Industries REST API

Version 49.0, Summer '20

@salesforcedocs

Last updated: June 5, 2020

CONTENTS

CHAPTER 1 Salesforce Industries REST API

In this chapter ...

The Salesforce Industries REST API gives you access to individuals in your org. Use the POST, GET, and PUT operations to create, read, and update your records.

The API provides powerful and simple web services for interacting with Lightning Platform. It's customized for use with the Industries products, such as Salesforce Health Cloud and Salesforce Financial Services Cloud, and gives you easy access to the special objects and features these products provide.

To use the API requires basic familiarity with software development, web services, and the Salesforce user interface. For general information about the Salesforce REST API, see the *REST API Developer's Guide*. That guide explains the API's characteristics and architecture and how to set up your development environment. It also includes information to get you started using the API and tools, like cURL and Workbench REST Explorer.

Note: This guide is for use with the individual data model. It isn't intended for use with person accounts.

Getting Started with REST API for Industries

Run some sample REST requests in your development org to learn Industries REST API basics. Trying out the examples now makes it easier to build your applications later.

Start with a development platform with authorization set up on it. If you don't have a platform, see the *REST API Developer's Guide* for information about creating one.

Salesforce Industries REST API isn't part of the regular Lightning Platform REST API. It has its own versioning scheme and makes more use of namespaces.

Quick Start

Review this short demonstration of how the Industries REST API works.

A basic Industries REST API request has this format.

```
/services/apexrest/package namespace/current version/resource name/optional parameters
```

The *package namespace* value is omitted for orgs that don't use a namespace. If you don't know whether your org has a namespace, here's how to check.

For example, this GET request returns all individuals in the org.

```
/services/apexrest/org1_gs0/v1/individual/
```

Here's the response with three records returned, with most values omitted for readability.

```
{
  "statusCode" : 3,
  "responseBody" : {
    "nextRecordUrl" : null,
    "individuals" : {
      "001B000000AXnplIAD1454697446538" : {
        "org1_gs0__birthdate__c" : "None ,",
        [Values omitted]
        "isdeleted" : false
      },
      "001B000000AXncDIAT1454697283406" : {
        "org1_gs0__birthdate__c" : "None ,",
        [Values omitted]
        "isdeleted" : false
      },
      "001B000000AXnQ2IAL1454697097426" : {
        "org1_gs0__birthdate__c" : "None ,",
```

```
        [Values omitted]
        "isdeleted" : false
      }
    }
  },
  "message" : " Number of Individuals retrieved: 3"
}
```

You can retrieve up to 200 records at once. If there are more than 200 records, the `nextRecordUrl` value shows the limit used and the offset to the next batch of records. For example, if this request had returned 200 records and more records were available, the next record value would be `"nextRecordUrl" : "/v1/individual/?limit=200&offset=200"`.

Response Status Codes

Each response includes a status code indicating whether the request was successful or not.

Status Code	Description
1	Request completed successfully
2	`Create Individual` operation was successful
3	`Read Individual` operation was successful
4	`Update Individual` operation was successful
5	`Create Individual` operation failed
6	`Read Individual` operation failed
7	`Update Individual` operation failed
8	`Read Individual` operation is not yet complete

Namespaces

Check if your org has a namespace. If it does, include the namespace in every REST call you make. Only Developer Edition orgs can have namespaces. If you're not using a DE org, you don't have to use an org namespace in your REST requests.

1. Enter *`Package`* in the Quick Find box or select **Create** > **Packages** from Setup.
2. Look at the Namespace Prefix value on the Packages page. If the namespace is set, use that name in your REST calls.

Using the Industries REST API

The Industries REST API provides get, post, and put operations for working with Individual records, plus filtering.

Post Operation

Create individuals one at a time or in batches. Use the POST method to create individuals in your org. You can create more than one individual at a time, but it's an all-or-none operation. If an error occurs while creating any individual's record, no individuals are created.

Put Operation

Make updates to individuals in your org, one at a time or in groups.

Filter Results

Filter the fields returned by a GET request so that you see only the values you want.

Get Operation

Get information for one individual, some individuals, or all individuals in your org, including all individuals with changes since a specific date.

The Industries REST API retrieves information from your org about:

- A specific individual
- All individuals
- Individuals with changes since a certain date

Retrieve One Individual

This request retrieves a specific individual, identified by the individual's ID value.

```
/services/apexrest/v1/individual/individual_id
```

The *individual_id* value is a string that uniquely identifies the individual. The response body follows this format:

```
{
  "statusCode" : 3,
  "responseBody" : {
    "nextRecordUrl" : null,
    "individuals" : {
      individualid__c : {
        [Values omitted]
      },
    }
  },
  "message" : "Number of Individuals retrieved: 1"
}
```

The fields and values returned are omitted here to save space. You can specify which fields to omit by using filtering.

Retrieve All Individuals

This request returns all individuals in your org. By default, a maximum of 200 individuals are returned at one time, but you can lower the number with the `limit` parameter. If there are more than 200 individuals, you can page through the results, retrieving all the individuals in batches from different offsets. The `offset` parameter specifies the end of the last batch retrieved. For example, to retrieve 100 individuals at a time:

1. Request the first 100 with an offset of 0.

2. Request the second 100 with an offset of 100.
3. Request the third 100 with an offset of 200.
4. Continue until you've retrieved all records.

The `nextRecordUrl` builds the next request for you. Here's the REST code for this example.

First, the initial GET request. We've omitted the offset value because it would be zero for a first request.

```
/services/apexrest/v1/individual/?limit=100
```

And then the response:

```
{
  "statusCode" : 8,
  "responseBody" : {
    "nextRecordUrl" : "/v1/individual/?limit=100&offset=100",
    "individuals" : {
      [Values omitted]
    }
  },
  "message" : " Number of Individuals retrieved: 100"
}
```

The status code tells us that the read operation isn't finished. The next record URL tells us how to retrieve the next batch of records. Our next request is:

```
/services/apexrest/v1/individual/?limit=100&offset=100
```

Notice how we used the previous response's `nextReordURL` value in the new request. The response is:

```
{
  "statusCode" : 8,
  "responseBody" : {
    "nextRecordUrl" : "/v1/individual/?limit=100&offset=200",
    "individuals" : {
      [Values omitted]
    }
  },
  "message" : " Number of Individuals retrieved: 100"
}
```

According to the status code, more records remain. The next record URL value is almost the same as the previous value. The only difference is that the offset is now 200, because we've read the first 200 records. If we have 240 records total, the next request gets the remainder.

```
/services/apexrest/v1/individual/?limit=100&offset=200
```

The response is:

```
{
  "statusCode" : 3,
  "responseBody" : {
    "nextRecordUrl" : "null",
    "individuals" : {
      [Values omitted]
    }
  },
```

```
  "message" : " Number of Individuals retrieved: 40"
}
```

The status code shows that the read is complete and the next record URL is not set because there are no more records. The message tells us that we retrieved 40 individuals, instead of the 100 we expected.

Retrieve Individuals with Changes

A common task is to get only those individuals with changes since a certain date. For example, you need all individuals that are new or have changes since the last quarterly report. You specify the date with the number of days to today from the past date. For example, today's date is May 11, 2016, and you want the records created or changed since January 1, 2016 for all individuals. Set `duration` to 131, the number of days from January 1 to May 11. You use the `limit`, `offset`, and `nextRecordUrl` parameters in the same way as when retrieving multiple individuals.

Let's get all the records created or modified in the last week. Here's the GET request.

```
/services/apexrest/v1/individual/?duration=7&limit=100&offset=0
```

And the response:

```
{
  "statusCode" : 8,
  "responseBody" : {
    "nextRecordUrl" : "/v1/individual/?duration=7&limit=100&offset=100",
    "individuals" : {
      [Values omitted]
    }
  },
  "message" : " Number of Individuals retrieved: 100"
}
```

We see from the status code that there are more records, and the `nextRecordUrl` value has prepared our next request.

```
/services/apexrest/v1/individual/?duration=7&limit=100&offset=100
```

Here's the response:

```
{
  "statusCode" : 3,
  "responseBody" : {
    "nextRecordUrl" : "null",
    "individuals" : {
      [Values omitted]
    }
  },
  "message" : " Number of Individuals retrieved: 95"
}
```

This time we got 95 records, so we know that 195 records were created or modified in the past week.

Post Operation

Create individuals one at a time or in batches. Use the POST method to create individuals in your org. You can create more than one individual at a time, but it's an all-or-none operation. If an error occurs while creating any individual's record, no individuals are created.

Individuals are created using a map of field names and field values. You include only the fields that must have values in your org. Here's an example that creates two individuals with minimal information, using an org with the namespace `clinic01`.

```
/services/apexrest/clinic01/v1/individual/
```

Here's the request body that defines the individuals to add. `lastname` is required here because it's required by the Salesforce Contact object. Contact is the parent object of an individual.

```
{
"individuals":
  [
    {
      "firstname" : "Robert",
      "lastname" : "Jones",
      "Email" : "RJ@gmail.com",
      "SourceSystemId__c" : "abc-123-4567"
    },
    {
      "firstname" : "Virgil",
      "lastname" : "James",
      "Email" : "James@gmail.com",
      "SourceSystemId__c" : "xyz-789-1234"
    }
  ]
}
```

The complete response contains all the individual's fields, including the ones we didn't set. The unset fields are omitted in this response for brevity.

```
{
  "statusCode" : 2,
  "responseBody" :
  [
    {
      "email" : "RJ@gmail.com",
      "firstname" : "Robert",
      "lastname" : "Jones",
      "accountid" : "001B000000AfFR6IAN",
      "clinic01__individualtype__c" : "Individual",
      "clinic01__sourcesystemid__c" : "abc-123-4567",
      "clinic01__primarycontact__c" : "003B0000006A5QKIA0",
      "recordtypeid" : "012B00000006W3cIAE",
      "id" : "003B0000006A5QKIA0"
    },
    {
      "email" : "James@gmail.com",
      "firstname" : "Virgil",
      "lastname" : "James",
      "accountid" : "001B000000AfFR7IAN",
      "clinic01__individualtype__c" : "Individual",
      "clinic01__sourcesystemid__c" : "xyz-789-1234",
      "clinic01__primarycontact__c" : "003B0000006A5QLIA0",
      "recordtypeid" : "012B00000006W3cIAE",
      "id" : "003B0000006A5QLIA0"
    }
  ],
```

```
  "message" : "Number of Individuals created successfully: 2"
}
```

Note: The `clinic01__individualtype__c` field represents the custom field, `individualtype__c`, which is for Salesforce internal-use only.

The status code of 2 tells us that the create operation was successful, and the message confirms it.

Put Operation

Make updates to individuals in your org, one at a time or in groups.

The PUT method allows partial updates, so if some updates fail but others succeed, the successful updates aren't rolled back. This behavior is different than POST, which is to roll back all changes if there is any error. The individual's ID value, `individualid__c`, uniquely identifies the record, so you must always include it in the request body. If you omit the ID value, the update request is ignored for that individual.

In the POST example, we created two records. Now let's modify those individuals to change their email addresses, again using an org with the namespace `clinic01`. The PUT method is `/services/apexrest/clinic01/v1/individual/`.

Here's the request body that identifies the individuals and lists the values to update.

Tip: Use the GET method with a duration of one day to retrieve the `individualid__c` values.

```
{
  "individuals" :
  [
    {
      "individualid__c" : "001B000000AfFR6IAN1455067013084",
      "email" : "RJ@yahoo.com"
    },
    {
      "individualid__c" : "001B000000AfFR7IAN1455067013084",
      "email" : "James@yahoo.com"
    }
  ]
}
```

Here's the complete response.

```
{
  "statusCode" : 4,
  "responseBody" :
  {
    "001B000000AfFR7IAN1455067013084" :
    {
      "errorFields" : "",
      "errorMessage" : "",
      "individualUpdated" : "true"
    },
    "001B000000AfFR6IAN1455067013084" :
    {
      "errorFields" : "",
      "errorMessage" : "",
      "individualUpdated" : "true"
    }
```

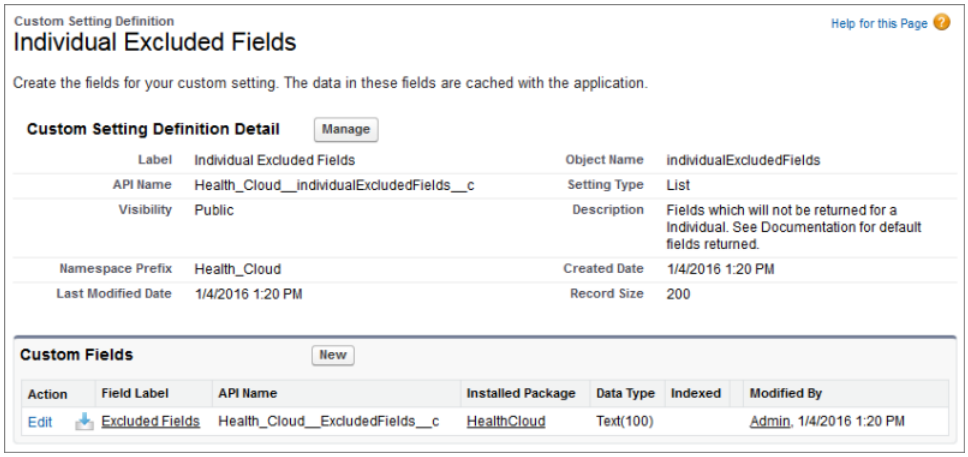
```
  },
  "message" : "Individuals Updated,Please check response body for detailed update status"
}
```

The status code of 4 tells us that the update operation was successful. Looking at the `individualUpdated` values in the response body tells us that all the updates were successful.

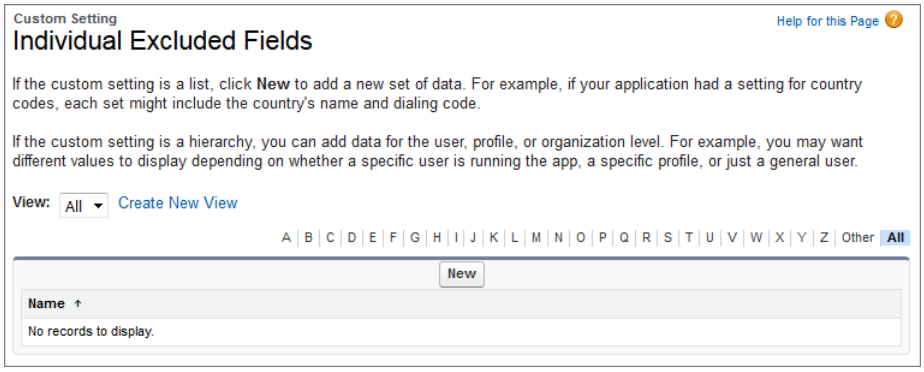
Filter Results

Filter the fields returned by a GET request so that you see only the values you want.

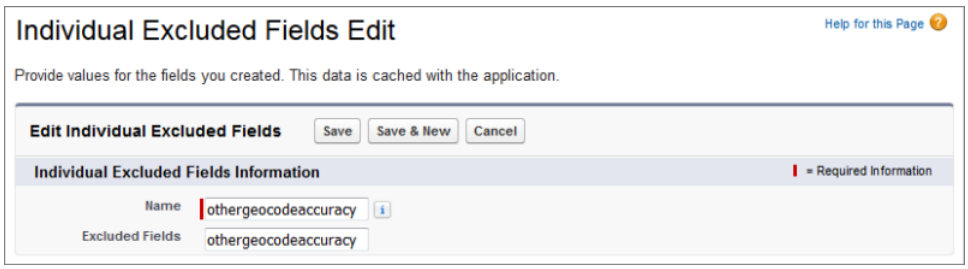
1. Enter *`Custom Settings`* in the Quick Find box or select **Develop** > **Custom Settings** from Setup.
2. Find **Individual Excluded Fields** in the Label column, and click the label.



3. Click **Manage** on the next screen.



4. Click **New** and enter a field name. Don't include the namespace as part of the field name.



For more information, search for “Custom Settings” in the Salesforce help.