

Request Root Key and Response Root Key

Last Modified on 08/17/2020 2:01 am EDT

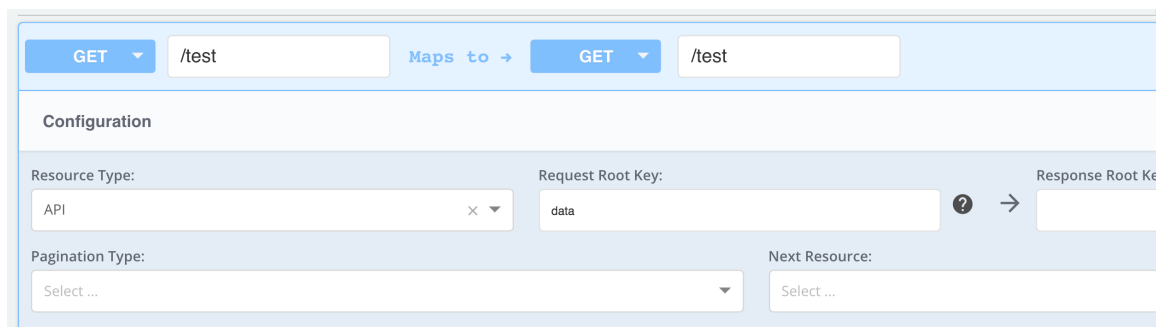
When you are building your first connector using Connector Builder, you may wonder what exactly the Request Root Key and the Response Root Key are intended to deliver. This article provides you with a quick introduction to this concept and allows you to understand a little better how to make use of these two keys and properly approach this feature.

Let's start from the very beginning, let's use the payload below as an illustration:

```
1 {
2   "data": {
3     "contacts": [{
4       "firstName": "Steve",
5       "lastName": "Jones",
6       "Age": 33,
7       "emailAddress": "example@example.com"
8     }, {
9       "firstName": "Jessica",
10      "lastName": "Muller",
11      "Age": 29,
12      "emailAddress": "example2@example.com"
13    }]
14  }
15 }
```

As you can see, this payload contains a parent property called "data" and a child called "contacts"; now, let's say that our aim was to only return "contacts". In that case, you would have to essentially go through the parent "data" to be able to tap into the "contacts" child property (i.e. the dot notation => data.contacts). To avoid this extra effort, we have essentially implemented the Request and Response Root Keys to allow you to access the child property directly without having to handle additional logic on your side.

To understand how it works, let's use our payload above as an example again. Let's say you want to return the contact object as the payload; all that you would have to set up in your request is "data" as your Request Root Key as shown here:



The screenshot shows the configuration interface for a connector. At the top, there are two GET requests to /test, with a 'Maps to' arrow between them. Below this is a 'Configuration' section. The 'Resource Type' is set to 'API'. The 'Request Root Key' is set to 'data'. The 'Response Root Key' is currently empty, with a question mark icon and a right-pointing arrow next to it. The 'Pagination Type' is set to 'Select ...' and the 'Next Resource' is also set to 'Select ...'.

and our system automatically interprets that request to return only the child/children of the "data" parent property. The following is a GIF to help conceptualize the approach:

```
var payload = {
  "data": {
    "contacts": [{
      "firstName": "Steve",
      "lastName": "Jones",
      "Age": 33,
      "emailAddress": "example@example.com"
    }, {
      "firstName": "Jessica",
      "lastName": "Muller",
      "Age": 29,
      "emailAddress": "example2@example.com"
    }]
  }
}
```

```
console.log(payload.data)
```

```
Native Browser JavaScript
>
{ contacts:
  [ { firstName: 'Steve',
      lastName: 'Jones',
      Age: 33,
      emailAddress: 'example@example.com' },
    { firstName: 'Jessica',
      lastName: 'Muller',
      Age: 29,
      emailAddress: 'example2@example.com' } ] }
=> undefined
>
```