

# Ragic Builder API

---

## Ragic Builder API

---

---

# Table of Contents

- 1. How to use an Ragic Builder HTTP API ..... 1
  - 1.1. What is an HTTP API? ..... 1
  - 1.2. Using cURL to access HTTP API ..... 1
  - 1.3. Using API with other programming languages ..... 2
- 2. Working with Ragic Builder API ..... 3
  - 2.1. Authentication ..... 3
  - 2.2. Reading ..... 3
  - 2.3. Writing ..... 7

---

# Chapter 1. How to use an Ragic Builder HTTP API

## 1.1. What is an HTTP API?

Ragic Builder HTTP API is basically a RESTful API to read or write data to a Ragic Builder DB with programs. This way, you can do things like:

- Integrate your existing systems with functions that you built on Ragic.
- Create fully customized forms
- Create fully customized reports

An HTTP API endpoint is just like an URL that you would access with your browser. Only that it's designed for programs to access, and the return data is formatted for programs to read.

There are two types of HTTP request methods used in Ragic Builder API: GET and POST. GET is for retrieving data, POST is for writing data to the Ragic Builder DB.

## 1.2. Using cURL to access HTTP API

You can test most GET method APIs easily by entering API endpoint URLs in your browser. For example, you can try the following URL to access customer account information on the Ragic Builder demo:

```
http://api.ragic.com/demo/sales/1
```

But it's not as easy to create POST method requests on your browser, so we recommend a tool called cURL for you to create all types of HTTP requests you want to test our API. Our document will also be using cURL commands as samples API calls, but you can also use any tool that you're familiar with to create HTTP requests and parse responses.

You can download cURL for your platform at <http://curl.haxx.se/download.html> , and you can also read its full documentation at <http://curl.haxx.se/docs/manpage.html> [<http://curl.haxx.se/download.html>] . But don't worry, we'll tell you about the necessary usages of cURL as we explain the Ragic Builder HTTP API.

After you have downloaded cURL, you can use the following command to access the same endpoint we described above, and you should see the same output as you would in a browser.

```
curl http://api.ragic.com/demo/sales/1
```

## 1.3. Using API with other programming languages

HTTP based API is very common and is supported in almost all platforms. You should be able to look for ways to do this in the language of your choice, or use this [<http://rest.elkstein.org/2008/02/rest-examples-in-different-languages.html>] as a starting point.

Because the response is in JSON format, it should be a good idea to find a JSON parser in your language to speed up development. You can find lots of parsers in different languages at <http://json.org>.

---

# Chapter 2. Working with Ragic Builder API

## 2.1. Authentication

### Session ID Authentication

For programs to retrieve access right protected data on Ragic Builder, you need to authenticate yourself first as you would with a human user.

You send a request for a session id with a valid e-mail and password. You can issue a HTTP request using the `-d` argument containing the id and password. The `-c` parameter will store sessionId in the cookie jar file specified:

```
curl -d u=jeff@ragic.com' \  
      -d p=123456' \  
      -d login_type=sessionId' \  
      -c cookie.txt \  
      -k \  
      https://api.ragic.com/AUTH
```

If authentication failed, server will return -1. If authenticated, you will receive a session id in the response like this:

```
2z5u940y2tnkes4zm49t2d4
```

### OAuth 2.0 Authentication

Under construction.

## 2.2. Reading

### Finding API endpoints

You can find the API endpoint for your Ragic Builder form or entry on [api.ragic.com](https://api.ragic.com) under the same path.

```
https://api.ragic.com/<account>/<dbUrl>/<formNumber>[/<id>]?v=2
```

Please note that you're using the version 2 API by adding a "v" parameter `v=2`. It's a good practice to specify the API version every time you send your request to ensure a correct version. If the version is not specified, the latest version will be used, but unexpected changes to the API may cause problems in your application.

For example if you usually access your Ragic Builder form using the following URL:

```
https://www.ragic.com/demo/sales/1
```

Its HTTP API endpoint URL would be:

```
https://api.ragic.com/demo/sales/1?v=2
```

Or for a single entry in your form, you would include an id to specify an entry:

```
https://www.ragic.com/demo/sales/1/6
```

Its corresponding HTTP API endpoint URL is simply:

```
https://api.ragic.com/demo/sales/1/6?v=2
```

## Returned data JSON format

For example, this is the API response for `http://api.ragic.com/demo/sales/1?v=2` . Remember to specify your cookie jar file in the call like " -b cookie.txt"

```
{
  "17": {
    "Account Name": "Dunder Mifflin",
    "Account Owner": "Jim Halpert",
    "Phone": "1-267-922-5599", ...
  }
  "14": {
    "Account Name": "Ratshotel",
    "Account Owner": "Dwight Schrute",
    "Phone": "1-267-922-5529",
    ...
  }
}
```

The listing is defaulted to 1000 entries. You can add paging parameter "limit" according to the following sections to get more entries.

The data in the subtables are displayed as the sample below. The attribute name will start with "\_subtable\_" followed by the external id of the first field, to identify which subtable it is. The content of the subtable is presented the same way as the fields in a form.

```
"14": {
  "Account Name": "Ratshotel",
  "Account Owner": "Dwight Schrute",
}
```

```
"Phone": "1-267-922-5529",
...
"_subtable_800014": {
  "29": {
    "Contact Name": "Arden Jacobs",
    "Title": "Specialist",
    "E-mail": "Arden@ratshotel.com",
  },
  "30": {
    "Contact Name": "Kermit Moore2",
    "Title": "Manager",
    "E-mail": "Kermit@ratshotel.com",
  }
}
}
```

If your application does not need the data in the subtables, you can add the parameter `subtables=0` to turn off the fetching of subtable data.

The comments of the entries is retrieved as subtables also. It will be retrieved in a subtable with the external id of 61.

## Filter Conditions

Very often your database contain large amount of entries, so it's better to apply filters when you retrieve data. Ragic Builder API filters are in a special format

You can use the parameter "where" to add a filter condition to a query as below:

```
curl -d 'where=800003,eq,Dwight Schrute'
-b cookie.txt
http://api.ragic.com/demo/sales/1
```

The parameter is a "," comma delimited format, with at least 3 arguments.

- 1. External id of the field that you would like to filter.
- 2. Operand in the form of integer to specify your filter operation. The list of operands are listed below.
- 3. The value that you would like to filter the field with. Remember if your value might include a "," comma character, please use its URL encoded value "%2C" to avoid collision.

You can supply a query with multiple filter conditions as below:

```
curl -d 'where=800003,eq,Dwight Schrute'
-d 'where=800007,eq,Reseller'
-b cookie.txt
http://api.ragic.com/demo/sales/1
```

Here's the list of operands that you can use, because the equal sign needs to be escaped in an URL string, we provide another representation of the operand without using "=":

- Equals: =,eq
- Greater or equals: >=,gte
- Less or equals: <=,lte
- Greater: >,gt
- Less: <,lt
- Like: like
- Equals selection: ==,epeq

## Limiting Entry Number / Paging

Very often you do not want to fetch all entries with one request, you can use the "limit" parameter to specify how many entries that you would like to retrieve, and how many entries that you would like to skip at the start, so clients can implement pages for viewing entries.

The usage of limit parameter is similiar to SQL limit parameter. There are two arguments to the limit parameter, separated by comma. The first one is how many entries that you would like to skip in the beginning, and the second one is how many entries that you would like to be returned.

Returned data is defaulted to 1000 entries, you will need to provide limit parameters if your response has more than 1000 entries.

```
curl -d 'limit=5,8' -b cookie.txt http://api.ragic.com/demo/sales/1
```

## Ordering

You can specify how the entries are ordered by adding the "order" parameter. It's also kind of similar to the ORDER BY clause in SQL, its value is a comma separated string with two arguments. The first one is the external id of the domain that you would like the entries to be sorted according to, and the second one is the order: either ASC for ascending order, or DESC for descending order.

```
curl -d 'order=800236,DESC' -b cookie.txt http://api.ragic.com/demo/sales/1
```

## Callback Function

Adding a callback function parameter will enclose the returned JSON object as a argument in a call to the callback function specified by you. This is especially useful when you're accessing our API using Javascript, you may need this to do cross domain ajax calls.

For example, adding `callback=testFunc` as below:

```
curl -d callback=testFunc -b cookie.txt http://api.ragic.com/demo/sales/1
```

Will enclose the returned JSON object in a function call so you can process the returned data in your callback function:

```
testFunc({"17": {  
  "Account Name": "Dunder Mifflin",  
  "Account Owner": "Jim Halpert",  
  "Phone": "1-267-922-5599",  
  ...  
  ...  
});
```

## Field Naming

The JSON data format uses the field name string as the attribute name, you can use a "naming" parameter to specify that you would like to use external id as the attribute name to identify a field.

Possible values include: EID (External Id), FNAME (Field Name). This type of attribute will only need to be set once, and the configuration will be applied to all subsequent requests in this session.

```
curl -d naming=EID -b cookie.txt http://api.ragic.com/demo/sales/1
```

## 2.3. Writing

### Creating a New Entry

The endpoints for writing to a form is the same as reading them, but you issue a POST request instead of GET request.

What you need to do is use the external ids of the fields as name, and the values that you want to insert as parameter values.

Please note that your user will need write access to the form for this to work.

```
curl -F '800001=Dunder Mifflin' \  
-F '800002=1-267-922-5599' \  
-F '800003=Jim Halpert' \  
-F '800007=Customer' \  

```

```
-b cookie.txt \  
https://api.ragic.com/demo/sales/1
```

If the field is a multiple selection that can contain multiple values, you can have multiple parameters with the same external id as names, like below:

```
curl -F '800001=Dunder Mifflin' \  
-F '800002=1-267-922-5599' \  
-F '800003=Jim Halpert' \  
-F '800007=Customer' \  
-F '800007=Reseller' \  
-b cookie.txt \  
https://api.ragic.com/demo/sales/1
```

## Modifying an Entry

The endpoint for modifying an entry is also the same as reading an existing entry.

```
https://api.ragic.com/<account>/<dbUrl>/<formNumber>[/<id>]
```

All you need to provide is the external ids of the fields that you would like to modify to:

```
curl -F '800003=Dwight Schrute' \  
-F '800007=Partner' \  
-b cookie.txt \  
https://api.ragic.com/demo/sales/1/3
```

## Status Return

You will receive an returned JSON object containing the status of your request, it should contain the word "SUCCESS" if successfully posted. If there's a problem with your request, the attribute "msg" should give you some clue what's going on.

The id, or "ragicId" of your entry will be returned for your reference. Also the value of your first domain returned, because often the field value is automatically generated.

```
{  
  "status": "SUCCESS",  
  "msg": "&nbsp;",  
  "ragicId": 3,  
  "rv": "Dunder Mifflin"
```

}