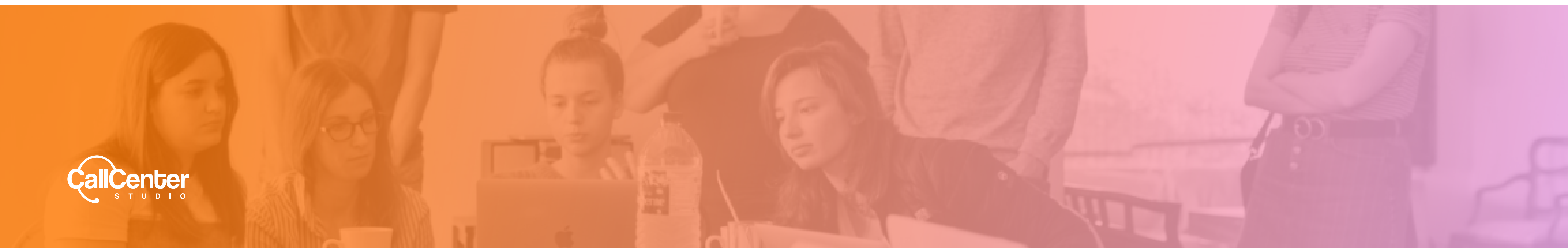
## APIs

**1. Login**: Restful API that is used for the agent to login to Call Center Studio.

**2. Generic Softphone:** Our 3rd party application for single page applications that can be recalled with iframe.

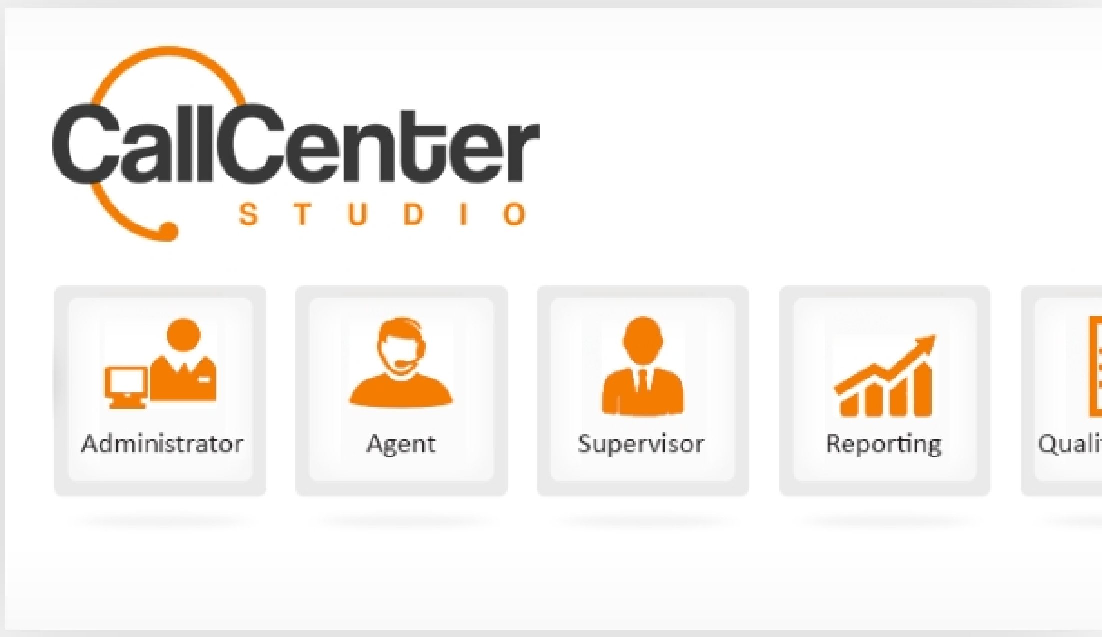
**3. click2call:** Restful API that allows the agent to initiate a call from anywhere, with the information of session and number to call.

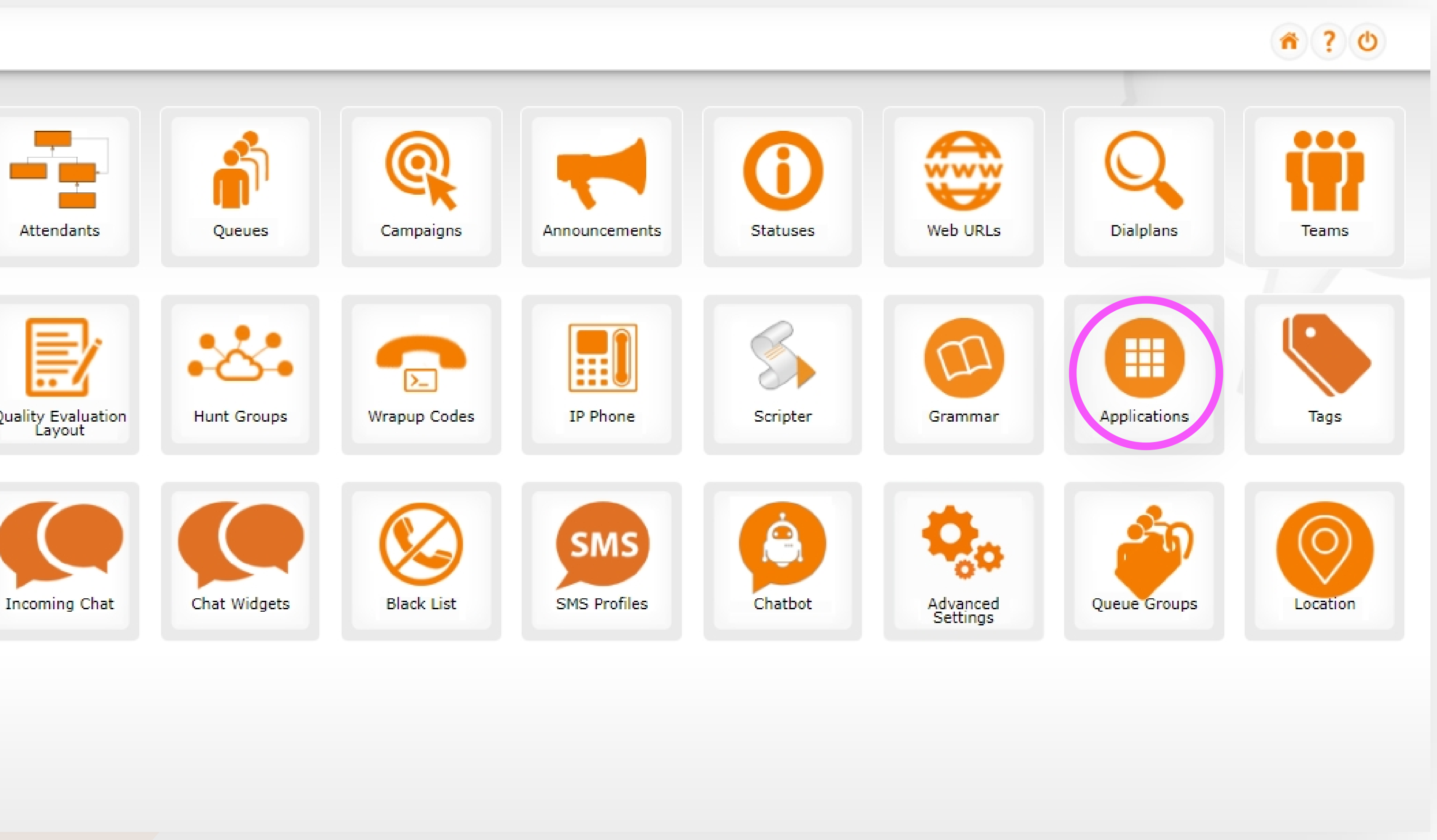
**4. reportsCDRLogs**: Restful API used to obtain CDR logs, where the information regarding the results of relevant calls are kept.



## Beginning

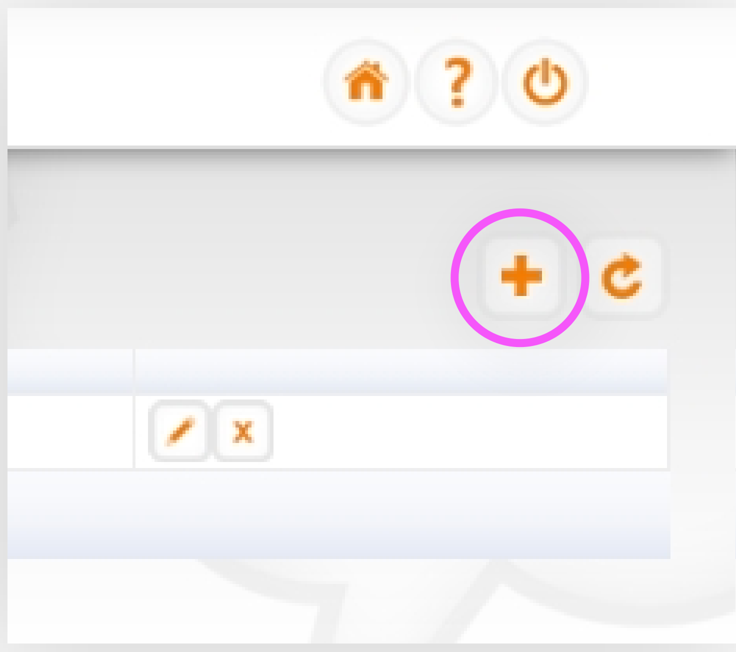
- The domain that you use as a subdomain on **tenantname**.callcenterstudio.com simply refers to the tenant name.

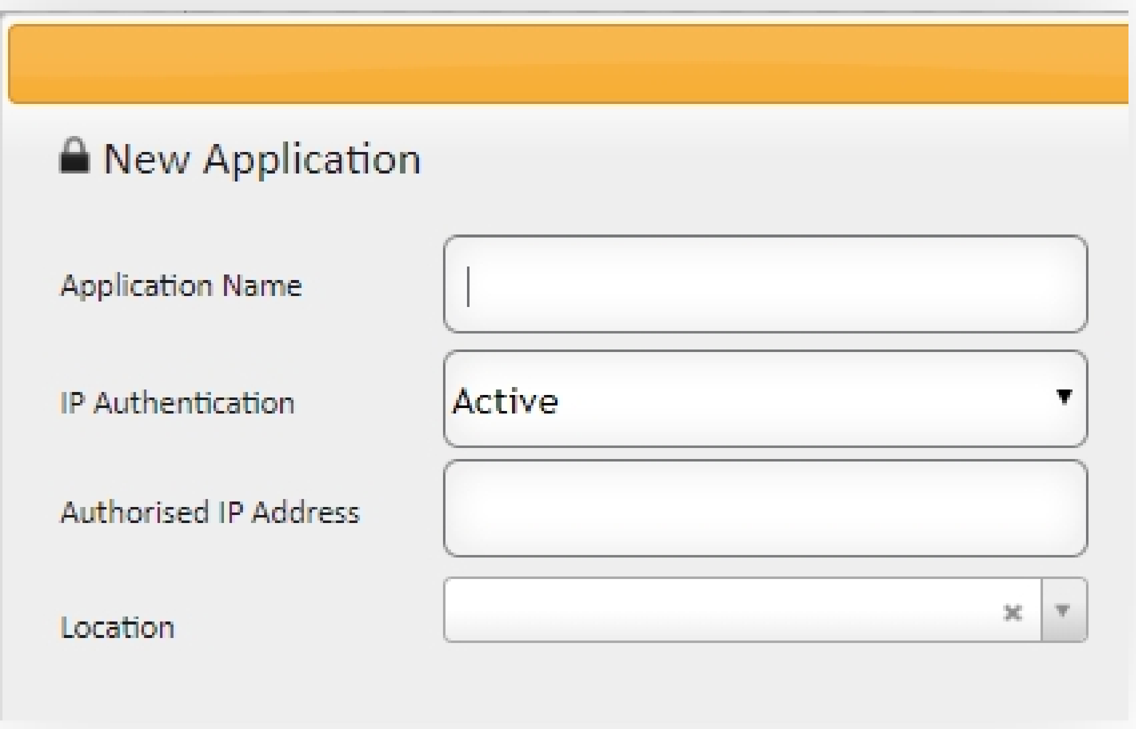




- To generate an app_token, please navigate to the "Administrator" module.
- Users then can choose the "Applications" tab within this interface to generate their app_token.

- Click on the "Plus" (+) button once you enter the applications menu to open up the "New Application" page

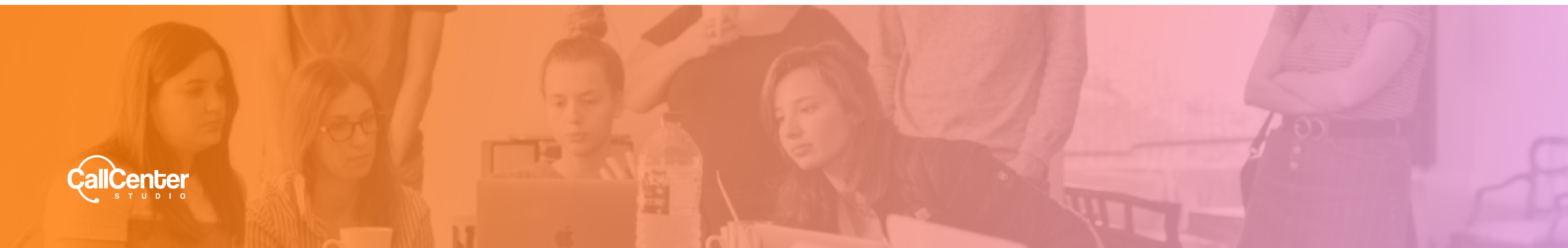




New Application

| | |
|---|---|
| Application Name | |
| IP Authentication | Active |
| Authorised IP Address | |
| Location | |

- All the required definitions are conducted on this box and saved via the save button.
- In order to see the saved App_token, the edit button, ✎ , of the relevant app_token on the interface must be clicked.



## Login

For the login process, you need to keep some global variables for your customers and local variables for users under a structure that you can construct in your CRM;

- The variables, that are for your registered customers in Call Center Studio and that will be defined globally for that customer in CRM, are listed below;

*1. Tenant Name:* It is a fixed value. It is the subdomain address which the customer receives Call Center Studio service. (The client may have more than one tenant. A situation we encounter in very large operations)

*2. Application Token:* It is not a fixed value. It can be deleted and updated with ease. It is useful to define it as a configurable area for security reasons.

- Variables defined in terms of users of your customers that will use your application;

*1. User email:* The email address used when logging in to Call Center Studio. It is a fixed value, yet I would recommend making it changeable.

- Variables to be limited in heap memory

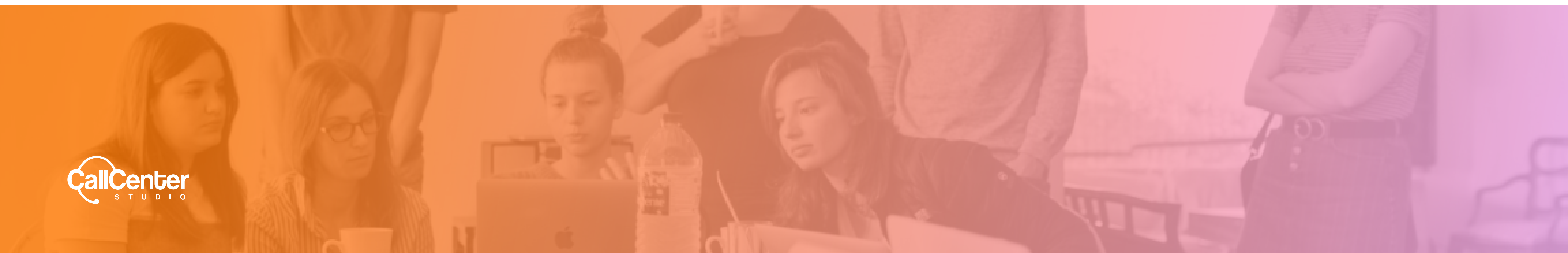
*1. Session Key (session):* The value of the session parameter returned from the Call Center Studio login service. It is not a fixed value. Each time the customer representative (agent) triggers the login process, a new session value is generated.

When the user first logs in to CRM, if your customer uses the Call Center Studio application, you have to log in automatically to the Call Center Studio application using the login service.

You can do this either front side or server side, according to your preference. The fictional flow needs to be determined specifically for your CRM.

**Note:** If you have a single customer here or if you use a custom CRM, the tenant -that we refer to as a customer- is the tenant address that you have in Call Center Studio

Access our Login & Authentication Page here.



**Method:** Get
**URL Endpoint:** https://[tenantname].callcenterstudio.com/api/
**Query String:**
1. function = login
2. email = [agent’s registered email address to Call Center Studio.]
3. app_token = [apptoken should be created via the application.]

https://TENANTNAME./api/?function=login&email=[Your Email Here]&app_token=[Your App_Token Here]

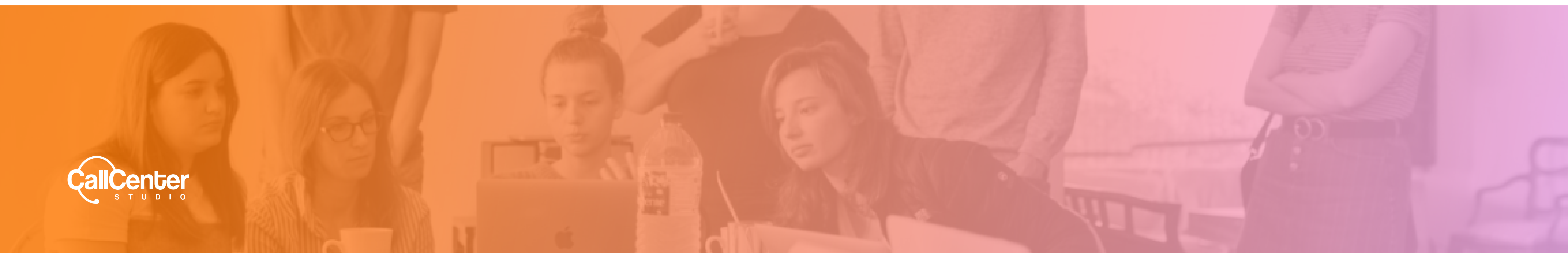
**Response:**

| | |
|---|---|
| Successful Login | {"login": true, "session": "[Your App Token Here]", "message": "Login Successful!", "language": "en"} |
| Incorrect User | {"login": false, "message": "User not found"} |
| Incorrect Token | {"login": false, "message": "Invalid Application Token"} |

As it is evident above, if the login parameter returns as true within the response message, then the authorization process is completed successfully.

The value of the mentioned session parameter should be stored in Heap memory (browser cookie) by Call Center Studio for 8-12 hours. This session parameter should be used when performing automatic softphone login via iframe and using other Public API. You can simply renew the session and repeat the request, if the login parameter returns false in the response,  You can continue to store your new session in heap memory.

We do not recommend you to create sessions continuously for every request. For security reasons, a complex structure was established in the backend. Old sessions will not work above a certain limit, and situations that you may not want to encounter such as getting stuck in calls, having problems while assigning requests to api may occur. When a session is expired, you can safely use the system by generating a new session.



## Generic Softphone SSO

Call Center Studio Generic Softphone application (referred to as iframe in conversations) is a webRTC-based single page application designed for the softphone component used by agents on the Call Center Studio agent screen to work embedded within other applications.

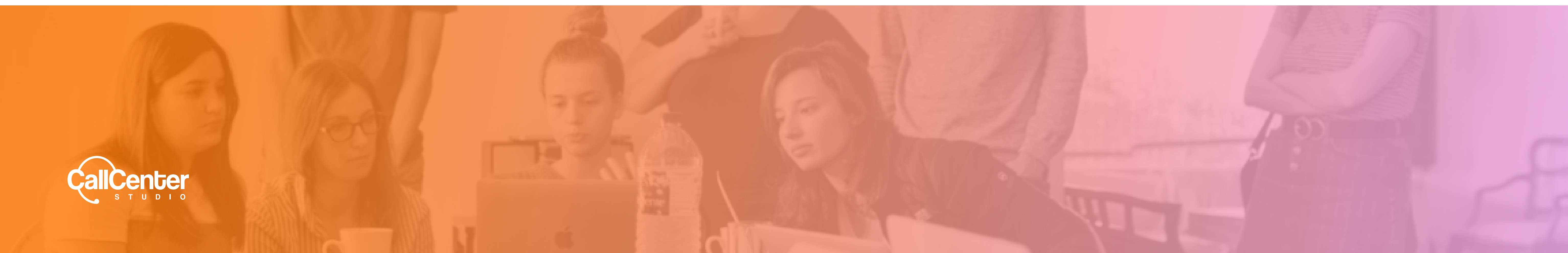
In our application where the transfer is made with the webRTC, the socket connection between the client and the server can leave the active call plugged in the system because the websocket is not reconnected when events such as closing occur, that call is open until the customer hangs up or the supervisor intervenes.

This is a natural process for applications that communicate via webRTC. This is a critical situation in call-based systems. For this reason, our generic softphone application can work perfectly within the applications that are built with single page application architecture.

**Method:** Get
**URL Endpoint:** https://softphone.alo-tech.com/generic/sso/
**Query String:**
1. tenant = [name of the owned tenant.]
2. session = [session information bouncing back from the login api.]

**Configuration**:
Chrome notifications and microphone must be allowed.
Required allowances must be given in order for CRM to run iframe.

```
<script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>

<iframe id="alotechiframe" scrolling="yes" allow="camera *; geolocation *; microphone *"
src="https://softphone.alo-tech.com/generic/sso/?tenant=TENANTNAME&session=SESSION"
scrolling="no"></iframe>

<script>

        function onmessage(e) {

           var event = JSON.parse(e.data);

           console.log(event);

        }

        window.addEventListener('message', onmessage, false);

</script>
```



## Pop-Up of the Contact Card for Inbound Calls

When the **messageType** parameter sent by the listened event is **IncomingCall**, it can be displayed by taking the phone number from the json pattern and querying the customer information with the phone number on the CRM.

If the phone number is masked for Tenant, the **activecallkey** parameter in the incoming json parameters can be sent to the Public API.

```
<script>
   function onmessage(e) {
       var event = JSON.parse(e.data);
       console.log(event);
    }
    window.addEventListener('message', onmessage, false);
</script>
```

[Access our Active Call documention page here.]

## Click2Call

Used to make calls with a single click, by clicking onto the phone number on CRM.
You can access the Click2Call document here

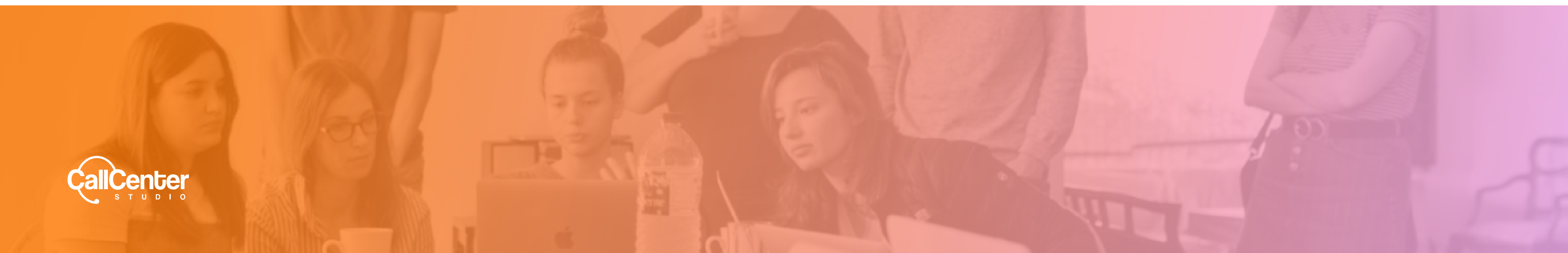
**Method**:Get
**URL Endpoint:**https://[tenantname].callcenterstudio.com/api/

**Query Strings:**
1. function = click2call
2. session = [session information bouncing back from the login api.]
3. hangup_url = [used to notify the defined service at the end of the call.]
4.transactionid = [used to define a unique ID by CRM, whilst initiating a call.]

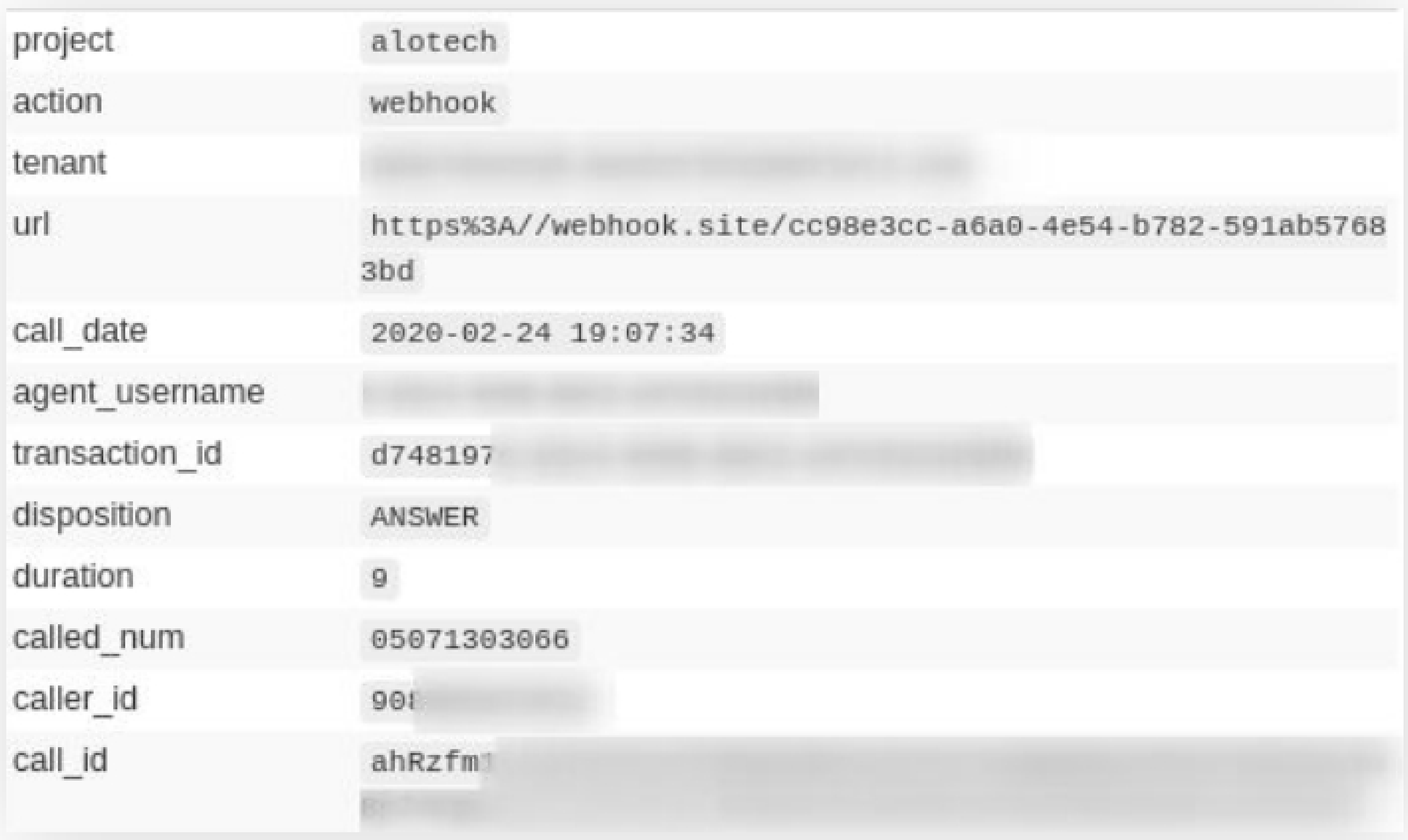
https://TENANTNAME/api/?function=click2call&session=[session]&transactionid=[crmuniqeueid]&hangup_url=[service]

**Response:**

| | |
|---|---|
| Success Message from the API | {"success": true} |



**Post Body That Goes to Hangup_url (Triggered at the End of a Call)**



| | |
|---|---|
| project | `alotech` |
| action | `webhook` |
| tenant | [redacted] |
| url | `https%3A//webhook.site/cc98e3cc-a6a0-4e54-b782-591ab57683bd` |
| call_date | `2020-02-24 19:07:34` |
| agent_username | [redacted] |
| transaction_id | `d748197`[redacted] |
| disposition | `ANSWER` |
| duration | `9` |
| called_num | `05071303066` |
| caller_id | `90`[redacted] |
| call_id | `ahRzfm`[redacted] |

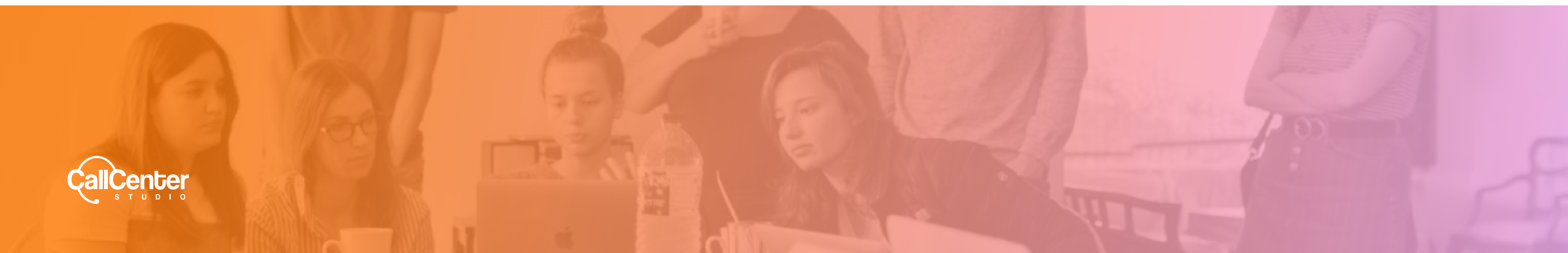
Here, all the details regarding the call can be accessed by requesting the **activecallkey** parameter to the CDR logs API with the **call_id** parameter.

[Access our Click2Call documention page here.]()

## ReportsCDRLogs

Used to access details regarding the calls.

**Method:**Get
**URL Endpoint:**https://[tenantadı].alo-tech.com/api/
**Query String:**
1. function = reportsCDRLogs
2. Search by date
   a) startdate = [initiation date of calls.]
   b) finishdate = [termination date of calls.]
3. Search of a Single Call
   a) activecallkey = [Unique ID of the call]
4. If 100 records return
   a) cursor = [cursor value that differs with each request.]
5. app_token = [app token should be created via the application.]

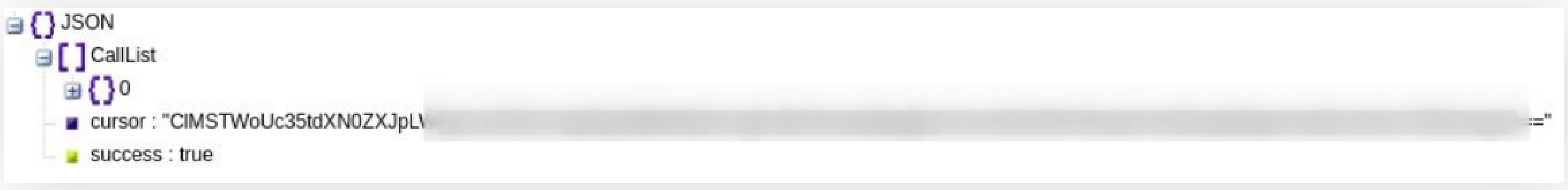


**Response:**

*Successful Answer Coming from API:*

The cursor value returns on every request. If the record is found, it becomes a list of calls in the **CallList** sequence. If the request is thrown with the active call key, there will be 1 record if there is a match and the array will be empty if there is no match.

If the start and end date requests are sent, a maximum of 100 records will be returned, if there is no match, the array will be empty.



```
{} JSON
  [] CallList
    {} 0
  cursor : "CIMSTWoUc35tdXN0ZXJpLV[redacted]="
  success : true
```

When you make a request with defining a start date and a finish date; since there is a **cursor** operation in the database layer, you can access a maximum of 100 data at a time.

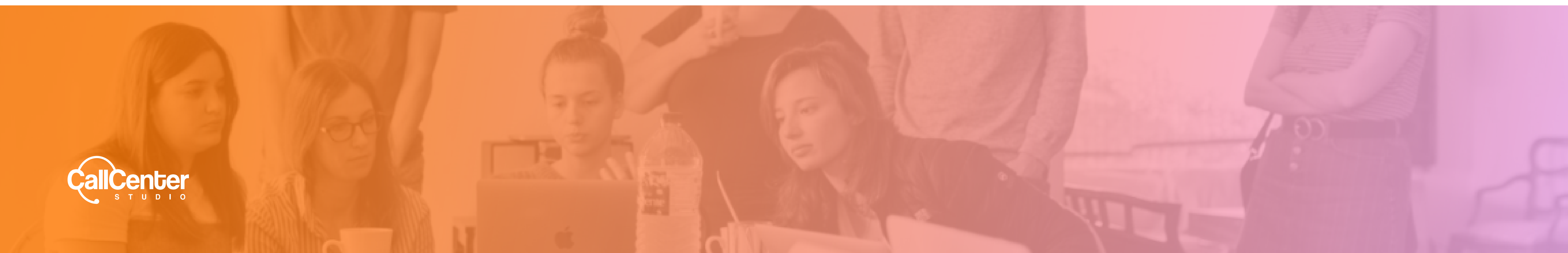
The API will return the cursor data to you every time. If 100 call records return when you make a request, you can reach your records between certain dates by repeating the request, triggering the API until the call returns under 100 records in the response.

If you run the request by adding the cursor parameter in qs each time until **CallList.length <100**, you can reach all your data.

https://[TENANNAME].callcenterstudio.com/api/?function=reportsCDRLogs&startdate=2019-12-18%2001:00:00&finishdate=2019-12-18%2009:00:00&app_token=[APPTOKEN]&cursor=[responsecursor]

https://[TENANTNAME].callcenterstudio.com/api/?function=reportsCDRLogs&activecallkey=[activecallkey]&app_token=[APPTOKEN]

[Access our Report CDR Logs documention page here.]()



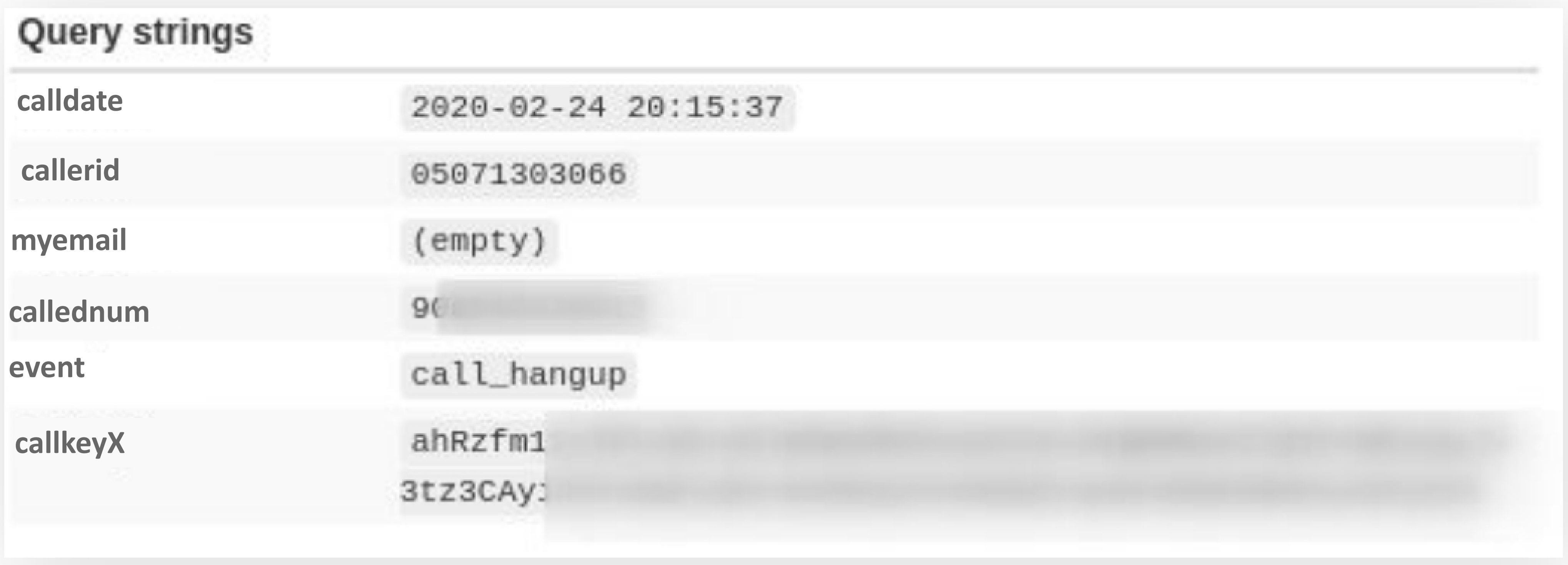
## Call and Incoming Call Notifications

While designing an IVR, the call notification step is defined into it.

When the call is terminated, the request is forwarded to the specified service as a querystring. (Transmitted as GET)

If the transmitted information is not sufficient, the variables related to the call can be reached by sending a request to the CDR API with the **activecallkey** information.

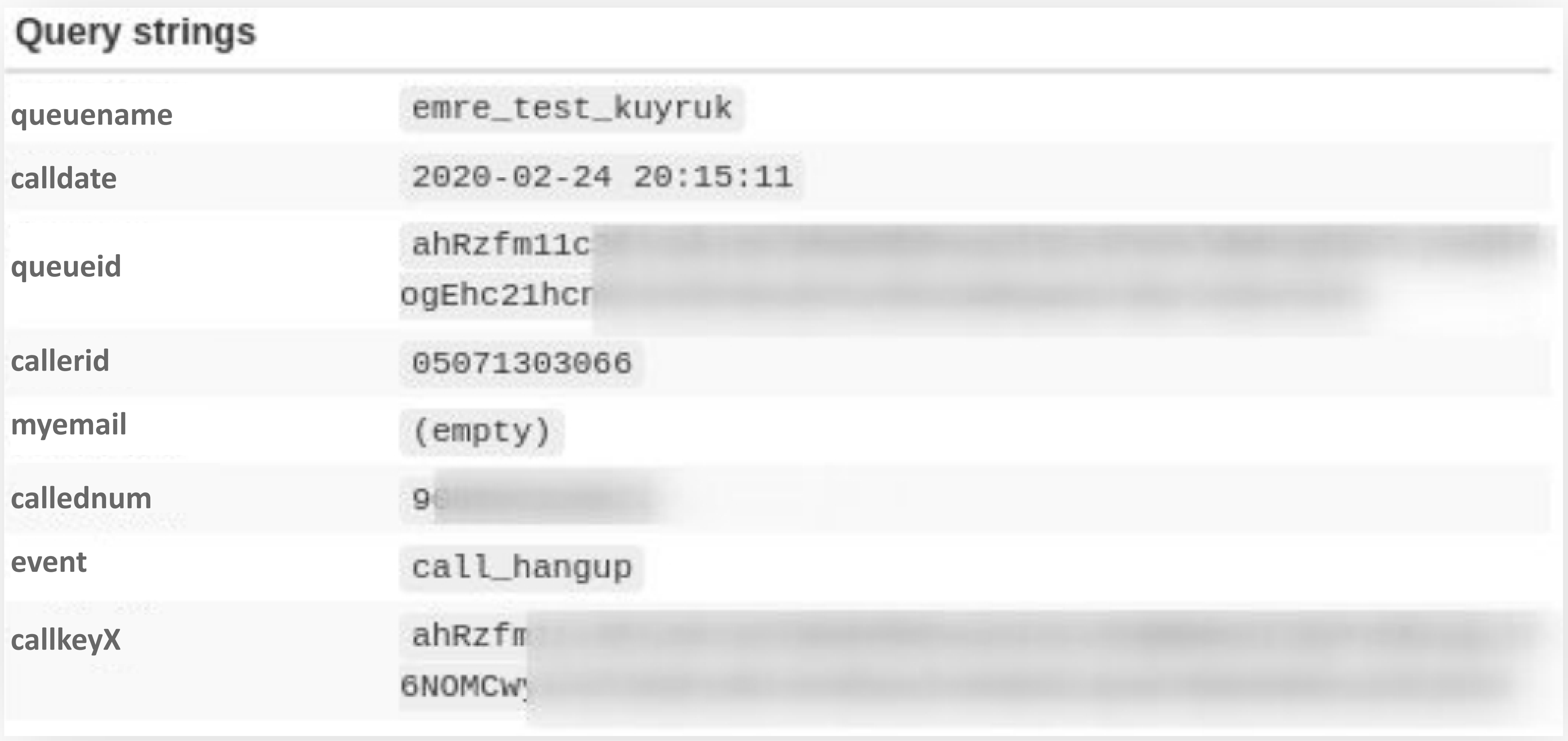
**Notification Examples:**



**Query strings**

| | |
|---|---|
| calldate | 2020-02-24 20:15:37 |
| callerid | 05071303066 |
| myemail | (empty) |
| callednum | 90[redacted] |
| event | call_hangup |
| callkeyX | ahRzfm1[redacted] 3tz3CAy[redacted] |

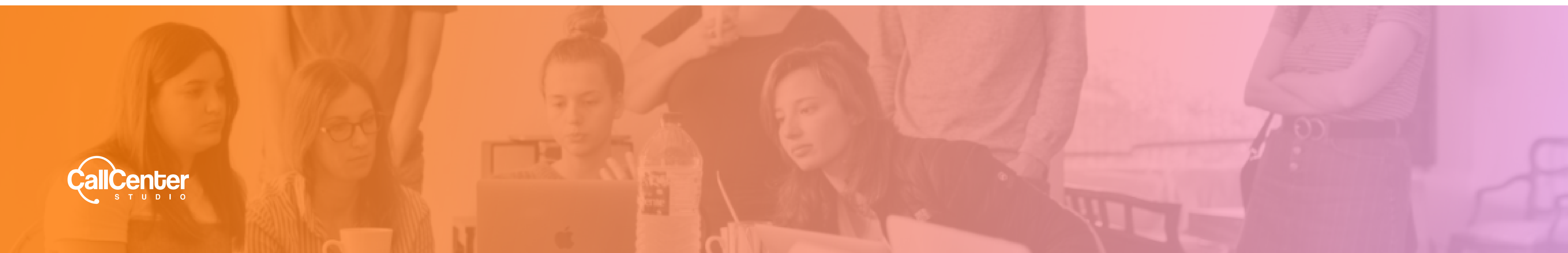
Call Ended without Connecting to a Queue

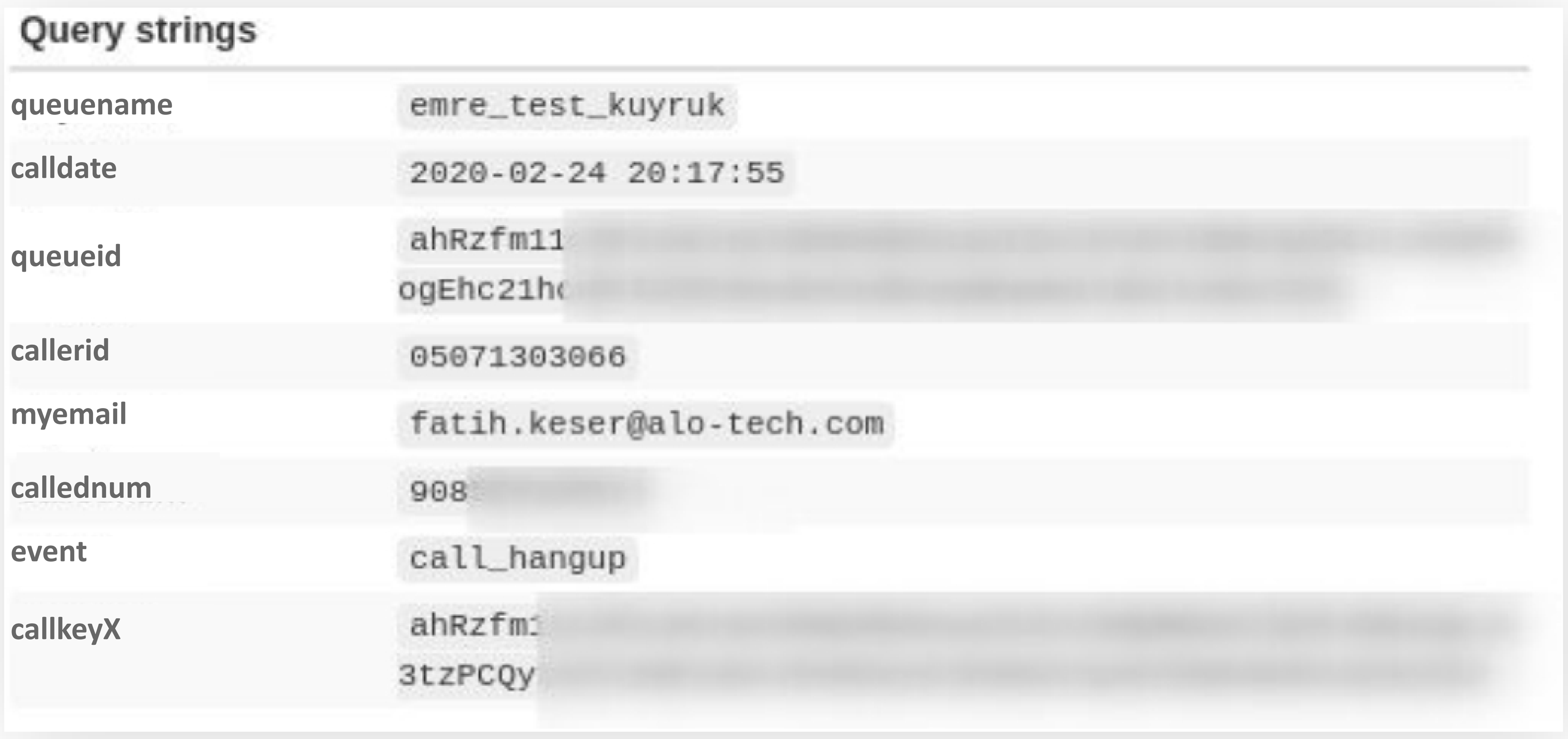


**Query strings**

| | |
|---|---|
| queuename | emre_test_kuyruk |
| calldate | 2020-02-24 20:15:11 |
| queueid | ahRzfm11c[redacted] ogEhc21hcr[redacted] |
| callerid | 05071303066 |
| myemail | (empty) |
| callednum | 90[redacted] |
| event | call_hangup |
| callkeyX | ahRzfm[redacted] 6NOMCw[redacted] |

Call Ended  in a Queue without Connecting to an Agent





**Query strings**

| | |
|---|---|
| queuename | emre_test_kuyruk |
| calldate | 2020-02-24 20:17:55 |
| queueid | ahRzfm11[redacted] ogEhc21h[redacted] |
| callerid | 05071303066 |
| myemail | fatih.keser@alo-tech.com |
| callednum | 908[redacted] |
| event | call_hangup |
| callkeyX | ahRzfm[redacted] 3tzPCQy[redacted] |

Call Ended After Speaking to an Agent

**Note:** Parameter names of the relative parameters, that are conducted via this module, are dynamic.

Parameter names can be updated or changed upon request.

## Happy customers

ebay FedEx L'ORÉAL PARIS Deloitte. pepsi xerox TOSHIBA

CAT KASPERSKY GARNIER AIR ALBANIA Brightstar VICHY Fakir

THE BODY SHOP DECATHLON Domino's Europcar VOLVO KPMG Vaillant