sales_Q

API-Documentation

updated: 11.04.2019

Inhaltsverzeichnis

1. How to Use

In the following, some explanations and representations of code snippets are intended to make it easier for you to use our interface. If you have any questions, please do not hesitate to contact us. We can also implement a campaign script for you.

In the following, the various functions that are available to you within the framework of our interface will first be explained. Finally, a description of the export is given. This contains various formats in which you can get back the qualified lead information.

2. Funktionen

The functions available to you are described below. The code snippets inserted show an example of the implementation. The code can be copied and used by you at any time.

2.1. Import

This function is used to send leads to the Sales_Q system and start a chat.

URL	https://server.sales-q.com:8000/import
Method	POST
Content-Type	application/json

Mandatory Parameter:

Parameter	Type	Description
token	String	Contains your Sales_Q API token. It is used to uniquely identify the campaign.
leads	Object [ ]	Contains at least one lead.
leads[x].Phone	String	Contains the mobile phone number of the lead.

The array ***leads*** consists of one or more leads. A lead can have any number of parameters (the ***phone*** parameter is mandatory). The data points to be queried are automatically added from the sent (custom) or the campaign script and filled during the process.

The parameters must be parser-compliant and must not contain dots or other special characters.

By default, the campaign script created at the start of the project is used. If you want a lead to have a custom *script*, just send it as a parameter, as shown in the example below. The other parameters of the lead can then be accessed in the *script*.

An element in the optional custom *script* always consists of:

Parameter	Type	Example
content	String	The message for the element is transferred here.
state	String	Is unique (starts with state: 1 and this is mandatory) and represents the id of the script element.
next	Number, Object [ ]	A reference to the next building block element.
dataconnect	String	The name of the variable in which a response from the user can be stored.

There are 3 types of messages:

Type	next-Type	dataconnect	Description
Simple Message	Number	empty	The next message is triggered immediately. This is very handy to make the dialogue clearer for the reader.
Free text question	Number	Contains a variable	A free text question waits for an answer from the end user and writes it to dataconnect.
Multiple-Choice	Object [ ]	Contains a variable	A multiple-choice message waits for a predefined response and writes it to *dataconnect*.

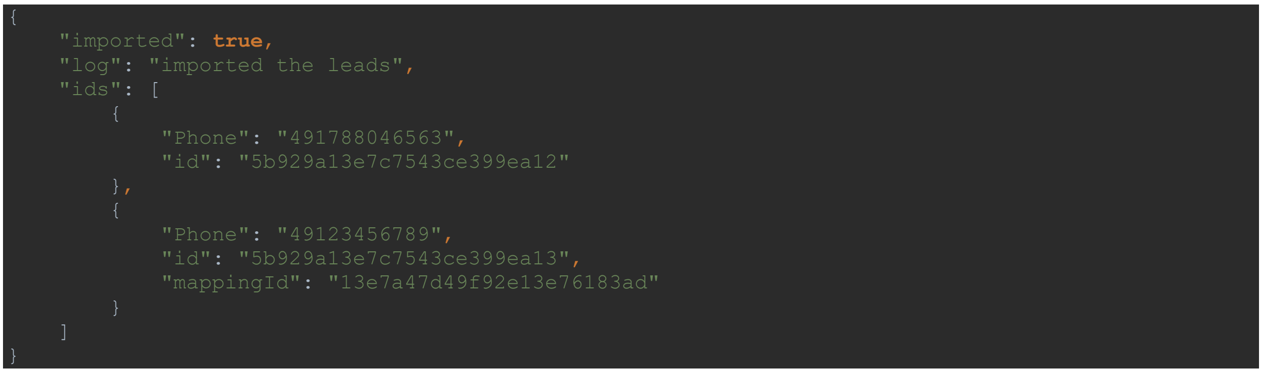
Multiple-Choice: If the message is a multiple-choice question, then the next parameter consists of an array.

For an object from next in a multiple-choice question:

Parameter	Type	Description
answer	String	*Answer* is the answer that the end user can enter. Our systems automatically recognize variations and synonyms for this term.
next	String	*Next* describes which script module is to be sent next.
value	String, Boolen	Is optional and contains the value to be written to *dataconnect*. If no value is given, the answer is written to *dataconnect*.

Expected response:

Parameter	Type	Description
response	Boolean	Specifies whether the request was successful.
log	String	Contains more information about the response.
ids	Object [ ]	Contains the Sales_Q-*ID* and the *Phone* of the imported leads. The variable *mappingId* can also be contained if it was specified in the lead during the import. *ids* is only sent if response contains true.



```
{
    "imported": true,
    "log": "imported the leads",
    "ids": [
        {
            "Phone": "491788046563",
            "id": "5b929a13e7c7543ce399ea12"
        },
        {
            "Phone": "49123456789",
            "id": "5b929a13e7c7543ce399ea13",
            "mappingId": "13e7a47d49f92e13e76183ad"
        }
    ]
}
```

Example import with all cases:

```
var request = require("request");
var options = {
    method: 'POST',
    url: 'https://server.sales-q.com:8000/import',
    body:
        {
            "token": "qf8ei4yHu2pzcaneFhdA...",
            "leads": [
                {
                    "Phone": "491788046563",
                    "Anrede": "Herr",
                    "Vorname": "Peter",
                    "Nachname": "Müller"
                },
                {
                    "Phone": "49123456789",
                    "Anrede": "Herr",
                    "Vorname": "Johannes",
                    "Nachname": "Muster",
                    "Alter": "23",
                    "mappingId": "13e7a47d49f92e13e76183ad",
                    "script": [
                        {
                            "content": "{Anrede} {Nachname} ich kontaktiere Sie wegen... usw.",
                            "state": "1",
                            "next": "2"
                        },
                        {
                            "content": "Wann sollen wir Sie kontaktieren \n " +
                                "A) Morgens \n " +
                                "B) Mittags \n " +
                                "C) Abends",
                            "state": "2",
                            "next": [
                                {"answer": "A", "next": "3", "value": "Morgens"},
                                {"answer": "B", "next": "3", "value": "Mittags"},
                                {"answer": "C", "next": "4", "value": "Abends"}
                            ],
                            "dataconnect": "Wann kontaktieren"
                        },
                        {
                            "content": "Wann passt es Ihnen am besten",
                            "state": "3",
                            "next": "5",
                            "dataconnect": "Uhrzeit"
                        },
                        {
                            "content": "Das liegt leider außerhalb unserer Geschäftszeiten.",
                            "state": "4",
                            "next": "4"
                        },
                        {
                            "content": "Danke wir kontaktieren Sie dann im Laufe" +
                                " der Woche {Wann kontaktieren} um {Uhrzeit}",
                            "state": "5",
                            "next": "5"
                        }

                    ]
                }
            ]
        },
    json: true
};
request(options, function (error, response, body) {
    if (error) throw new Error(error);
    res.send(response);

});
```

2.2. Remove

If Sales_Q should terminate or interrupt the qualification process for a specific lead, simply remove the lead from our system.

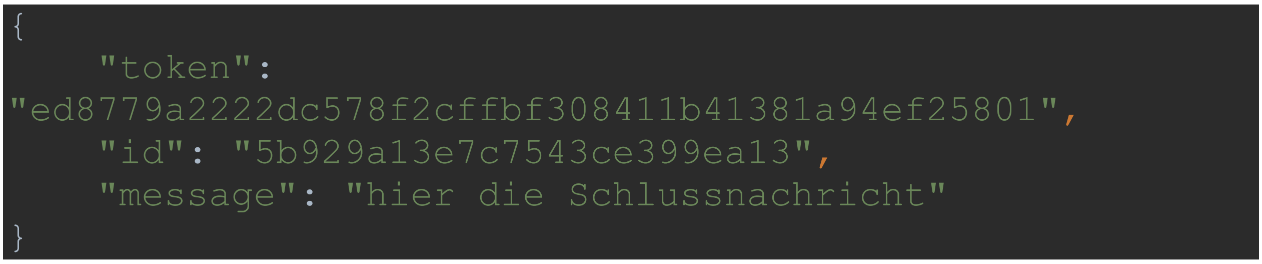
URL	https://server.sales-q.com:8000/remove
Method	POST
Content-Type	application/json

Mandatory Parameters:

Parameter	Type	Description
token	String	Contains your Sales_Q API token. It is used to uniquely identify the campaign.
id	String	Contains the ID of a lead that was returned during import.

Optional Parameters:

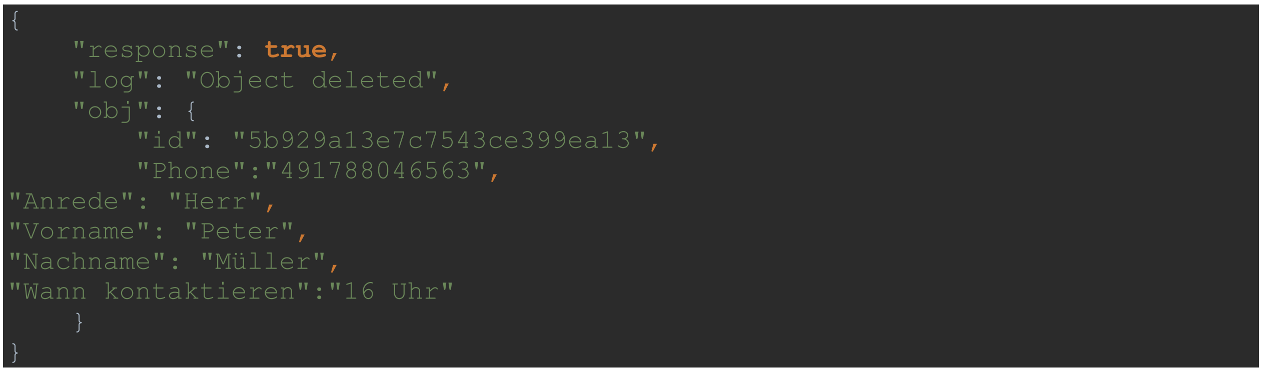
Parameter	Type	Description
message	String	Allows the lead to send a message before removing it.



```
{
    "token":
"ed8779a2222dc578f2cffbf308411b41381a94ef25801",
    "id": "5b929a13e7c7543ce399ea13",
    "message": "hier die Schlussnachricht"
}
```

Expected response:

Parameter	Type	Description
response	Boolean	Specifies whether the request was successful.
log	String	Contains more information about the response.
obj	Object [ ]	Contains the remote lead with all previously qualified data points. obj does not exist if response is false.



```
{
    "response": true,
    "log": "Object deleted",
    "obj": {
        "id": "5b929a13e7c7543ce399ea13",
        "Phone":"491788046563",
"Anrede": "Herr",
"Vorname": "Peter",
"Nachname": "Müller",
"Wann kontaktieren":"16 Uhr"
    }
}
```

2.3. Update

To get the current status of active leads.

URL	https://server.sales-q.com:8000/update
Method	GET
Content-Type	application/json

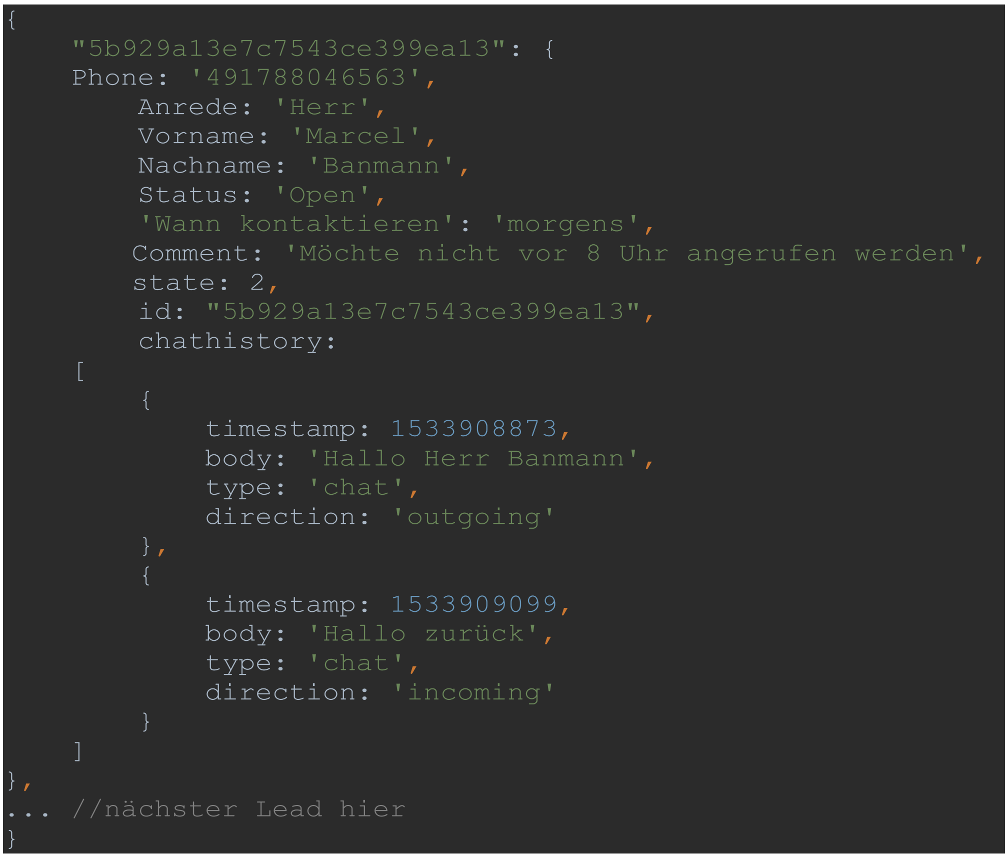
Mandatory Header:

Header	Type	Description
token	String	Contains your Sales_Q API token. It is used to uniquely identify the campaign.

Expected response:

Parameter	Type	Description
[leadid]	Object [ ]	Includes one of the leads with the data points.
[leadid].Status	String	An element of `Not contacted`, `Not on WhatsApp`, `Open`, `Engaged`, `Dead`, `Done` and `Archive` describes the status of the lead.
[leadid].state	Number	Contains the state in which the lead is currently located in the script.
[leadid].Comment	String	Sometimes contains an important comment from an agent.

A JSON object follows as ***response***, which contains the individual leads with their chat histories, whereby the IDs of the leads serve as keys.



```
{
    "5b929a13e7c7543ce399ea13": {
    Phone: '491788046563',
        Anrede: 'Herr',
        Vorname: 'Marcel',
        Nachname: 'Banmann',
        Status: 'Open',
        'Wann kontaktieren': 'morgens',
        Comment: 'Möchte nicht vor 8 Uhr angerufen werden',
        state: 2,
        id: "5b929a13e7c7543ce399ea13",
        chathistory:
    [
        {
            timestamp: 1533908873,
            body: 'Hallo Herr Banmann',
            type: 'chat',
            direction: 'outgoing'
        },
        {
            timestamp: 1533909099,
            body: 'Hallo zurück',
            type: 'chat',
            direction: 'incoming'
        }
    ]
},
... //nächster Lead hier
}
```

3. Export

If a lead has been qualified, it will be played back to a webhook.
For the export we set a webhook at the beginning of the campaign. POST-Requsts with the qualified leads will be sent to this webhook.

URL	Sales_Q will call your URL
Method	POST
Content-Type	application/json

Request Parameter:

Parameter	Type	Description
token	String	Contains your Sales_Q API token. It is used to uniquely identify the campaign.
lead	Object	The qualified lead with the data points is located in lead.
chathistory	Object [ ]	The entire chat history will be sent to you and will be available to you afterwards.
chathistory[x].timestamp	Number	Contains the time at which the message was sent/received. This is formatted as a Unix timestamp and can therefore be converted to all formats.
chathistory[x].type	String	Distinguishes between chat, pictures or other files.
chathistory[x].direction	String	Specifies whether the message is incoming or outgoing.
chathistory[x].body	String	Contains the message.

These additional data points are included:
Status: An element of `Not contacted`, `Not on WhatsApp`, `Open`, `Engaged`, `Dead`, `Done`, `Callback` and `Archive` describes the status of the lead.

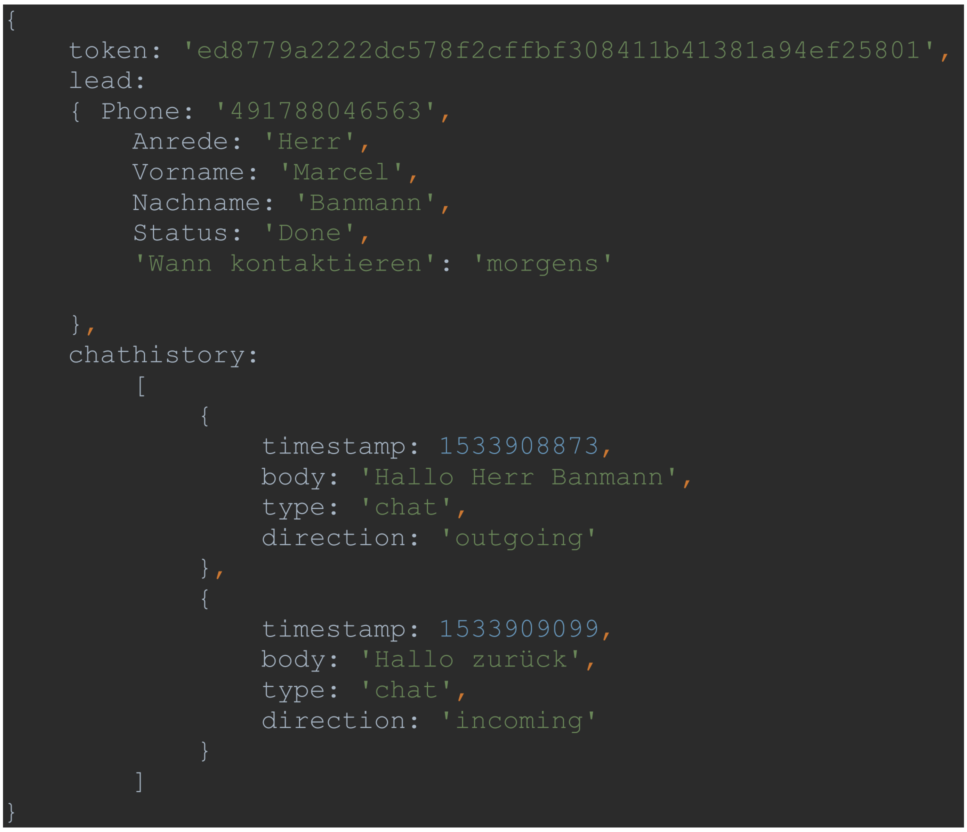
Name	Beschreibung
Not contacted	Lead has not yet been contacted.
Not on WhatsApp	Lead has no WhatsApp Account.
Open	The first message was delivered.
Engaged	There was a response message from the lead.
Dead	Lead has no more interest.
Done	Qualification process completed.
Callback	Lead wants to be called. Date in Comment.
Archive	Lead sent important information after qualification or deletion.

Comment: Sometimes contains an important comment from one of our agents.

state: Contains the state in which the lead is currently located in the script.

id: Contains the id of the lead.

Example:



```
{
    token: 'ed8779a2222dc578f2cffbf308411b41381a94ef25801',
    lead:
    { Phone: '491788046563',
        Anrede: 'Herr',
        Vorname: 'Marcel',
        Nachname: 'Banmann',
        Status: 'Done',
        'Wann kontaktieren': 'morgens'

    },
    chathistory:
        [
            {
                timestamp: 1533908873,
                body: 'Hallo Herr Banmann',
                type: 'chat',
                direction: 'outgoing'
            },
            {
                timestamp: 1533909099,
                body: 'Hallo zurück',
                type: 'chat',
                direction: 'incoming'
            }
        ]
}
```

If you have any further questions, simply contact us:
Marcel Banmann
marcel@sales-q.com
+49178 804 656 3