Salesforce.com: Winter '11

Force.com REST API Developer's Guide

Last updated: October 22, 2010

Table of Contents

Chapter 1

Introducing the Force.com REST API

The Force.com REST API provides you with a powerful, convenient, and simple Web services interface for interacting with Salesforce.com. Its advantages include ease of integration and development, and it is an excellent choice of technology for use with mobile applications and Web 2.0 projects.

The REST API uses the same underlying data model and standard objects as those in the SOAP-based Web services API. See the Web Services API Developer's Guide for details.

To use this document, you should have a basic familiarity with software development, Web services, and the Salesforce.com user interface.

In the following sections you will learn:

- The key characteristics and architecture of the Force.com REST API. This will help you understand how your applications can best use the Force.com REST resources.
- How to set up your development environment so you can begin working with the REST API immediately.
- How to use the REST API by following a quick start that leads you step by step through a typical use case.

Note: This is an early access guide for Salesforce.com pilot customers only. To provide feedback on this document, email Alex Toussaint at atoussaint@salesforce.com.

Understanding Force.com REST Resources

Each resource in the Force.com REST API is a named URI that is used with an HTTP method (HEAD, GET, POST, PATCH, or DELETE). You use a resource to interact with your Salesforce.com organization. For example, you can retrieve summary information about the Salesforce.com versions available to you, obtain detailed information about a Salesforce.com object such as an Account, perform a query or search, or update records.

The Force.com REST API is based on the usage of resources, their URIs, and the links between them. The resources are accessed using a standard set of HTTP methods.

Suppose you would like to retrieve information about the Salesforce.com version. To do this, submit a request for the Versions resource (this example uses cURL on the `na1` instance):

```
curl http://na1.salesforce.com/services/data/
```

The output from this request is as follows:

```
[
    {
        "version":"20.0",
        "url":"/services/data/v20.0",
        "label":"Winter '11"
    }
]
```

Note: Salesforce.com runs on multiple server instances. The examples in this guide use the `na1` instance. The instance your organization uses may be different.

Important characteristics of the Force.com REST API resources and architecture:

Stateless
: Each request from client to server must contain all the information necessary to understand the request, and not use any stored context on the server. However, the representations of the resources are interconnected using URLs, which allow the client to progress between states.

Caching Behavior
: Responses are labeled as cacheable or non-cacheable.

Uniform interface
: All resources are accessed with a generic interface over HTTP.

Named resources
: All resources are named using a base URI that follows your Salesforce.com URI.

Layered components
: The Force.com REST API architecture allows for the existence of such intermediaries as proxy servers and gateways to exist between the client and the resources.

Authentication
: The Force.com REST API supports OAuth 2.0.

Support for JSON and XML
: The JavaScript Object Notation (JSON) format is supported with UTF-8. Date-time information is in ISO8601 format.

XML serialization is similar to the SOAP-based Web services API. XML requests are supported in UTF-8 and UTF-16, and XML responses are provided in UTF-8.

JSON is the default. You can use the `HTTP ACCEPT` header to select either JSON or XML, or append `.json` or `.xml` to the URI (for example, `/Account/001D000000INjVe.json`).

Using Compression

The REST API allows the use of compression on the request and the response, using the standards defined by the HTTP 1.1 specification. This is automatically supported by some clients, and can be manually added to others. Visit Developer Force for more information on particular clients.

For better performance, we suggest that clients accept and support compression as defined by the HTTP 1.1 specification.

To use compression, include the HTTP header `Accept-Encoding: gzip` or `Accept-Encoding: deflate` in a request. The REST API compresses the response if the client properly specifies this header. The response includes the header `Content-Encoding: gzip` or `Accept-Encoding: deflate`. You can also compress any request by including a `Content-Encoding: gzip` or `Content-Encoding: deflate` header.

Response Compression

The REST API can optionally compress responses. Responses are compressed only if the client sends an `Accept-Encoding` header. The REST API is not required to compress the response even if you have specified `Accept-Encoding`, but it normally does. If the REST API compresses the response, it also specifies a `Content-Encoding` header.

Request Compression

Clients can also compress requests. The REST API decompresses any requests before processing. The client must send a `Content-Encoding` HTTP header in the request with the name of the appropriate compression algorithm. For more information, see:

- Content-Encoding at: `www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.11`
- Accept-Encoding at: `www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.3`
- Content Codings at: `www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.5`

Chapter 2

Quick Start

Create a sample REST application in your development environment to see the power and flexibility of the REST API.

Prerequisites

Completing the prerequisites makes it easier to build and use the quick-start sample.

- Install your development platform according to its product documentation.
- Become familiar with cURL, the tool used to execute REST requests in this quick start. If you use another tool, you should be familiar enough with it to translate the example code.
- Become familiar with JavaScript Object Notation (JSON), which is used in this quick start, or be able to translate samples from JSON to the standard you use.
- Become familiar with OAuth 2.0, which requires some setup. We provide the steps, but it will help if you are familiar with the basic concepts and workflow.
- Read through all the steps before beginning this quick start. You may also wish to review the rest of this document to familiarize yourself with terms and concepts.

Step One: Obtain a Salesforce.com Developer Edition Account

If you are not already a member of the developer community, go to `http://developer.force.com/join` and follow the instructions for signing up for a Developer Edition account. Even if you already have an Enterprise Edition or Unlimited Edition account, it is strongly recommended that you use Developer Edition for developing, staging, and testing your solutions against sample data to protect your organization's live data. This is especially true for applications that will be inserting, updating, or deleting data (as opposed to simply reading data).

If you already have a Developer Edition account, verify that your user profile has the "API Enabled" permission selected. This permission is enabled by default. For more information see the help in the Salesforce.com user interface.

Step Two: Set Up Authorization

You can set up authorization using OAuth 2.0 or by passing a session ID.

Important: If you're handling someone else's password, don't use session ID.

Setting Up OAuth 2.0

Setting up OAuth 2.0 requires that you take some steps within the Salesforce.com app and in other locations. If any of the steps are unfamiliar, you can consult the Salesforce.com online help or OAuth 2.0 documentation.

Note: If you are unable to use OAuth, you can use a session ID instead of the access token

1. In Salesforce.com, navigate to **Setup** ➤ **Develop** ➤ **Remote Access**, and click **New** to create a new remote access application if you have not already done so. The `Callback URL` you supply here is the same as your Web application's callback URL. Usually it is a servlet if you work with Java. It must be secure: `http://` does not work, only `https://`. When you click Save, the `Consumer Key` is created and displayed, and a `Consumer Secret` is created (click the link to reveal it). The values here correspond to the following values in the sample code in the rest of this procedure:

   - `client_id` is the `Consumer Key`
   - `client_secret` is the `Consumer Secret`
   - `redirect_uri` is the `Callback URL`.

   There is one additional value you specify in the code sample, the `grant_type`. For OAuth 2.0, the value is `authorization_code` as shown in the sample.

2. From your Java or other client application, make a request to the authentication URL that passes in `grant_type`, `client_id`, `client_secret`, and `redirect_uri`, which is the URI that Salesforce.com sends a callback to. For example:

```
initParams = {
    @WebInitParam(name = "clientId", value =
"3MVG9lKcPoNINVBJSoQsNCD.HHDdbugPsNXwwyFbgb47KWa_PTv"),
    @WebInitParam(name = "clientSecret", value = "5678471853609579508"),
    @WebInitParam(name = "redirectUri", value =
"https://localhost:8443/RestTest/oauth/_callback"),
    @WebInitParam(name = "environment", value =
"https://na1.salesforce.com/services/oauth2/token")  }

HttpClient httpclient = new HttpClient();
PostMethod post = new PostMethod(environment);
post.addParameter("code",code);
post.addParameter("grant_type","authorization_code");
   /** For session ID instead of OAuth 2.0, use "grant_type", "password" */
post.addParameter("client_id",clientId);
post.addParameter("client_secret",clientSecret);
post.addParameter("redirect_uri",redirectUri);
```

   If the value of `client_id` (or `consumer key`) and `client_secret` (or `consumer secret`) are valid, Salesforce.com sends a callback to the URI specified in `redirect_uri` that contains a value for `access_token`.

3. Store the access token value as a cookie to use in all subsequent requests. For example:

```
//exception handling removed for brevity...
  //this is the post from step 2
  httpclient.executeMethod(post);
     String responseBody = post.getResponseBodyAsString();

  String accessToken = null;
  JSONObject json = null;
   try {
       json = new JSONObject(responseBody);
         accessToken = json.getString("access_token");
         issuedAt = json.getString("issued_at");
         /** Use this to validate session
          * instead of expiring on browser close.
          */

         } catch (JSONException e) {
            e.printStackTrace();
         }

         HttpServletResponse httpResponse = (HttpServletResponse)response;
          Cookie session = new Cookie(ACCESS_TOKEN, accessToken);
         session.setMaxAge(-1); //cookie not persistent, destroyed on browser exit
         httpResponse.addCookie(session);
```

This completes the authentication.

4. Once authenticated, every request made to the REST service must pass in the `access_token` value in the header. It cannot be passed as a request parameter.

```
HttpClient httpclient = new HttpClient();
   GetMethod gm = new GetMethod(serviceUrl);

   //set the token in the header
   gm.setRequestHeader("Authorization", "OAuth "+accessToken);
   //set the SOQL as a query param
   NameValuePair[] params = new NameValuePair[1];

   /**
    * other option instead of query string, pass just the fields you want back:
    *  https://instance_name.salesforce.com/services/data/v20.0/sobjects/Account/
    *       001D000000INjVe?fields=AccountNumber,BillingPostalCode
    */
   params[0] = new NameValuePair("q","SELECT name, title FROM Contact LIMIT 100");
   gm.setQueryString(params);

   httpclient.executeMethod(gm);
   String responseBody = gm.getResponseBodyAsString();
       //exception handling removed for brevity
   JSONObject json = new JSONObject(responseBody);

   JSONArray results = json.getJSONArray("records");

   for(int i = 0; i < results.length(); i++)
       response.getWriter().write(results.getJSONObject(i).getString("Name")+     ",
         "+results.getJSONObject(i).getString("Title")+"\n");
```

The syntax to provide the access token in your REST requests:

```
Authorization: OAuth token
```

For example:

```
curl http://instance_name.salesforce.com/services/data/v20.0/ -H "Authorization: OAuth
token"
```

Session ID Authorization

You can use a session ID instead of an OAuth 2.0 access token if you aren't handling someone else's password:

1. Obtain a session ID, for example, a SOAP Web services API `login()` call returns the session ID. You may also have the session ID, for example as part of the Apex current context.
2. Use the session ID when you send a request to the resource. Substitute the ID for the *`token`* value. The syntax is the same:

   ```
   Authorization: OAuth token
   ```

   For example:

   ```
   curl http://instance_name.salesforce.com/services/data/v20.0/ -H "Authorization: OAuth
   token"
   ```

Step Three: Send HTTP Requests with cURL

To interact with the Force.com REST API, you need to set up your client application (we use cURL) to construct HTTP requests.

Setting Up Your Client Application

The REST API uses HTTP GET and HTTP POST methods to send and receive CSV and XML content, so it is very simple to build client applications using the tool or the language of your choice. We use a command-line tool called cURL to simplify sending and receiving HTTP requests and responses.

cURL is pre-installed on many Linux and Mac systems. Windows users can download a version at `curl.haxx.se/`. When using HTTPS on Windows, ensure that your system meets the cURL requirements for SSL.

Sending HTTP Requests Using REST API Resources

Your HTTP requests to a REST API resource should contain the following information:

- An HTTP method (HEAD, GET, POST, PATCH, or DELETE).
- An OAuth 2.0 access token used to authenticate the request. For information on how to retrieve the token, see Quick Start on page 6.
- An HTTP ACCEPT header used to indicate the resource format (XML or JSON), or a `.json` or `.xml` extension to the URI. The default is JSON.
- The Force.com REST resource.
- Any JSON or XML files containing information needed for requests, such as updating a record with new information.

The HTTP methods are used to indicate the desired action, such as retrieving information, as well as creating, updating, and deleting records.

- HEAD is used to retrieve resource metadata.

- GET is used to retrieve information, such as basic resource summary information.
- POST is used to create a new object.
- PATCH is used to update a record.
- DELETE is used to delete a record.

To access a resource, submit an HTTP request containing a header, method, and resource name.

For example, assume you want to create an Account record using a JSON-formatted file called `newaccount.json`. It contains the information to be stored in the new account:

```
{
    "Name" : "test"
}
```

Using cURL on instance na1, the request would appear as follows:

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/ -H "Content-Type:
application/json"
-d @newaccount.json -H "Authorization: OAuth token" -H "X-PrettyPrint:1"
```

The request HTTP header:

```
POST /services/data/v20.0/sobjects/Account HTTP/1.1
User-Agent: curl/7.19.7 (universal-apple-darwin10.0) libcurl/7.19.7 OpenSSL/0.9.8l zlib/1.2.3
Host: na7.salesforce.com
Accept: */*
Content-Length: 1411
Content-Type: application/json
Authorization: OAuth XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X-PrettyPrint:1
```

The response:

```
Date: Thu, 21 Oct 2010 22:16:22 GMT
Content-Length: 71
Location: /services/data/v20.0/sobjects/Account/001T000000NU96UIAT
Content-Type: application/json; charset=UTF-8 Server:
{ "id" : "001T000000NU96UIAT",
  "errors" : [ ],
  "success" : true }
```

For a list of the resources and their corresponding URIs, see Reference on page 26.

Step Four: Walk Through the Sample Code

In this section you will create a series of REST requests. cURL will be used to construct the requests, and JSON will be used as the format for all requests and responses. In each request, a base URI will be used in conjunction with the REST resource. The base URI for these examples will be `http://na1.salesforce.com/services/data`. For more information, see Understanding Force.com REST Resources on page 3.

In this example, a series of REST requests will be used in the following scenario:

1. Get the Salesforce.com version.
2. Use the Salesforce.com version to get a list of the resources available.
3. Use one of the resources to get a list of the available objects.

4. Select one of the objects and get a description of its metadata.
5. Get a list of fields on that same object.
6. Execute a SOQL query to retrieve values from all `name` fields on Account records.
7. Update the Billing City for one of the Account objects.

Get the Salesforce.com Version

Begin by retrieving information about each available Salesforce.com version. To do this, submit a request for the Versions resource. In this case the request does not require authentication:

```
curl http://na1.salesforce.com/services/data/
```

The output from this request, including the response header:

```
Date: Thu, 21 Oct 2010 22:46:54 GMT
Content-Length: 88
Content-Type: application/json;
charset=UTF-8 Server:
[
    {
        "version":"20.0",
        "url":"/services/data/v20.0",
        "label":"Winter '11"
    }
]
```

The output specifies the resources available for all valid versions (your result may include more than one value). Next, use one of these versions to discover the resources it contains.

Get a List of Resources

The next step is to retrieve a list of the resources available for Salesforce.com, in this example for version 20.0. To do this, submit a request for the Resources by Version:

```
curl http://na1.salesforce.com/services/data/v20.0/ -H "Authorization: OAuth access_token"
 -H "X-PrettyPrint:1"
```

The output from this request is as follows:

```
{
    "sobjects" : "/services/data/v20.0/sobjects",
    "search" : "/services/data/v20.0/search",
    "query" : "/services/data/v20.0/query",
    "recent" : "/services/data/v20.0/recent"
}
```

From this output you can see that `sobjects` is one of the available resources in Salesforce.com version 20.0. You will be able to use this resource in the next request to retrieve the available objects.

Get a List of Available Objects

Now that you have the list of available resources, you can request a list of the available objects. To do this, submit a request for the Describe Global:

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/ -H "Authorization: OAuth
access_token" -H "X-PrettyPrint:1"
```

The output from this request is as follows:

```
Date: Thu, 21 Oct 2010 22:48:18 GMT
Transfer-Encoding: chunked
Content-Type: application/json;
charset=UTF-8 Server:
{
 "encoding" : "UTF-8",
 "maxBatchSize" : 200,
 "sobjects" : [ {
    "name" : "Account",
    "label" : "Account",
    "custom" : false,
    "keyPrefix" : "001",
    "updateable" : true,
    "searchable" : true,
    "labelPlural" : "Accounts",
    "layoutable" : true,
    "activateable" : false,
    "urls" : { "sobject" : "/services/data/v20.0/sobjects/Account",
    "describe" : "/services/data/v20.0/sobjects/Account/describe",
    "rowTemplate" : "/services/data/v20.0/sobjects/Account/{ID}" },
    "createable" : true,
    "customSetting" : false,
    "deletable" : true,
    "deprecatedAndHidden" : false,
    "feedEnabled" : false,
    "mergeable" : true,
    "queryable" : true,
    "replicateable" : true,
    "retrieveable" : true,
    "undeletable" : true,
    "triggerable" : true },
   },
...
```

From this output you can see that the Account object is available. You will be able to get more information about the Account object in the next steps.

Get Basic Object Information

Now that you have identified the Account object as an available resource, you can retrieve some basic information about its metadata. To do this, submit a request for the SObject Basic Information:

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/ -H "Authorization:
OAuth access_token" -H "X-PrettyPrint:1"
```

The output from this request is as follows:

```
{
    "objectDescribe" :
    {
        "name" : "Account",
        "updateable" : true,
        "label" : "Account",
        "keyPrefix" : "001",

        ...

        "replicateable" : true,
        "retrieveable" : true,
        "undeletable" : true,
        "triggerable" : true
    },
    "recentItems" :
    [
        {
            "attributes" :
            {
                "type" : "Account",
                "url" : "/services/data/v20.0/sobjects/Account/001D000000INjVeIAL"
            },
            "Id" : "001D000000INjVeIAL",
            "Name" : "asdasdasd"
        },

        ...

    ]
}
```

From this output you can see some basic attributes of the Account object, such as its name and label, as well as a list of the most recently used Accounts. Since you may need more information about its fields, such as length and default values, in the next step you will retrieve more detailed information about the Account object.

Get a List of Fields

Now that you have some basic information about the Account object's metadata, you may be interested in retrieving more detailed information. To do this, submit a request for the SObject Describe:

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/describe/ -H
"Authorization: OAuth access_token" -H "X-PrettyPrint:1"
```

The output from this request is as follows:

```
{
    "name" : "Account",
    "fields" :
    [
        {
            "length" : 18,
            "name" : "Id",
            "type" : "id",
            "defaultValue" : { "value" : null },
            "updateable" : false,
            "label" : "Account ID",
            ...
        },
        ...
    ],
    "updateable" : true,
    "label" : "Account",
    ...
    "urls" :
    {
        "uiEditTemplate" : "https://na1-blitz03.soma.salesforce.com/{ID}/e",
        "sobject" : "/services/data/v20.0/sobjects/Account",
        "uiDetailTemplate" : "https://na1-blitz03.soma.salesforce.com/{ID}",
        "describe" : "/services/data/v20.0/sobjects/Account/describe",
        "rowTemplate" : "/services/data/v20.0/sobjects/Account/{ID}",
        "uiNewRecord" : "https://na1-blitz03.soma.salesforce.com/001/e"
    },
    "childRelationships" :
    [
        {
            "field" : "ParentId",
            "deprecatedAndHidden" : false,
            ...
        },
        ...
    ],

    "createable" : true,
    "customSetting" : false,
    ...
}
```

From this output you can see much more detailed information about the Account object, such as its field attributes and child relationships. Now you have enough information to construct useful queries and updates for the Account objects in your organization, which you will do in the next steps.

Execute a SOQL Query

Now that you know the field names on the Account object, you can execute a SOQL query, for example, to retrieve a list of all the Account name values. To do this, submit a Query request:

```
curl http://na1.salesforce.com/services/data/v20.0/query?q=SELECT+name+from+Account -H
"Authorization: OAuth access_token" -H "X-PrettyPrint:1"
```

The output from this request is as follows:

```
{
    "done" : true,
    "totalSize" : 14,
    "records" :
    [
        {
            "attributes" :
            {
                "type" : "Account",
                "url" : "/services/data/v20.0/sobjects/Account/001D000000IRFmaIAH"
            },
            "Name" : "Test 1"
        },
        {
            "attributes" :
            {
                "type" : "Account",
                "url" : "/services/data/v20.0/sobjects/Account/001D000000IomazIAB"
            },
            "Name" : "Test 2"
        },
        ...
    ]
}
```

From this output you have a listing of the available Account names, and each name's preceding attributes include the Account IDs. In the next step you will use this information to update one of the accounts.

Note: You can find more information about SOQL in the Force.com *Web Services API Developer's Guide*.

Update a Field on a Record

Now that you have the Account names and IDs, you can retrieve one of the accounts and update its Billing City. To do this, you will need to submit an SObject Row request. To update the object, supply the new information about the Billing City. Create a text file called `patchaccount.json` containing the following information:

```
{
    "BillingCity" : "Fremont"
}
```

Specify this JSON file in the REST request. The cURL notation requires the `—d` option when specifying data. For more information, see http://curl.haxx.se/docs/manpage.html.

Also, specify the `PATCH` method, which is used for updating a REST resource. The following cURL command retrieves the specified Account object using its ID field, and updates its Billing City.

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/001D000000IroHJ -H
"Authorization: OAuth access_token" -H "X-PrettyPrint:1" -H "Content-Type: application/json"
 --data-binary @patchaccount.json -X PATCH
```

Refresh the page on the account and you will see that the Billing Address has changed to Fremont.

Chapter 3

Working with Objects and Records

In this chapter ...

The examples in this section explain how to use REST Resources to perform tasks such as creating, updating, or deleting a record.

For complete reference, see Reference on page 26.

List Available REST API Versions

Lists summary information about each REST API version currently available, including the version, label, and a link to each version's root. You do not need authentication to retrieve the list of versions.

Example usage

```
curl http://na1.salesforce.com/services/data/
```

Example request body

none required

Example response body

```
[
    {
        "label":"Winter '10"
        "version":"20.0",
        "url":"/services/data/v20.0",
    }
]
```

List Available REST Resources

List the resources available for the specified API version. It provides the name and URI of each resource.

Example

```
curl http://na1.salesforce.com/services/data/v20.0/ -H "Authorization: OAuth token" -H
"X-PrettyPrint:1"
```

Example request body

none required

Example response body

```
{
    "sobjects" : "/services/data/v20.0/sobjects",
    "search" : "/services/data/v20.0/search",
    "query" : "/services/data/v20.0/query",
    "recent" : "/services/data/v20.0/recent"
}
```

Get a List of Objects

List the objects that are available in your organization and available to the logged-in user. This request also returns the organization encoding as well as maximum batch size permitted in queries.

Example usage

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/ -H "Authorization: OAuth
token" -H "X-PrettyPrint:1"
```

Example request body

none required

Example response body

```
Date: Thu, 21 Oct 2010 22:48:18 GMT
Transfer-Encoding: chunked
Content-Type: application/json;
charset=UTF-8 Server:
{
 "encoding" : "UTF-8",
 "maxBatchSize" : 200,
 "sobjects" : [ {
    "name" : "Account",
    "label" : "Account",
    "custom" : false,
    "keyPrefix" : "001",
    "updateable" : true,
    "searchable" : true,
    "labelPlural" : "Accounts",
    "layoutable" : true,
    "activateable" : false,
    "urls" : { "sobject" : "/services/data/v20.0/sobjects/Account",
    "describe" : "/services/data/v20.0/sobjects/Account/describe",
    "rowTemplate" : "/services/data/v20.0/sobjects/Account/{ID}" },
    "createable" : true,
    "customSetting" : false,
    "deletable" : true,
    "deprecatedAndHidden" : false,
    "feedEnabled" : false,
    "mergeable" : true,
    "queryable" : true,
    "replicateable" : true,
    "retrieveable" : true,
    "undeletable" : true,
    "triggerable" : true },
   },
...
```

Retrieve Metadata for an Object

Retrieve metadata for an object using the HTTP GET method.

Example for retrieving Account metadata

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/ -H "Authorization:
OAuth token" -H "X-PrettyPrint:1"
```

Example request body for retrieving Account metadata

none required

Example response body for retrieving Account metadata

```
{
    "objectDescribe" :
    {
        "name" : "Account",
        "updateable" : true,
        "label" : "Account",
        "keyPrefix" : "001",

        ...

        "replicateable" : true,
        "retrieveable" : true,
        "undeletable" : true,
        "triggerable" : true
    },
    "recentItems" :
    [
        {
            "attributes" :
            {
                "type" : "Account",
                "url" : "/services/data/v20.0/sobjects/Account/001D000000INjVeIAL"
            },
            "Id" : "001D000000INjVeIAL",
            "Name" : "asdasdasd"
        },

        ...

    ]
}
```

To get a complete description of an object, including field names and their metadata, see Get a List of Objects on page 17.

Get Field and Other Metadata for an Object

Retrieve all the metadata for an object, including information about each field, URLs, and child relationships.

Example

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/describe/ -H
"Authorization: OAuth token" -H "X-PrettyPrint:1"
```

Example request body

none required

Example response body

```
{
    "name" : "Account",
    "fields" :
    [
        {
            "length" : 18,
            "name" : "Id",
            "type" : "id",
            "defaultValue" : {    "value" : null  },
            "updateable" : false,
            "label" : "Account ID",
            ...
        },

        ...

    ],


    "updateable" : true,
    "label" : "Account",
    "keyPrefix" : "001",
    "custom" : false,

    ...

    "urls" :
    {
        "uiEditTemplate" : "https://na1.salesforce.com/{ID}/e",
        "sobject" : "/services/data/v20.0/sobjects/Account",
        "uiDetailTemplate" : "https://na1.salesforce.com/{ID}",
        ...
    },

    "childRelationships" :
    [
        {
            "field" : "ParentId",
            "deprecatedAndHidden" : false,
            ...
        },

        ....

    ],

    "createable" : true,
    "customSetting" : false,
    ...
}
```

Get Field Values from Records

Retrieve the value from fields on a record using the HTTP GET.

Example for retrieving values from fields on an Account object

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/001D000000INjVe
?fields=AccountNumber,BillingPostalCode -H "Authorization: OAuth token" -H
"X-PrettyPrint:1"
```

Example request body

None required

Example response body

```
{
    "AccountNumber" : "CD656092",
    "BillingPostalCode" : "27215",
}
```

Get Attachment Content from a Record

Retrieve the specified base64 field from an individual record's attachment.

Example

```
curl
http://na1.salesforce.com/services/data/v20.0/sobjects/attachment/001D000000INjVe/body
-H "Authorization: OAuth token" -H "X-PrettyPrint:1"
```

Example request body

none required

Example response body

Content is contained in the attachment.

Because data is binary, you can't use JSON or XML.

Execute a SOQL Query

You can execute a SOQL query that returns all the results in a single response, or if needed, returns part of the results and an identifier used to retrieve the remaining results.

Example usage for executing a query

The following query requests the value from `name` fields from all Account records.

```
curl http://na1.salesforce.com/services/data/v20.0/query/?q=SELECT+name+from+Account
-H "Authorization: OAuth token" -H "X-PrettyPrint:1"
```

Example request body for executing a query

none required

Example response body for executing a query

```
{
    "done" : true,
    "totalSize" : 14,
    "records" :
    [
        {
            "attributes" :
            {
                "type" : "Account",
                "url" : "/services/data/v20.0/sobjects/Account/001D000000IRFmaIAH"
            },
            "Name" : "Test 1"
        },
        {
            "attributes" :
```

```
            {
                "type" : "Account",
                "url" : "/services/data/v20.0/sobjects/Account/001D000000IomazIAB"
            },
            "Name" : "Test 2"
        },

        ...

    ]
}
```

Retrieving the Remaining SOQL Query Results

If the initial query returns only part of the results, the end of the response will contain a field called `nextRecordsUrl`. For example, you might find this attribute at the end of your query:

```
"nextRecordsUrl" : "/services/data/v20.0/query/01gD0000002HU6KIAW-2000"
```

In such cases, request the next batch of records and repeat until all records have been retrieved. These requests use `nextRecordsUrl`, and do not include any parameters.

Example usage for retrieving the remaining query results

```
curl http://na1.salesforce.com/services/data/v20.0/query/01gD0000002HU6KIAW-2000 -H
"Authorization: OAuth token" -H "X-PrettyPrint:1"
```

Example request body for retrieving the remaining query results

none required

Example response body for retrieving the remaining query results

```
{
    "done" : true,
    "totalSize" : 3214,
    "records" : [...]
}
```

Search for a String

Execute the specified SOSL search. The search string, in this example `{test}`, must be URL-encoded.

Example usage

```
curl http://na1.salesforce.com/services/data/v20.0/search/?q=FIND+%7Btest%7D -H
"Authorization: OAuth token" -H "X-PrettyPrint:1"
```

Example request body

none required

Example response body

```
[
    {
        "attributes" :
         {
            "type" : "Account",
            "url" : "/services/data/v20.0/sobjects/Account/001D000000IqhSLIAZ"
        },
        "Id" : "001D000000IqhSLIAZ"
    },
    {
        "attributes" :
        {
            "type" : "Account",
            "url" : "/services/data/v20.0/sobjects/Account/001D000000IomazIAB"
        },
        "Id" : "001D000000IomazIAB"
    }
]
```

Create a Record

Use the HTTP POST method to create a new record.

Example cURL command

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/ -H "Content-Type:
application/json" -d @newaccount.json -H "Authorization: OAuth token" -H
"X-PrettyPrint:1"
```

Example request body for posting a new Account (contents of `newaccount.json`)

```
{
    "Name" : "test"
}
```

Example response body for posting a new Account

```
{
    "id" : "001D000000IqhSLIAZ",
    "errors" : [ ],
    "success" : true
}
```

Error responses

See Error Response on page 38.

Creating records with Patch

The following example shows how to use PATCH with Apache's HttpClient, even though there is not yet a PatchMethod class in HttpClient.

```
public static void patch(String url, String sid) throws IOException {
 PostMethod m = new PostMethod(url) {
  @Override public String getName() { return "PATCH"; }
 };
```

```
  m.setRequestHeader("Authorization", "OAuth " + sid);

  Map<String, Object> accUpdate = new HashMap<String, Object>();
  accUpdate.put("Name", "Patch test");
  accUpdate.put("AnnualRevenue", 10);
  ObjectMapper mapper = new ObjectMapper();
  m.setRequestEntity(new StringRequestEntity(mapper.writeValueAsString(accUpdate),
"application/json", "UTF-8"));

  HttpClient c = new HttpClient();
  int sc = c.executeMethod(m);
  System.out.println("PATCH call returned a status code of " + sc);
  if (sc > 299) {
   // deserialize the returned error message
   List<ApiError> errors = mapper.readValue(m.getResponseBodyAsStream(), new
TypeReference<List<ApiError>>() {} );
   for (ApiError e : errors)
    System.out.println(e.errorCode + " " + e.message);
  }
 }

 private static class ApiError {
  public String errorCode;
  public String message;
  public String [] fields;
 }
```

This example uses the Jackson JSON library, especially useful for Java clients.

If you use an HTTP library that doesn't allow overriding or setting an arbitrary HTTP method name, you can send a POST request and provide an override to the HTTP method via the query string parameter `_HttpMethod`. In the PATCH example, you can replace the PostMethod line with one that doesn't use override:

```
PostMethod m = new PostMethod(url + "?_HttpMethod=PATCH");
```

Update a Record

Update a record using HTTP PATCH. First, create a file that contains the changes, then specify it in the cURL command.Records in a single file must be of the same object type.

Example for updating two fields in an Account object

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/001D000000INjVe
-H "Authorization: OAuth token" -H "X-PrettyPrint:1" -H "Content-Type: application/json"
-d @patchaccount.json -X PATCH
```

Example request body for updating fields in an Account object

```
{
    "BillingCity" : "San Francisco"
}
```

Example response body for updating fields in an Account object

none returned

Error response

See Error Response on page 38.

Delete a Record

Delete a record using HTTP DELETE.

You can use the DELETE method to delete an object.

Example for deleting an Account record

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/001D000000INjVe
-H "Authorization: OAuth token" -H "X-PrettyPrint:1" -H "Authorization: OAuth token"
-H "X-PrettyPrint:1" -X DELETE
```

Example request body

None needed

Example response body

None returned

Chapter 4

Reference

The table lists supported REST resources in the API in alphabetical order, and provides a brief description for each. In each case, the URI for the resource follows the base URI, which you retrieve from the authentication service: `http://`***`domain`***`/services/data`. ***`domain`*** might be the Salesforce.com instance you are using, or a custom domain. For example, to retrieve basic information about an Account object in version 20.0: `http://na1.salesforce.com/services/data/v20.0/sobjects/Account/`.

Click a call name to see syntax, usage, and more information for that call.

Resource Name	URI	Description
Versions	`/`	Lists summary information about each Salesforce.com version currently available, including the version, label, and a link to each version's root.
Resources by Version	`/vXX.X/`	Lists the resources available for the specified API version. It provides the name and URI of each resource.
Describe Global	`/vXX.X/sobjects/`	Lists the available objects and their metadata for your organization's data.
SObject Basic Information	`/vXX.X/sobjects/`***`SObject`***`/`	Describes the individual metadata for the specified object.
SObject Describe	`/vXX.X/sobjects/`***`SObject`***`/describe/`	Completely describes the individual metadata at all levels for the specified object.
SObject Row	`/vXX.X/sobjects/`***`SObject`***`/`***`id`***`/`	Accesses individual records from an object based on the specified object ID. You can retrieve or update individual records within a specific object, or delete the specified object.
SObject Blob Retrieve	`/vXX.X/sobjects/`***`SObject`***`/`***`id`***`/`***`blobField`***	Retrieves the specified blob field from an individual record.
Query	`/vXX.X/query/?q=`***`soql`***	Executes the specified SOQL query.

Resource Name	URI	Description
Search	`/vXX.X/search/?s=`***`sosl`***	Executes the specified SOSL search. The search string must be URL-encoded.

Versions

Lists summary information about each Salesforce.com version currently available, including the version, label, and a link to each version's root.

URI

`/`

Formats

JSON, XML

HTTP Method

GET

Authentication

none

Parameters

none

Example usage

`curl http://na1.salesforce.com/services/data/`

Example request body

none required

Example response body

```
[
    {
        "label":"Winter '10"
        "version":"20.0",
        "url":"/services/data/v20.0",
    }
]
```

Resources by Version

Lists the resources available for the specified API version. It provides the name and URI of each resource.

URI

`/vXX.X/`

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: OAuth` ***`token`***

Parameters

none

Example

```
curl http://na1.salesforce.com/services/data/v20.0/ -H "Authorization: OAuth token" -H
"X-PrettyPrint:1"
```

Example request body

none required

Example response body

```
{
    "sobjects" : "/services/data/v20.0/sobjects",
    "search" : "/services/data/v20.0/search",
    "query" : "/services/data/v20.0/query",
    "recent" : "/services/data/v20.0/recent"
}
```

Describe Global

Lists the available objects and their metadata for your organization's data. In addition, it provides the organization encoding, as well as maximum batch size permitted in queries. For more information, see Internationalization and Character Sets.

URI

`/vXX.X/sobjects/`

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: OAuth` ***`token`***

Parameters

none required

Example usage

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/ -H "Authorization: OAuth
token" -H "X-PrettyPrint:1"
```

Example request body

none required

Example response body

```
{
    "encoding" : "UTF-8",
    "maxBatchSize" : 200,
    "sobjects" : [

        {
            "name" : "Account",
            "updateable" : true,
            "label" : "Account",
            "keyPrefix" : "001",
            ...
        }

        {
            "name" : "AccountContactRole",
            "updateable" : true,
            "label" : "Contact Role",
            "keyPrefix" : "02Z",
            ...
        }

        ...

    ]
}
```

Error responses

See Error Response on page 38.

SObject Basic Information

Describes the individual metadata for the specified object. For example, this can be used to retrieve the metadata for the Account object or post a new Account object.

URI

`/vXX.X/sobjects/`***`SObjectName`***`/`

Formats

JSON, XML

HTTP Method

GET, POST

Authentication

`Authorization: OAuth` ***`token`***

Parameters

none required

Retrieving Account Metadata

You can use the GET method to retrieve the metadata for an object. In this example, the Account object's metadata are retrieved.

Example usage for retrieving Account metadata

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/ -H "Authorization:
OAuth token" -H "X-PrettyPrint:1"
```

Example request body for retrieving Account metadata

none required

Example response body for retrieving Account metadata

```
{
    "objectDescribe" :
    {
        "name" : "Account",
        "updateable" : true,
        "label" : "Account",
        "keyPrefix" : "001",

        ...

        "replicateable" : true,
        "retrieveable" : true,
        "undeletable" : true,
        "triggerable" : true
    },
    "recentItems" :
    [
        {
            "attributes" :
            {
                "type" : "Account",
                "url" : "/services/data/v20.0/sobjects/Account/001D000000INjVeIAL"
            },
            "Id" : "001D000000INjVeIAL",
            "Name" : "asdasdasd"
        },

        ...

    ]
}
```

Error responses

See Error Response on page 38.

Posting a new Account

You can use the POST method to create a new object. In this example, a new Account object is created.

Syntax for posting a new Account

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/ -H "Content-Type:
application/json" -d @newaccount.json -H "Authorization: OAuth token" -H
"X-PrettyPrint:1"
```

Example request body for posting a new Account

```
{
    "Name" : "test"
}
```

Example response body for posting a new Account

```
{
    "id" : "001D000000IqhSLIAZ",
    "errors" : [ ],
    "success" : true
}
```

Example error response

```
{
  "fields" : [ "Id" ],
  "message" : "Account ID: id value of incorrect type: 001900K0001pPuOAAU",
  "errorCode" : "MALFORMED_ID"
}
```

For more information about error codes, see Error Response on page 38.

SObject Describe

Completely describes the individual metadata at all levels for the specified object. For example, this can be used to retrieve the fields, URLs, and child relationships for the Account object.

URI

`/vXX.X/sobjects/SObjectName/describe/`

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: OAuth token`

Parameters

none required

Example usage

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/describe/ -H
"Authorization: OAuth token" -H "X-PrettyPrint:1"
```

Example request body

none required

Example response body

```
{
    "name" : "Account",
    "fields" :
    [
        {
            "length" : 18,
            "name" : "Id",
            "type" : "id",
            "defaultValue" : {    "value" : null  },
            "updateable" : false,
            "label" : "Account ID",
            ...
        },

        ...

    ],


    "updateable" : true,
    "label" : "Account",
    "keyPrefix" : "001",
    "custom" : false,

    ...

    "urls" :
    {
        "uiEditTemplate" : "https://na1.salesforce.com/{ID}/e",
        "sobject" : "/services/data/v20.0/sobjects/Account",
        "uiDetailTemplate" : "https://na1.salesforce.com/{ID}",
        ...
    },

    "childRelationships" :
    [
        {
            "field" : "ParentId",
            "deprecatedAndHidden" : false,
            ...
        },

        ....

    ],

    "createable" : true,
    "customSetting" : false,
    ...
}
```

SObject Row

Accesses individual records from an object based on the specified object ID. You can retrieve or update individual records within a specific object, or delete the specified object.

URI

`/vXX.X/sobjects/`***`SObjectName`***`/`***`id`***`/`

Formats

JSON, XML

HTTP Method

GET, PATCH, DELETE

Authentication

`Authorization: OAuth` ***`token`***

Parameters

Parameter	Description
`fields`	Optional list of fields used to return values for.

Retrieving Fields from an Account Object

You can use the GET method to retrieve individual records for an object. In this example, the Account Number and Billing Postal Code are retrieved from an Account.

Example usage for retrieving fields from an Account object

```
curl
http://na1.salesforce.com/services/data/v20.0/sobjects/Account/001D000000INjVe?fields=AccountNumber,BillingPostalCode
-H "Authorization: OAuth token" -H "X-PrettyPrint:1"
```

Example request body for retrieving fields from an Account object

none required

Example response body for retrieving fields from an Account object

```
{
    "AccountNumber" : "CD656092",
    "BillingPostalCode" : "27215",
}
```

Error responses

See Error Response on page 38.

Updating Fields in an Account Object

You can use the PATCH method to update individual records for an object. In this example, the Billing City within an Account is updated.

Example usage for updating fields in an Account object

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/001D000000INjVe
-H "Authorization: OAuth token" -H "X-PrettyPrint:1" -H "Content-Type: application/json"
-d @patchaccount.json -X PATCH
```

Example request body for updating fields in an Account object

```
{
    "BillingCity" : "San Francisco"
}
```

Example response body for updating fields in an Account object

none returned

Error responses

See Error Response on page 38.

Deleting an Account Object

You can use the DELETE method to delete an object. In this example, an Account object is deleted.

Example usage for deleting fields in an Account object

```
curl http://na1.salesforce.com/services/data/v20.0/sobjects/Account/001D000000INjVe
-H "Authorization: OAuth token" -H "X-PrettyPrint:1" -X DELETE
```

Example request body for updating fields in an Account object

none required

Example response body for updating fields in an Account object

none returned

SObject Blob Retrieve

Retrieves the specified blob field from an individual record.

URI

`/vXX.X/sobjects/SObjectName/id/blobField`

Formats

Because blob fields contain binary data, you can't use JSON or XML to retrieve this data.

HTTP Method

GET

Authentication

`Authorization: OAuth token`

Parameters

none required

Example usage

```
curl
http://na1.salesforce.com/services/data/v20.0/sobjects/attachment/001D000000INjVe/body
-H "Authorization: OAuth token" -H "X-PrettyPrint:1"
```

Example request body

none required

Example response body

Content is contained in the attachment.

Error responses

See Error Response on page 38.

Query

Executes the specified SOQL query.

URI

`/vXX.X/query/?q=soql`

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: OAuth token`

Parameters

Parameter	Description
`q`	A SOQL query.

Executing the SOQL Query

You can execute a SOQL query that can return all the results in a single response, or return part of the results and an identifier used to retrieve the remaining results.

Example query

```
curl http://na1.salesforce.com/services/data/v20.0/query/?q=SELECT+name+from+Account
-H "Authorization: OAuth token" -H "X-PrettyPrint:1"
```

Example request body for executing a query

none required

Example response body for executing a query

```
{
    "done" : true,
    "totalSize" : 14,
    "records" :
    [
        {
            "attributes" :
            {
                "type" : "Account",
                "url" : "/services/data/v20.0/sobjects/Account/001D000000IRFmaIAH"
            },
            "Name" : "Test 1"
        },
        {
            "attributes" :
            {
                "type" : "Account",
                "url" : "/services/data/v20.0/sobjects/Account/001D000000IomazIAB"
            },
            "Name" : "Test 2"
        },

        ...

    ]
}
```

Error responses

See Error Response on page 38.

Retrieving the Remaining SOQL Query Results

If the initial query returns only part of the results, the end of the response will contain a field called `nextRecordsUrl`. For example, you might find this attribute at the end of your query:

```
"nextRecordsUrl" : "/services/data/v20.0/query/01gD0000002HU6KIAW-2000"
```

In such cases, request the next batch of records and repeat until all records have been retrieved. These requests use `nextRecordsUrl`, and do not include any parameters.

Example usage for retrieving the remaining query results

```
curl http://na1.salesforce.com/services/data/v20.0/query/01gD0000002HU6KIAW-2000 -H
"Authorization: OAuth token" -H "X-PrettyPrint:1"
```

Example request body for retrieving the remaining query results

none required

Example response body for retrieving the remaining query results

```
{
    "done" : true,
    "totalSize" : 3214,
    "records" : [...]
}
```

Error responses

See Error Response on page 38.

Search

Executes the specified SOSL search. The search string must be URL-encoded.

URI

`/vXX.X/search/?q=`***`sosl`***

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: OAuth `***`token`***

Parameters

Parameter	Description
q	A SOSL statement.

Example usage

The search string, `{test}`, has been URL-encoded.`curl http://na1.salesforce.com/services/data/v20.0/search/?q=FIND+%7Btest%7D -H "Authorization: OAuth `***`token`***`" -H "X-PrettyPrint:1"`

Example request body

none required

Example response body

```
[
    {
        "attributes" :
         {
            "type" : "Account",
            "url" : "/services/data/v20.0/sobjects/Account/001D000000IqhSLIAZ"
        },
        "Id" : "001D000000IqhSLIAZ"
    },
    {
        "attributes" :
        {
            "type" : "Account",
            "url" : "/services/data/v20.0/sobjects/Account/001D000000IomazIAB"
        },
        "Id" : "001D000000IomazIAB"
    }
]
```

Error Response

When errors occur, the response you receive usually contains the error code, the message accompanying that error, and the field or object where the error occurred.

Incorrect ID example

Using a non-existent ID in a request using JSON or XML (`*request_body*.json` or `*request_body*.xml`)

```
{
  "fields" : [ "Id" ],
  "message" : "Account ID: id value of incorrect type: 001900K0001pPuOAAU",
  "errorCode" : "MALFORMED_ID"
}
```

Resource does not exist

Requesting a resource that does not exist, for example, you try to create a record using a misspelled object name

```
{
  "message" : "The requested resource does not exist",
  "errorCode" : "NOT_FOUND"
}
```

Index