Version 28.0: Summer '13

Force.com REST API Developer's Guide

Last updated: August 30, 2013

Table of Contents

Chapter 1

Introducing Force.com REST API

In this chapter ...

REST API provides a powerful, convenient, and simple Web services API for interacting with Force.com. Its advantages include ease of integration and development, and it's an excellent choice of technology for use with mobile applications and Web 2.0 projects. However, if you have a large number of records to process, you may wish to use Bulk API, which is based on REST principles and optimized for large sets of data.

REST API uses the same underlying data model and standard objects as those in SOAP API. See the SOAP API Developer's Guide for details. REST API also follows the same limits as SOAP API. See the Limits section in the SOAP API Developer's Guide.

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

Use this introduction to understand:

- The key characteristics and architecture of 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 REST API immediately.
- How to use REST API by following a quick start that leads you step by step through a typical use case.

Understanding Force.com REST Resources

A REST resource is an abstraction of a piece of information, such as a single data record, a collection of records, or even dynamic real-time information. Each resource in the Force.com REST API is identified by a named URI, and is accessed using standard HTTP methods (HEAD, GET, POST, PATCH, DELETE). The Force.com REST API is based on the usage of resources, their URIs, and the links between them. You use a resource to interact with your Salesforce or Force.com organization. For example, you can:

- Retrieve summary information about the API versions available to you.
- Obtain detailed information about a Salesforce object such as an Account or a custom object.
- Obtain detailed information about Force.com objects, such as User or a custom object.
- Perform a query or search.
- Update or delete records.

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

```
curl https://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 runs on multiple server instances. The examples in this guide use the *na1* instance. The instance your organization uses might 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 Force.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 (an open protocol to allow secure API authorization). See Understanding Authentication for more details.

Support for JSON and XML

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`).

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

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

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. Compression is automatically supported by some clients, and can be manually added to others. Visit Developer Force for more information on particular clients.

Tip: 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`

Using cURL in the REST Examples

The examples in this guide use the cURL tool to send HTTP requests to access, create, and manipulate REST resources on the Force.com platform. 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.

Note: cURL is an open source tool and is not supported by salesforce.com.

Escaping the Session ID or Using Single Quotes on Mac and Linux Systems

When running the cURL examples for the REST resources, you may get an error on Mac and Linux systems due to the presence of the exclamation mark special character in the session ID argument. To avoid getting this error, do one of the following:

- Escape the exclamation mark (!) special character in the session ID by inserting a backslash before it (\!) when the session ID is enclosed within double quotes. For example, the session ID string in this cURL command has the exclamation mark (!) escaped:

```
curl https://instance_name.salesforce.com/services/data/v28.0/
-H "Authorization: Bearer
00D50000000IehZ\!AQcAQH0dMHZfz972Szmpkb58urFRkgeBGsxL_QJWwYMfAbUeeG7c1E6
LYUfiDUkWe6H34r1AAwOR8B8fLEz6n04NPGRrq0FM"
```

- Enclose the session ID within single quotes. For example:

```
curl https://instance_name.salesforce.com/services/data/v28.0/
-H 'Authorization: Bearer sessionID'
```

Understanding Authentication

Salesforce uses authentication to allow users to securely access data without having to reveal username and password credentials.

Before making REST API calls, you must authenticate the user using OAuth 2.0. To do so, you'll need to:

- Set up a remote access application definition in Salesforce.
- Determine the correct OAuth endpoint to use.
- Authenticate the user via one of several different OAuth 2.0 authentication flows. An OAuth authentication flow defines a series of steps used to coordinate the authentication process between your application and Salesforce. Supported OAuth flows include:
  - ◊ Web server flow, where the server can securely protect the consumer secret.
  - ◊ User-agent flow, used by applications that cannot securely store the consumer secret.
  - ◊ Username-password flow, where the application has direct access to user credentials.

After successfully authenticating the user, you'll receive an access token which can be used to make authenticated REST API calls.

Defining Remote Access Applications

To authenticate using OAuth, you must define a remote access application in Salesforce.

A *remote access application* is an application external to Salesforce that uses the OAuth protocol to verify both the Salesforce user and the external application.When you develop a new external application that needs to authenticate with Salesforce, you need to define a new remote access application that informs Salesforce of this new authentication entry point.

Use the following steps to create a new remote access application.

1. From Setup, click **Develop** > **Remote Access** and click **New**.
2. Enter the name of your application.
3. Enter a `Callback URL`. Depending on which OAuth flow you use, this is typically the URL that a user's browser is redirected to after successful authentication. As this URL is used for some OAuth flows to pass an access token, the URL must use secure HTTP (HTTPS) or a custom URI scheme.
4. Enter a URL for `Info URL`. This is where the user can go for more information about your application.
5. Enter the contact email information, as well as any other information appropriate for your application.
6. Click **Save**. The `Consumer Key` is created and displayed, and the `Consumer Secret` is created (click the link to reveal it).

Once you define a remote access application, you use the consumer key and consumer secret to authenticate your application.

Understanding OAuth Endpoints

OAuth endpoints are the URLs you use to make OAuth authentication requests to Salesforce.

You need to use the correct Salesforce OAuth endpoint when issuing authentication requests in your application. The primary OAuth endpoints are:

- For authorization: `https://login.salesforce.com/services/oauth2/authorize`
- For token requests: `https://login.salesforce.com/services/oauth2/token`
- For revoking OAuth tokens: `https://login.salesforce.com/services/oauth2/revoke`

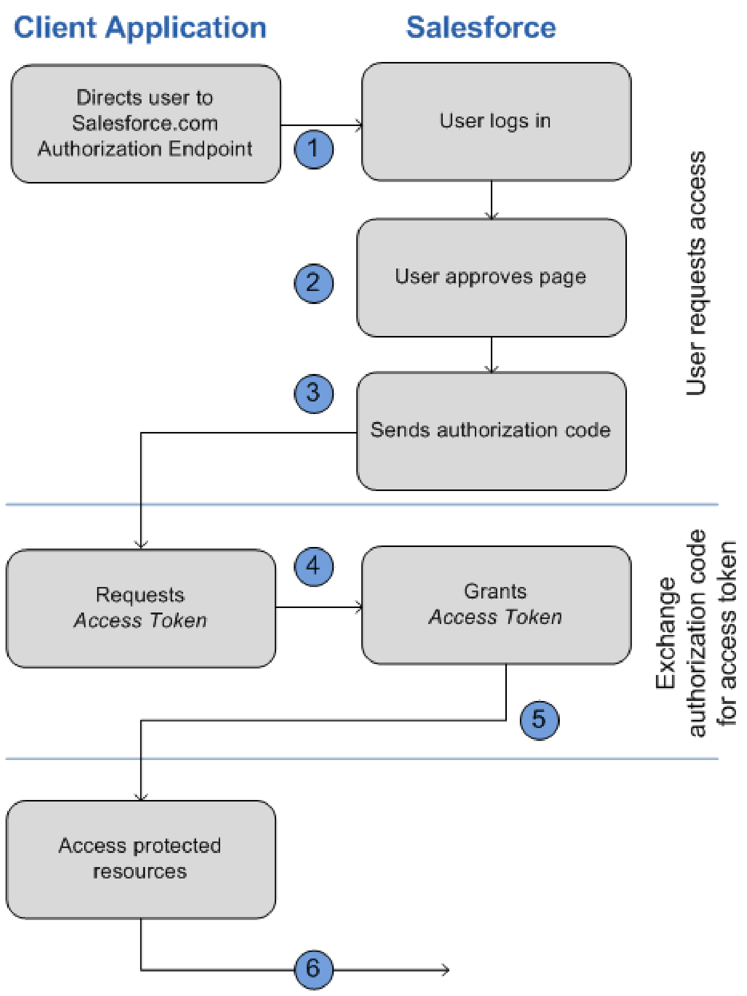
All endpoints require secure HTTP (HTTPS). Each OAuth flow defines which endpoints you need to use and what request data you need to provide.

If you're verifying authentication on a sandbox organization, use "test.salesforce.com" instead of "login.salesforce.com" in all the OAuth endpoints listed above.

Understanding the Web Server OAuth Authentication Flow

The Web server authentication flow is used by applications that are hosted on a secure server. A critical aspect of the Web server flow is that the server must be able to protect the consumer secret.

In this flow, the client application requests the authorization server to redirect the user to another web server or resource that authorizes the user and sends the application an authorization code. The application uses the authorization code to request an access token. The following shows the steps for this flow.



1. The application redirects the user to the appropriate Salesforce authorization endpoint, such as `https://login.salesforce.com/services/oauth2/authorize`. The following parameters are required:

Parameter	Description
`response_type`	Must be `code` for this authentication flow.
`client_id`	The `Consumer Key` from the remote access application definition.
`redirect_url`	The `Callback URL` from the remote access application definition.

The following parameters are optional:

Parameter	Description
`display`	Changes the login page's display type. Valid values are: • `page`—Full-page authorization screen. This is the default value if none is specified.

Parameter	Description
	• `popup`—Compact dialog optimized for modern Web browser popup windows. • `touch`—Mobile-optimized dialog designed for modern smartphones such as Android and iPhone. • `mobile`—Mobile optimized dialog designed for smartphones such as BlackBerry OS 5 that don't support touch screens.
`immediate`	Determines whether the user should be prompted for login and approval. Values are either `true` or `false`. Default is `false`. • If set to `true`, and if the user is currently logged in and has previously approved the application, the approval step is skipped. • If set to `true` and the user is not logged in or has not previously approved the application, the session is immediately terminated with the `immediate_unsuccessful` error code.
`state`	Specifies any additional URL-encoded state data to be returned in the callback URL after approval.
`scope`	Specifies what data your application can access. See "Scope Parameter Values" in the online help for more information.

An example authorization URL might look something like the following:

```
https://login.salesforce.com/services/oauth2/authorize?response_type=code&client_id=
3MVG9lKcPoNINVBIPJjdw1J9LLM82HnFVVX19KY1uA5mu0QqEWhqKpoW3svG3XHrXDiCQjK1mdgAvhCscA
9GE&redirect_uri=https%3A%2F%2Fwww.mysite.com%2Fcode_callback.jsp&state=mystate
```

2. The user logs into Salesforce with their credentials. The user is interacting with the authorization endpoint directly, so the application never sees the user's credentials. After successfully logging in, the user is asked to authorize the application. Note that if the user has already authorized the application, this step is skipped.
3. Once Salesforce confirms that the client application is authorized, the end-user's Web browser is redirected to the callback URL specified by the `redirect_uri` parameter. Salesforce appends authorization information to the redirect URL with the following values:

Parameters	Description
`code`	Authorization code the consumer must use to obtain the access and refresh tokens.
`state`	The state value that was passed in as part of the initial request, if applicable.

An example callback URL with authorization information might look something like:

```
https://www.mysite.com/authcode_callback?code=aWekysIEeqM9PiThEfm0Cnr6MoLIfwWyRJcqOqHdF
8f9INokharAS09ia7UNP6RiVScerfhc4w%3D%3D
```

4. The application extracts the authorization code and passes it in a request to Salesforce for an access token. This request is a POST request sent to the appropriate Salesforce token request endpoint, such as `https://login.salesforce.com/services/oauth2/token`. The following parameters are required:

Parameter	Description
`grant_type`	Value must be `authorization_code` for this flow.
`client_id`	The `Consumer Key` from the remote access application definition.
`client_secret`	The `Consumer Secret` from the remote access application definition.
`redirect_uri`	The `Callback URL` from the remote access application definition.
`code`	Authorization code the consumer must use to obtain the access and refresh tokens.

The following parameter is optional:

Parameter	Description
`format`	Expected return format. The default is `json`. Values are: • `urlencoded` • `json` • `xml` The return format can also be specified in the header of the request using one of the following: • `Accept: application/x-www-form-urlencoded` • `Accept: application/json` • `Accept: application/xml`

An example access token POST request might look something like:

```
POST /services/oauth2/token HTTP/1.1
Host: login.salesforce.com
grant_type=authorization_code&code=aPrxsmIEeqM9PiQroGEWx1UiMQd95_5JUZ
VEhsOFhS8EVvbfYBBJli2W5fn3zbo.8hojaNW_1g%3D%3D&client_id=3MVG9lKcPoNI
NVBIPJjdw1J9LLM82HnFVVX19KY1uA5mu0QqEWhqKpoW3svG3XHrXDiCQjK1mdgAvhCs
cA9GE&client_secret=1955279925675241571&
redirect_uri=https%3A%2F%2Fwww.mysite.com%2Fcode_callback.jsp
```

5. If this request is successful, the server returns a response body that contains the following:

Parameters	Description
`access_token`	Access token that acts as a session ID that the application uses for making requests. This token should be protected as though it were user credentials.
`refresh_token`	Token that can be used in the future to obtain new access tokens. **Warning:** This value is a secret. You should treat it like the user's password and use appropriate measures to protect it.
`instance_url`	Identifies the Salesforce instance to which API calls should be sent.
`id`	Identity URL that can be used to both identify the user as well as query for more information about the user. Can be used in an HTTP request to get more information about the end user.
`issued_at`	When the signature was created, represented as the number of seconds since the Unix epoch (00:00:00 UTC on 1 January 1970).
`signature`	Base64-encoded HMAC-SHA256 signature signed with the consumer's private key containing the concatenated ID and `issued_at` value. The `signature` can be used to verify that the identity URL wasn't modified because it was sent by the server.

An example JSON response body might look something like:

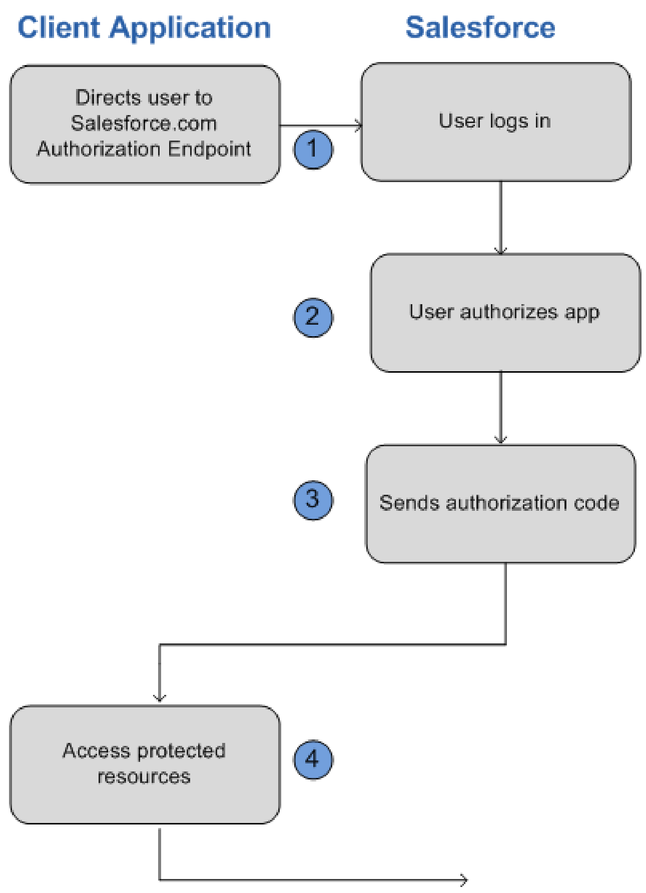
```
{"id":"https://login.salesforce.com/id/00Dx0000000BV7z/005x00000012Q9P",
"issued_at":"1278448101416","refresh_token":"5Aep8614iLM.Dq661ePDmPEgaAW9
Oh_L3JKkDpB4xReb54_pZebnUG0h6Sb4KUVDpNtWEofWM39yg==","instance_url":
"https://na1.salesforce.com","signature":"CMJ4l+CCaPQiKjoOEwEig9H4wqhpuLSk
4J2urAe+fVg=","access_token":"00Dx0000000BV7z!AR8AQP0jITN80ESEsj5EbaZTFG0R
NBaT1cyWk7TrqoDjoNIWQ2ME_sTZzBjfmOE6zMHq6y8PIW4eWze9JksNEkWUl.Cju7m4"}
```

6. The application uses the provided access token and refresh token to access protected user data.

Understanding the User-Agent OAuth Authentication Flow

The user-agent authentication flow is used by client applications (consumers) residing in the user's device. This could be implemented in a browser using a scripting language such as JavaScript, or from a mobile device or a desktop application. These consumers cannot keep the client secret confidential.

In this flow, the client application requests the authorization server to redirect the user to another Web server or resource which is capable of extracting the access token and passing it back to the application. The following shows the steps for this flow.



1. The application redirects the user to the appropriate Salesforce authorization endpoint, such as `https://login.salesforce.com/services/oauth2/authorize`. The following parameters are required:

Parameter	Description
`response_type`	Must be `token` for this authentication flow
`client_id`	The `Consumer Key` from the remote access application definition.
`redirect_url`	The `Callback URL` from the remote access application definition.

The following parameters are optional:

Parameter	Description
`display`	Changes the login page's display type. Valid values are: • `page`—Full-page authorization screen. This is the default value if none is specified. • `popup`—Compact dialog optimized for modern Web browser popup windows.

Parameter	Description
	• `touch`—Mobile-optimized dialog designed for modern smartphones such as Android and iPhone. • `mobile`—Mobile optimized dialog designed for smartphones such as BlackBerry OS 5 that don't support touch screens.
`scope`	Specifies what data your application can access. See "Scope Parameter Values" in the online help for more information.
`state`	Specifies any additional URL-encoded state data to be returned in the callback URL after approval.

An example authorization URL might look something like the following:

```
https://login.salesforce.com/services/oauth2/authorize?response_type=token&
client_id=3MVG9lKcPoNINVBIPJjdw1J9LLJbP_pqwoJYyuisjQhr_LLurNDv7AgQvDTZwCoZuD
ZrXcPCmBv4o.8ds.5iE&redirect_uri=https%3A%2F%2Fwww.mysite.com%2Fuser_callback.jsp&
state=mystate
```

2. The user logs into Salesforce with their credentials. The user interacts with the authorization endpoint directly, so the application never sees the user's credentials.
3. Once authorization is granted, the authorization endpoint redirects the user to the redirect URL. This URL is defined in the remote access application created for the application. Salesforce appends access token information to the redirect URL with the following values:

Parameters	Description
`access_token`	Access token that acts as a session ID that the application uses for making requests. This token should be protected as though it were user credentials.
`expires_in`	Amount of time the access token is valid, in seconds.
`refresh_token`	Token that can be used in the future to obtain new access tokens. **Warning:** This value is a secret. You should treat it like the user's password and use appropriate measures to protect it. The refresh token is only returned if the redirect URI is `https://login.salesforce.com/services/oauth2/success` or used with a custom protocol that is not HTTPS.
`state`	The state value that was passed in as part of the initial request, if applicable.
`instance_url`	Identifies the Salesforce instance to which API calls should be sent.

Parameters	Description
`id`	Identity URL that can be used to both identify the user as well as query for more information about the user. Can be used in an HTTP request to get more information about the end user.
`issued_at`	When the signature was created, represented as the number of seconds since the Unix epoch (00:00:00 UTC on 1 January 1970).
`signature`	Base64-encoded HMAC-SHA256 signature signed with the consumer's private key containing the concatenated ID and `issued_at` value. The `signature` can be used to verify that the identity URL wasn't modified because it was sent by the server.

An example callback URL with access information appended after the hash sign (#) might look something like:

```
https://www.mysite.com/user_callback.jsp#access_token=00Dx0000000BV7z%21AR8
AQBM8J_xr9kLqmZIRyQxZgLcM4HVi41aGtW0qW3JCzf5xdTGGGSoVim8FfJkZEqxbjaFbberKGk
8v8AnYrvChG4qJbQo8&refresh_token=5Aep8614iLM.Dq661ePDmPEgaAW9Oh_L3JKkDpB4xR
eb54_pZfVti1dPEk8aimw4Hr9ne7VXXVSIQ%3D%3D&expires_in=7200&state=mystate
```

4. The application uses the provided access token and refresh token to access protected user data.

Keep the following considerations in mind when using the user-agent OAuth flow:

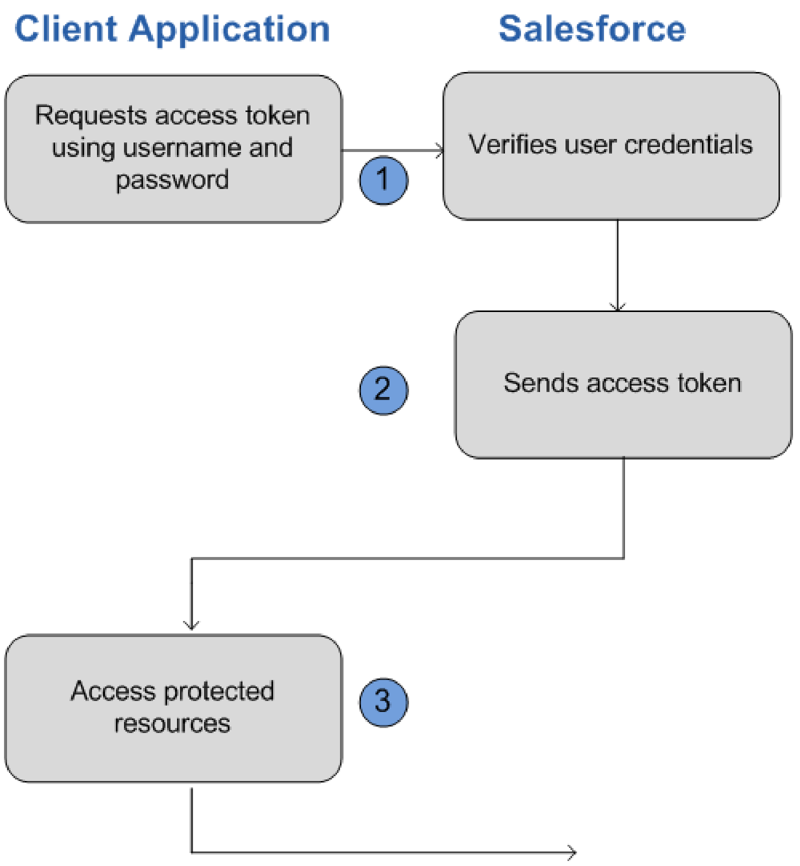
- Because the access token is encoded into the redirection URI, it might be exposed to the end-user and other applications residing on the computer or device. If you're authenticating using JavaScript, call `window.location.replace();` to remove the callback from the browser's history.

Understanding the Username-Password OAuth Authentication Flow

The username-password authentication flow can be used to authenticate when the consumer already has the user's credentials.

In this flow, the user's credentials are used by the application to request an access token as shown in the following steps.

Warning: This OAuth authentication flow involves passing the user's credentials back and forth. Use this authentication flow only when necessary. No refresh token will be issued.


Client Application
Salesforce
Requests access token using username and password
1
Verifies user credentials
2
Sends access token
Access protected resources
3

1. The application uses the user's username and password to request an access token. This is done via an out-of-band POST request to the appropriate Salesforce token request endpoint, such as `https://login.salesforce.com/services/oauth2/token`. The following request fields are required:

Parameter	Description
`grant_type`	Must be `password` for this authentication flow.
`client_id`	The `Consumer Key` from the remote access application definition.
`client_secret`	The `Consumer Secret` from the remote access application definition.
`username`	End-user's username.
`password`	End-user's password. **Note:** You must append the user's security token to their password A security token is an automatically-generated key from Salesforce. For example, if a user's password is mypassword, and their security token is XXXXXXXXXX, then the value provided for this parmeter must be mypasswordXXXXXXXXXX. For more

Parameter	Description
	information on security tokens see “Resetting Your Security Token” in the online help.

An example request body might look something like the following:

```
grant_type=password&client_id=3MVG9lKcPoNINVBIPJjdw1J9LLM82Hn
FVVX19KY1uA5mu0QqEWhqKpoW3svG3XHrXDiCQjK1mdgAvhCscA9GE&client_secret=
1955279925675241571&username=testuser%40salesforce.com&password=mypassword123456
```

2. Salesforce verifies the user credentials, and if successful, sends a response to the application with the access token. This response contains the following values:

Parameters	Description
`access_token`	Access token that acts as a session ID that the application uses for making requests. This token should be protected as though it were user credentials.
`instance_url`	Identifies the Salesforce instance to which API calls should be sent.
`id`	Identity URL that can be used to both identify the user as well as query for more information about the user. Can be used in an HTTP request to get more information about the end user.
`issued_at`	When the signature was created, represented as the number of seconds since the Unix epoch (00:00:00 UTC on 1 January 1970).
`signature`	Base64-encoded HMAC-SHA256 signature signed with the consumer's private key containing the concatenated ID and `issued_at` value. The `signature` can be used to verify that the identity URL wasn’t modified because it was sent by the server.

An example response body might look something like:

```
{"id":"https://login.salesforce.com/id/00Dx0000000BV7z/005x00000012Q9P",
"issued_at":"1278448832702","instance_url":"https://na1.salesforce.com",
"signature":"0CmxinZir53Yex7nE0TD+zMpvIWYGb/bdJh6XfOH6EQ=","access_token":
"00Dx0000000BV7z!AR8AQAxo9UfVkh8AlV0Gomt9Czx9LjHnSSpwBMmbRcgKFmxOtvxjTrKW1
9ye6PE3Ds1eQz3z8jr3W7_VbWmEu4Q8TVGSTHxs"}
```

3. The application uses the provided access token to access protected user data.

Keep the following considerations in mind when using the user-agent OAuth flow:

- Since the user is never redirected to login at Salesforce in this flow, the user can't directly authorize the application, so no refresh tokens can be used. If your application requires refresh tokens, you should consider using the Web server or user-agent OAuth flow.

Understanding the OAuth Refresh Token Process

The Web server OAuth authentication flow and user-agent flow both provide a refresh token that can be used to obtain a new access token.

Access tokens have a limited lifetime specified by the session timeout in Salesforce. If an application uses an expired access token, a "Session expired or invalid" error is returned. If the application is using the Web server or user-agent OAuth authentication flows, a refresh token may be provided during authorization that can be used to get a new access token.

The client application obtains a new access token by sending a POST request to the token request endpoint with the following request parameters:

Parameters	Description
`grant_type`	Value must be `refresh_token`.
`refresh_token`	The refresh token the client application already received.
`client_id`	The `Consumer Key` from the remote access application definition.
`client_secret`	The `Consumer Secret` from the remote access application definition. This parameter is optional.
`format`	Expected return format. The default is `json`. Values are: • `urlencoded` • `json` • `xml` The return format can also be specified in the header of the request using one of the following: • `Accept: application/x-www-form-urlencoded` • `Accept: application/json` • `Accept: application/xml` This parameter is optional.

An example refresh token POST request might look something like:

```
POST /services/oauth2/token HTTP/1.1
Host: https://login.salesforce.com/
grant_type=refresh_token&client_id=3MVG9lKcPoNINVBIPJjdw1J9LLM82HnFVVX19KY1uA5mu0
QqEWhqKpoW3svG3XHrXDiCQjK1mdgAvhCscA9GE&client_secret=1955279925675241571
&refresh_token=your token here
```

Once Salesforce verifies the refresh token request, it sends a response to the application with the following response body parameters:

Parameters	Description
`access_token`	Access token that acts as a session ID that the application uses for making requests. This token should be protected as though it were user credentials.
`instance_url`	Identifies the Salesforce instance to which API calls should be sent.
`id`	Identity URL that can be used to both identify the user as well as query for more information about the user. Can be used in an HTTP request to get more information about the end user.
`issued_at`	When the signature was created, represented as the number of seconds since the Unix epoch (00:00:00 UTC on 1 January 1970).
`signature`	Base64-encoded HMAC-SHA256 signature signed with the consumer's private key containing the concatenated ID and `issued_at` value. The `signature` can be used to verify that the identity URL wasn't modified because it was sent by the server.

An example JSON response body might look something like:

```
{ "id":"https://login.salesforce.com/id/00Dx0000000BV7z/005x00000012Q9P",
"issued_at":"1278448384422","instance_url":"https://na1.salesforce.com",
"signature":"SSSbLO/gBhmmyNUvN18ODBDFYHzakxOMgqYtu+hDPsc=",
"access_token":"00Dx0000000BV7z!AR8AQP0jITN80ESEsj5EbaZTFG0RNBaT1cyWk7T
rqoDjoNIWQ2ME_sTZzBjfmOE6zMHq6y8PIW4eWze9JksNEkWUl.Cju7m4"}
```

Keep in mind the following considerations when using the refresh token OAuth process:

- The session timeout for an access token can be configured in Salesforce from Setup by clicking **Security Controls** > **Session Settings**.
- If the application uses the username-password OAuth authentication flow, no refresh token is issued, as the user cannot authorize the application in this flow. If the access token expires, the application using username-password OAuth flow must re-authenticate the user.

Finding Additional Resources

The following resources provide additional information about using OAuth with Salesforce:

- Digging Deeper into OAuth on Force.com
- Using OAuth to Authorize External Applications

The following resources are examples of third party client libraries that implement OAuth that you might find useful:

- For Ruby on Rails: OmniAuth
- For Java: Apache Amber
- Additional OAuth client libraries: OAuth.net

Chapter 2
Quick Start

In this chapter ...

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.
- Enable an SSL endpoint in your application server.
- 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 Developer Edition Organization

If you are not already a member of the Force.com developer community, go to `http://developer.force.com/join` and follow the instructions for signing up for a Developer Edition organization. Even if you already have Enterprise Edition or Unlimited Edition, 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 insert, update, or delete data (as opposed to simply reading data).

If you already have a Developer Edition organization, verify that you have the "API Enabled" permission. This permission is enabled by default, but may have been changed by an administrator. For more information, see the help in the Salesforce 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.

Partners, who wish to get an OAuth consumer Id for authentication, can contact salesforce.com

Setting Up OAuth 2.0

Setting up OAuth 2.0 requires that you take some steps within Salesforce and in other locations. If any of the steps are unfamiliar, see Understanding Authentication or the Salesforce online help. The following example uses the Web server OAuth flow.

1. In Salesforce, from Setup, click **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://`. For development

environments, the callback URL is similar to `https://localhost:8443/RestTest/oauth/_callback`. When you click Save, the `Consumer Key` is created and displayed, and a `Consumer Secret` is created (click the link to reveal it).

Note: The OAuth 2.0 specification uses “client” instead of “consumer.” Salesforce supports OAuth 2.0.

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`.

In your client application, redirect the user to the appropriate Salesforce authorization endpoint. On successful user login, Salesforce will call your redirect URI with an authorization code. You use the authorization code in the next step to get the access token.

2. From your Java or other client application, make a request to the appropriate Salesforce token request endpoint that passes in `grant_type`, `client_id`, `client_secret`, and `redirect_uri`. The `redirect_uri` is the URI that Salesforce sends a callback to.

```
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 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 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", "Bearer "+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: Bearer access_token
```

For example:

```
curl https://instance_name.salesforce.com/services/data/v20.0/ -H 'Authorization: Bearer
access_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 API `login()` call returns the session ID. You may also have the session ID, for example as part of the Apex current context. If you need a session ID just for testing purposes during development, you can use the username-password OAuth flow in a cURL command similar to the following:

```
curl https://login.salesforce.com/services/oauth2/token -d "grant_type=password" -d
"client_id=myclientid" -d "client_secret=myclientsecret" -d "mylogin@salesforce.com"
    -d "password=mypassword123456"
```

You will need to provide your client id, client secret, username and password with user security token appended.

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: Bearer access_token
   ```

   For example:

   ```
   curl https://instance_name.salesforce.com/services/data/v20.0/ -H 'Authorization: Bearer
    access_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 JSON 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 17.
- 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 https://na1.salesforce.com/services/data/v20.0/sobjects/Account/ -H "Authorization:
Bearer token -H "Content-Type: application/json" -d @newaccount.json"
```

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: Bearer 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 53.

See Also:

Using cURL in the REST Examples

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 is `https://na1.salesforce.com/services/data`. For more information, see Understanding Force.com REST Resources on page 2.

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

1. Get the Salesforce version.
2. Use the Salesforce 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 Version

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

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

The output from this request, including the response header:

```
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, in this example for version 20.0. To do this, submit a request for the Resources by Version:

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

The output from this request is as follows:

```
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 https://na1.salesforce.com/services/data/v20.0/sobjects/Account/ -H "Authorization:
Bearer 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 https://na1.salesforce.com/services/data/v20.0/sobjects/Account/describe/ -H
"Authorization: Bearer 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.salesforce.com/{ID}/e",
        "sobject" : "/services/data/v20.0/sobjects/Account",
        "uiDetailTemplate" : "https://na1.soma.salesforce.com/{ID}",
        "describe" : "/services/data/v20.0/sobjects/Account/describe",
        "rowTemplate" : "/services/data/v20.0/sobjects/Account/{ID}",
        "uiNewRecord" : "https://na1.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 https://na1.salesforce.com/services/data/v20.0/query?q=SELECT+name+from+Account -H
"Authorization: Bearer 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 *Salesforce SOQL and SOSL Reference 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 Rows 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 https://na1.salesforce.com/services/data/v20.0/sobjects/Account/001D000000IroHJ -H
"Authorization: Bearer access_token" -H "X-PrettyPrint:1" -H "Content-Type: application/json"
 --data-binary @patchaccount.json -X PATCH
```

No response body is returned, just the headers:

```
HTTP/1.1 204 No Content
Server:
Content-Length: 0
```

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

Other Resources

- Search for Ruby on *developer.force.com*
- Force.com Cookbook recipe for getting started in Ruby
- Force.com REST API Board

Chapter 3

Using REST API Resources

In this chapter ...

This section provides examples of using REST API resources to do a variety of different tasks, including working with objects, organization information, and queries.

For complete reference information on REST API resources, see Reference on page 53.

Getting Information About My Organization

You can use REST API to get information about your organization.

The examples in this section use REST API resources to retrieve organization-level information, such as a list of all objects available in your organization.

List Available REST API Versions

Use the Versions resource to list 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 JSON response body

```
[
    {
        "version" : "20.0",
        "label" : "Winter '11",
        "url" : "/services/data/v20.0"
    },
    {
        "version" : "21.0",
        "label" : "Spring '11",
        "url" : "/services/data/v21.0"
    },
    ...
    {
        "version" : "26.0",
        "label" : "Winter '13",
        "url" : "/services/data/v26.0"
    }
]
```

Example XML response body

```
<?xml version="1.0" encoding="UTF-8"?>
<Versions>
    <Version>
        <label>Winter &apos;11</label>
        <url>/services/data/v20.0</url>
        <version>20.0</version>
    </Version>
    <Version>
        <label>Spring &apos;11</label>
        <url>/services/data/v21.0</url>
        <version>21.0</version>
    </Version>
    ...
    <Version>
        <label>Winter &apos;13</label>
        <url>/services/data/v26.0</url>
        <version>26.0</version>
    </Version>
</Versions>
```

List Available REST Resources

Use the Resources by Version resource to list the resources available for the specified API version. This provides the name and URI of each additional resource.

Example

```
curl https://na1.salesforce.com/services/data/v26.0/ -H "Authorization: Bearer token"
```

Example request body

none required

Example JSON response body

```
{
    "sobjects" : "/services/data/v26.0/sobjects",
    "licensing" : "/services/data/v26.0/licensing",
    "connect" : "/services/data/v26.0/connect",
    "search" : "/services/data/v26.0/search",
    "query" : "/services/data/v26.0/query",
    "tooling" : "/services/data/v26.0/tooling",
    "chatter" : "/services/data/v26.0/chatter",
    "recent" : "/services/data/v26.0/recent"
}
```

Example XML response body

```
<?xml version="1.0" encoding="UTF-8"?>
<urls>
    <sobjects>/services/data/v26.0/sobjects</sobjects>
    <licensing>/services/data/v26.0/licensing</licensing>
    <connect>/services/data/v26.0/connect</connect>
    <search>/services/data/v26.0/search</search>
    <query>/services/data/v26.0/query</query>
    <tooling>/services/data/v26.0/tooling</tooling>
    <chatter>/services/data/v26.0/chatter</chatter>
    <recent>/services/data/v26.0/recent</recent>
</urls>
```

Get a List of Objects

Use the Describe Global resource to list the objects available in your organization and available to the logged-in user. This resource also returns the organization encoding, as well as maximum batch size permitted in queries.

Example usage

```
curl https://na1.salesforce.com/services/data/v26.0/sobjects/ -H "Authorization: Bearer
 token"
```

Example request body

none required

Example response body

```
{
  "encoding" : "UTF-8",
  "maxBatchSize" : 200,
  "sobjects" : [ {
    "name" : "Account",
    "label" : "Account",
    "keyPrefix" : "001",
    "labelPlural" : "Accounts",
    "custom" : false,
    "layoutable" : true,
    "activateable" : false,
    "urls" : {
      "sobject" : "/services/data/v26.0/sobjects/Account",
      "describe" : "/services/data/v26.0/sobjects/Account/describe",
      "rowTemplate" : "/services/data/v26.0/sobjects/Account/{ID}"
    },
    "searchable" : true,
    "updateable" : true,
    "createable" : true,
    "deprecatedAndHidden" : false,
    "customSetting" : false,
    "deletable" : true,
    "feedEnabled" : true,
    "mergeable" : true,
    "queryable" : true,
    "replicateable" : true,
    "retrieveable" : true,
    "undeletable" : true,
    "triggerable" : true
  },
  ...
  ]
}
```

Working with Object Metadata

You can use REST API to get basic object metadata information.

The examples in this section use REST API resources to retrieve object metadata information. For modifying or creating object metadata information, see the *Metadata API Developer's Guide*.

Retrieve Metadata for an Object

Use the SObject Basic Information resource to retrieve metadata for an object.

Example for retrieving Account metadata

```
curl https://na1.salesforce.com/services/data/v20.0/sobjects/Account/ -H "Authorization:
 Bearer token"
```

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.

Get Field and Other Metadata for an Object

Use the SObject Describe resource to retrieve all the metadata for an object, including information about each field, URLs, and child relationships.

Example

```
https://na1.salesforce.com/services/data/v20.0/Account/describe/ -H "Authorization:
Bearer token"
```

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,
    ...
}
```

Working with Records

You can use REST API to work with records in your organization.

The examples in this section use REST API resources to create, retrieve, update, and delete records, along with other record-related operations.

Create a Record

Use the SObject Basic Information resource to create new records. You supply the required field values in the request data, and then use the POST method of the resource. The response body will contain the ID of the created record if the call is successful.

The following example creates a new Account record, with the field values provided in `newaccount.json`.

Example for creating a new Account

```
curl https://na1.salesforce.com/services/data/v20.0/sobjects/Account/ -H "Authorization:
 Bearer token -H "Content-Type: application/json" -d @newaccount.json"
```

Example request body `newaccount.json` file for creating a new Account

```
{
    "Name" : "Express Logistics and Transport"
}
```

Example response body after successfully creating a new Account

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

Update a Record

You use the SObject Rows resource to update records. Provide the updated record information in your request data and use the PATCH method of the resource with a specific record ID to update that record. Records in a single file must be of the same object type.

In the following example, the Billing City within an Account is updated. The updated record information is provided in `patchaccount.json`.

Example for updating an Account object

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

Example request body `patchaccount.json` file 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 Status Codes and Error Responses on page 69.

The following example uses Java and HttpClient to update a record using REST API. Note that there is no PatchMethod in HttpClient, so PostMethod is overridden to return “PATCH” as its method name. This example assumes the resource URL has been passed in and contains the object name and record ID.

```
 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");
  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;
 }
```

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");
```

Delete a Record

Use the SObject Rows resource to delete records. Specify the record ID and use the DELETE method of the resource to delete a record.

Example for deleting an Account record

```
curl https://na1.salesforce.com/services/data/v20.0/sobjects/Account/001D000000INjVe
-H "Authorization: Bearer token" -X DELETE
```

Example request body

None needed

Example response body

None returned

Get Field Values from Records

You use the SObject Rows resource to retrieve field values from a record. Specify the fields you want to retrieve In the `fields` parameter and use the GET method of the resource. In the following example, the Account Number and Billing Postal Code are retrieved from an Account.

Example for retrieving values from fields on an Account object

```
curl https://na1.salesforce.com/services/data/v20.0/sobjects/Account/001D000000INjVe
?fields=AccountNumber,BillingPostalCode -H "Authorization: Bearer token"
```

Example request body

None required

Example response body

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

Retrieve a Record Using an External ID

You can use the GET method of the SObject Rows by External ID resource to retrieve records with a specific external ID.

The following example assumes there is a Merchandise__c custom object with a MerchandiseExtID__c external ID field.

Example usage for retrieving a Merchandise__c record using an external ID

```
curl
https://na1.salesforce.com/services/data/v20.0/sobjects/Merchandise__c/MerchandiseExtID__c/123
 -H "Authorization: Bearer token"
```

Example request body

none required

Example response body

```
{
    "attributes" : {
        "type" : "Merchandise__c",
        "url" : "/services/data/v20.0/sobjects/Merchandise__c/a00D0000008oWP8IAM"
     },
    "Id" : "a00D0000008oWP8IAM",
    "OwnerId" : "005D0000001KyEIIA0",
    "IsDeleted" : false,
    "Name" : "Example Merchandise",
    "CreatedDate" : "2012-07-12T17:49:01.000+0000",
    "CreatedById" : "005D0000001KyEIIA0",
    "LastModifiedDate" : "2012-07-12T17:49:01.000+0000",
    "LastModifiedById" : "005D0000001KyEIIA0",
    "SystemModstamp" : "2012-07-12T17:49:01.000+0000",
    "Description__c" : "Merch with external ID",
    "Price__c" : 10.0,
    "Total_Inventory__c" : 100.0,
    "Distributor__c" : null,
    "MerchandiseExtID__c" : 123.0
}
```

Insert or Update (Upsert) a Record Using an External ID

You can use the SObject Rows by External ID resource to create new records or update existing records (upsert) based on the value of a specified external ID field.

- If the specified value doesn't exist, a new record is created.
- If a record does exist with that value, the field values specified in the request body are updated.
- If the value is not unique, REST API returns a 300 response with the list of matching records.

The following sections show you how to work with the external ID resource to retrieve records by external ID and upsert records.

Upserting New Records

This example uses the PATCH method to insert a new record. It assumes that an external ID field, “customExtIdField__c,” has been added to Account. It also assumes that an Account record with a customExtIdField value of 11999 does not already exist.

Example for upserting a record that does not yet exist

```
curl
https://na1.salesforce.com/services/data/v20.0/sobjects/Account/customExtIdField__c/11999
 -H "Authorization: Bearer token" -H "Content-Type: application/json" -d @newrecord.json
 -X PATCH
```

Example JSON request body `newrecord.json` file

```
{
    "Name" : "California Wheat Corporation",
    "Type" : "New Customer"

}
```

Response

Successful response:

```
{
    "id" : "00190000001pPvHAAU",
    "errors" : [ ],
    "success" : true
}
```

The response body is empty. HTTP status code 201 is returned if a new record is created.

Error responses

Incorrect external ID field:

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

For more information, see Status Codes and Error Responses on page 69.

Upserting Existing Records

This example uses the PATCH method to update an existing record. It assumes that an external ID field, "customExtIdField__c," has been added to Account and an Account record with a customExtIdField value of 11999 exists. The request uses `updates.json` to specify the updated field values.

Example for upserting a record that already exists

```
curl
https://na1.salesforce.com/services/data/v20.0/sobjects/Account/customExtIdField__c/11999
 -H "Authorization: Bearer token" -H "Content-Type: application/json" -d @updates.json
 -X PATCH
```

Example JSON request body `updates.json` file

```
{

    "BillingCity" : "San Francisco"

}
```

JSON example response

HTTP status code 204 is returned if an existing record is updated.

Error responses

If the external ID value isn't unique, an HTTP status code 300 is returned, plus a list of the records that matched the query. For more information about errors, see Status Codes and Error Responses on page 69.

If the external ID field doesn't exist, an error message and code is returned:

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

Upserting Records and Associating with an External ID

If you have an object that references another object using a relationship, you can use REST API to both insert or update a new record, and also reference another object using an external ID.

The following example creates a new record and associates it with a parent record via external ID. It assumes the following:

- A Merchandise__c custom object that has an external ID field named MerchandiseExtID__c.
- A Line_Item__c custom object that has an external ID field named LineItemExtID__c, and a relationship to Merchandise__c.
- A Merchandise__c record exists that has a MerchandiseExtID__c value of 123.
- A Line_Item__c record with a LineItemExtID__c value of 456 does **not** exist. This is the record that will get created and associated to the Merchandise__c record.

Example for upserting a new record and referencing a related object

```
curl
https://na1.salesforce.com/services/data/v25.0/sobjects/Line_Item__c/LineItemExtID__c/456
 -H "Authorization: Bearer token" -H "Content-Type: application/json" -d @new.json -X
PATCH
```

Example JSON request body `new.json` file

Notice that the related Merchandise__c record is referenced using the Merchandise__c's external ID field.

```
{
   "Name" : "LineItemCreatedViaExtID",
   "Merchandise__r" :
   {
       "MerchandiseExtID__c" : 123
   }
}
```

JSON example response

HTTP status code 201 is returned on successful create.

```
{
    "id" : "a02D0000006YUHrIAO",
    "errors" : [ ],
    "success" : true
}
```

Error responses

If the external ID value isn't unique, an HTTP status code 300 is returned, plus a list of the records that matched the query. For more information about errors, see Status Codes and Error Responses on page 69.

If the external ID field doesn't exist, an error message and code is returned:

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

You can also use this approach to update existing records. For example, if you created the Line_Item__c shown in the example above, you can try to update the related Merchandise__c using another request.

Example for updating a record

```
curl
https://na1.salesforce.com/services/data/v25.0/sobjects/Line_Item__c/LineItemExtID__c/456
 -H "Authorization: Bearer token" -H "Content-Type: application/json" -d @updates.json
 -X PATCH
```

Example JSON request body `updates.json` file

This assumes another Merchandise__c record exists with a MerchandiseExtID__c value of 333.

```
{
   "Merchandise__r" :
   {
       "MerchandiseExtID__c" : 333
   }
}
```

JSON example response

HTTP status code 204 is returned if an existing record is updated.

Note that if your relationship type is master-detail and the relationship is set to not allow reparenting, and you try to update the parent external ID, you will get an HTTP status code 400 error with an error code of INVALID_FIELD_FOR_INSERT_UPDATE.

Get Attachment Content from a Record

Use the SObject Blob Retrieve resource to retrieve blob data for a given record.

The following example retrieves the blob data for an Attachment record. The Attachment can be associated with a Case, Campaign, or other object that allows attachments.

Example for retrieving blob body for an Attachment record

```
curl
https://na1.salesforce.com/services/data/v20.0/sobjects/Attachment/001D000000INjVe/body
 -H "Authorization: Bearer token"
```

Example request body

none required

Example response body

Attachment body content is returned in binary form. Note that the response content type will not be JSON or XML since the returned data is binary.

The following example retrieves the blob data for a Document record.

Example for retrieving blob body for a Document record

```
curl
https://na1.salesforce.com/services/data/v20.0/sobjects/Document/015D0000000NdJOIA0/body
 -H "Authorization: Bearer token"
```

Example request body

none required

Example response body

Document body content is returned in binary form. Note that the response content type will not be JSON or XML since the returned data is binary.

Insert or Update Blob Data

Note: This REST feature is currently available through a pilot program. For information on enabling it for your organization, contact salesforce.com.

You can use SObject Basic Information and SObject Rows REST resources to insert or update blob data in Salesforce standard objects. You can upload files of any type with a size of up to 500 MB, and you must use a multipart message that conforms to the MIME multipart content-type standard. For more information, see the WC3 Standards. You can insert or update files on any standard object that contains a blob field.

Note: You can insert or update blob data using a non-multipart message, but if you do, you are limited to 50 MB of text data or 37.5 MB of base64–encoded data.

The first part of the request message body contains non–binary field data such as the Description or Name. The second part of the message contains the binary data of the file that you're uploading.

The following sections provide JSON examples of how to insert or update blob data using a multipart content-type.

- Inserting a New Document
- Updating a Document
- Inserting a ContentVersion
- Multipart Message Considerations

Inserting a New Document

This section contains the syntax and code for creating a new Document. In addition to the binary data of the file itself, this code also specifies other field data such as the Description, Keywords, Name, and so on.

Tip: After you add a new Document, you can view the results of your changes on the Documents tab.

Example for creating a new Document

```
curl https://na1.salesforce.com/services/data/v23.0/sobjects/Document/ -H "Authorization:
 Bearer token" -H "Content-Type: multipart/form-data; boundary=\"boundary_string\""
--data-binary @newdocument.json
```

Example request body for creating a new Document

This code is the contents of `newdocument.json`. Note that the binary data for the PDF content has been omitted for brevity and replaced with "Binary data goes here." An actual request will contain the full binary content.

```
--boundary_string
Content-Disposition: form-data; name="entity_document";
Content-Type: application/json

{
    "Description" : "Marketing brochure for Q1 2011",
    "Keywords" : "marketing,sales,update",
    "FolderId" : "005D0000001GiU7",
    "Name" : "Marketing Brochure Q1",
    "Type" : "pdf"
}

--boundary_string
Content-Type: application/pdf
Content-Disposition: form-data; name="Body"; filename="2011Q1MktgBrochure.pdf"

Binary data goes here.

--boundary_string--
```

Example response body for creating a new Document

On success, the ID of the new Document is returned.

```
{
    "id" : "015D0000000N3ZZIA0",
    "errors" : [ ],
    "success" : true
}
```

Example error response

```
{
    "fields" : [ "FolderId" ],
    "message" : "Folder ID: id value of incorrect type: 005D0000001GiU7",
    "errorCode" : "MALFORMED_ID"
}
```

Updating a Document

This section contains the syntax and code for updating an existing Document. In addition to the binary data of the file itself, this code also updates other field data such as the Name and Keywords.

Example usage for updating fields in a Document object

```
curl https://na1.salesforce.com/services/data/v23.0/Document/015D0000000N3ZZIA0 -H
"Authorization: Bearer token" -H "Content-Type: multipart/form-data;
boundary=\"boundary_string\"" --data-binary @UpdateDocument.json -X PATCH
```

Example request body for updating fields in a Document object

This code is the contents of the file `UpdateDocument.json`. Note that the binary data for the PDF content has been omitted for brevity and replaced with "Updated document binary goes here." An actual request will contain the full binary content.

```
--boundary_string
Content-Disposition: form-data; name="entity_content";
Content-Type: application/json

{
    "Name" : "Marketing Brochure Q1 - Sales",
    "Keywords" : "sales, marketing, first quarter"
}

--boundary_string
Content-Type: application/pdf
Content-Disposition: form-data; name="Body"; filename="2011Q1MktgBrochure.pdf"

Updated document binary data goes here.

--boundary_string--
```

Example response body for updating fields in a Document object

none returned

Error responses

See Status Codes and Error Responses on page 69.

Inserting a ContentVersion

This section contains the syntax and code for inserting a new ContentVersion. In addition to the binary data of the file itself, this code also updates other fields such as the ReasonForChange and PathOnClient. This message contains the ContentDocumentId because a ContentVersion is always associated with a ContentDocument.

Tip: The ContentVersion object doesn't support updates. Therefore, you cannot update a ContentVersion, you can only insert a new ContentVersion. You can see the results of your changes on the Content tab.

Example usage for inserting a ContentVersion

```
curl https://na1.salesforce.com/services/data/v23.0/sobjects/ContentVersion -H
"Authorization: Bearer token" -H "Content-Type: multipart/form-data;
boundary=\"boundary_string\"" --data-binary @NewContentVersion.json
```

Example request body for inserting a ContentVersion

This code is the contents of the file `NewContentVersion.json`. Note that the binary data for the PDF content has been omitted for brevity and replaced with "Binary data goes here." An actual request would will the full binary content.

```
--boundary_string
Content-Disposition: form-data; name="entity_content";
Content-Type: application/json

{
    "ContentDocumentId" : "069D00000000so2",
    "ReasonForChange" : "Marketing materials updated",
    "PathOnClient" : "Q1 Sales Brochure.pdf"
}

--boundary_string
Content-Type: application/octet-stream
Content-Disposition: form-data; name="VersionData"; filename="Q1 Sales Brochure.pdf"

Binary data goes here.

--boundary_string--
```

Example response body for inserting a ContentVersion

```
{
    "id" : "068D00000000pgOIAQ",
    "errors" : [ ],
    "success" : true
}
```

Error responses for inserting a ContentVersion

See Status Codes and Error Responses on page 69.

Multipart Message Considerations

Following are some considerations for the format of a multipart message when you insert or update blob data.

Boundary String

- Separates the various parts of a multipart message.
- Required in a multipart content-type.
- Can be up to 70 characters.
- Cannot be a string value that appears anywhere in any of the message parts.
- The first boundary string must be prefixed by two hyphens (--).
- The last boundary string must be postfixed by two hyphens (--).

Content-Disposition Header

- Required in each message part.
- Must be the value `form-data` and have a `name` attribute.
  - ◊ In the non-binary part of the message, the `name` attribute can be any value.
  - ◊ In the binary part of the message, the `name` attribute should contain the name of the object field that will contain the binary data. In the previous example of adding a new Document, the name of the binary field that contains the file is Body.

- The binary part of the message must have a `filename` attribute which represents the name of the local file.

Content-Type Header

- Required in each message part.
- The content types supported by the non-binary message part are `application/json` and `application/xml`.
- The `Content-Type` header for the binary part of the message can be any value.

New Line

There must be a new line between the message part header and the data of the part. As shown in the code examples, there must be a new line between the `Content-Type` and `Content-Disposition` headers and the JSON or XML. In the binary part, there must be a new line between the `Content-Type` and `Content-Disposition` headers and the binary data.

Working with Searches and Queries

You can use REST API to make complex searches and queries across your data.

The examples in this section use REST API resources to search and query records using Salesforce Object Search Language (SOSL) and Salesforce Object Query Language (SOQL). For more information on SOSL and SOQL see the *Force.com SOQL and SOSL Reference*.

Execute a SOQL Query

Use the Query resource to 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.

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

Example usage for executing a query

```
curl https://na1.salesforce.com/services/data/v20.0/query/?q=SELECT+name+from+Account
-H "Authorization: Bearer token"
```

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 https://na1.salesforce.com/services/data/v20.0/query/01gD0000002HU6KIAW-2000 -H
"Authorization: Bearer token"
```

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

Use the Search resource to execute a SOSL search.

The following example executes a SOSL search for `{test}`. The search string in this example must be URL-encoded.

Example usage

```
curl https://na1.salesforce.com/services/data/v20.0/search/?q=FIND+%7Btest%7D -H
"Authorization: Bearer token"
```

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"
    }
]
```

Get the Default Search Scope and Order

Use the Search Scope and Order resource to retrieve the default global search scope and order for the logged-in user, including any pinned objects in the user's search results page.

In the following example, the default global search scope of the logged-in user consists of the site, idea, case, opportunity, account, and user objects, in the order in which they are returned in the response body.

Example usage

```
curl https://na1.salesforce.com/services/data/v26.0/search/scopeOrder -H "Authorization:
 Bearer token"
```

Example request body

none required

Example response body

```
[
    {
        "type":"Site",
        "url":"/services/data/v26.0/sobjects/Site/describe"
    },
    {
        "type":"Idea",
        "url":"/services/data/v26.0/sobjects/Idea/describe"
    },
    {
        "type":"Case",
        "url":"/services/data/v26.0/sobjects/Case/describe"
    },
    {
        "type":"Opportunity",
        "url":"/services/data/v26.0/sobjects/Opportunity/describe"
    },
    {
        "type":"Account",
        "url":"/services/data/v26.0/sobjects/Account/describe"
    },
    {
        "type":"User",
        "url":"/services/data/v26.0/sobjects/User/describe"
    }
]
```

Get Search Result Layouts for Objects

Use the Search Result Layouts resource to retrieve the search result layout configuration for each object specified in the query string.

Example usage

```
curl
https://na1.salesforce.com/services/data/v28.0/searchlayout/?q=Account,Contact,Lead,Asset
 "Authorization: Bearer token"
```

Example request body

None required

Example response body

```
[ { "label" : "Search Results",
    "limitRows" : 25,
    "searchColumns" : [ { "field" : "Account.Name",
          "format" : null,
          "label" : "Account Name",
          "name" : "Name"
        },
        { "field" : "Account.Site",
          "format" : null,
          "label" : "Account Site",
          "name" : "Site"
        },
        { "field" : "Account.Phone",
          "format" : null,
          "label" : "Phone",
          "name" : "Phone"
        },
        { "field" : "User.Alias",
          "format" : null,
          "label" : "Account Owner Alias",
          "name" : "Owner.Alias"
        }
      ]
  },
  { "label" : "Search Results",
    "limitRows" : 25,
    "searchColumns" : [ { "field" : "Contact.Name",
          "format" : null,
          "label" : "Name",
          "name" : "Name"
        },
        { "field" : "Account.Name",
          "format" : null,
          "label" : "Account Name",
          "name" : "Account.Name"
        },
        { "field" : "Account.Site",
          "format" : null,
          "label" : "Account Site",
          "name" : "Account.Site"
        },
        { "field" : "Contact.Phone",
          "format" : null,
          "label" : "Phone",
          "name" : "Phone"
        },
        { "field" : "Contact.Email",
          "format" : null,
          "label" : "Email",
          "name" : "Email"
        },
        { "field" : "User.Alias",
          "format" : null,
          "label" : "Contact Owner Alias",
          "name" : "Owner.Alias"
        }
      ]
  },
  { "label" : "Search Results",
    "limitRows" : 25,
    "searchColumns" : [ { "field" : "Lead.Name",
          "format" : null,
          "label" : "Name",
          "name" : "Name"
        },
        { "field" : "Lead.Title",
          "format" : null,
          "label" : "Title",
          "name" : "Title"
        },
        { "field" : "Lead.Phone",
          "format" : null,
          "label" : "Phone",
          "name" : "Phone"
        },
        { "field" : "Lead.Company",
          "format" : null,
          "label" : "Company",
          "name" : "Company"
        },
        { "field" : "Lead.Email",
          "format" : null,
          "label" : "Email",
          "name" : "Email"
        },
        { "field" : "Lead.Status",
          "format" : null,
          "label" : "Lead Status",
          "name" : "toLabel(Status)"
        },
        { "field" : "Name.Alias",
          "format" : null,
          "label" : "Owner Alias",
          "name" : "Owner.Alias"
        }
      ]
  },
]
```

Managing User Passwords

You can use REST API to manage credentials for the users in your organization.

The examples in this section use REST API resources to manage user passwords, such as setting or resetting passwords.

Manage User Passwords

Use the SObject User Password resource to set, reset, or get information about a user password. Use the HTTP GET method to get password expiration status, the HTTP POST method to set the password, and the HTTP DELETE method to reset the password.

The associated session must have permission to access the given user password information. If the session does not have proper permissions, an HTTP error 403 response is returned from these methods.

These methods are available for both users and self-service users. For managing self-service user passwords, use `SelfServiceUser` instead of `User` in the REST API URL.

Here is an example of retrieving the current password expiration status for a user:

Example usage for getting current password expiration status

```
curl
https://na1.salesforce.com/services/data/v25.0/sobjects/User/005D0000001KyEIIA0/password
 -H "Authorization: Bearer token"
```

Example request body for getting current password expiration status

None required

JSON example response body for getting current password expiration status

```
{
    "isExpired" : false
}
```

XML example response body for getting current password expiration status

```
<Password>
    <isExpired>false</isExpired>
</Password>
```

Example error response if session has insufficient privileges

```
{
    "message" : "You do not have permission to view this record.",
    "errorCode" : "INSUFFICIENT_ACCESS"
}
```

Here is an example of changing the password for a given user:

Example usage for changing a user password

```
curl
https://na1.salesforce.com/services/data/v25.0/sobjects/User/005D0000001KyEIIA0/password
 -H "Authorization: Bearer token" —H "Content-Type: application/json" —d @newpwd.json
—X POST
```

Contents for file newpwd.json

```
{
    "NewPassword" : "myNewPassword1234"
}
```

Example response for changing a user password

No response body on successful password change, HTTP status code 204 returned.

Example error response if new password does not meet organization password requirements

```
{
    "message" : "Your password must have a mix of letters and numbers.",
    "errorCode" : "INVALID_NEW_PASSWORD"
}
```

And finally, here is an example of resetting a user password:

Example usage for resetting a user password

```
curl
https://na1.salesforce.com/services/data/v25.0/sobjects/User/005D0000001KyEIIA0/password
 -H "Authorization: Bearer token" —X DELETE
```

Example request body for resetting a user password

None required

JSON example response body for resetting a user password

```
{
    "NewPassword" : "2sv0xHAuM"
}
```

XML example response body for resetting a user password

```
<Result>
    <NewPassword>2sv0xHAuM</NewPassword>
</Result>
```

Chapter 4

Reference

The following table lists supported REST resources in the API 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 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 version currently available, including the version, label, and a link to each version's root.
Resources by Version	`/vXX.X/`	Lists available resources for the specified API version, including resource name and URI.
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. Can also be used to create a new record for a given object.
SObject Describe	`/vXX.X/sobjects/`***`SObject`***`/describe/`	Completely describes the individual metadata at all levels for the specified object.
SObject Rows	`/vXX.X/sobjects/`***`SObject`***`/`***`id`***`/`	Accesses records based on the specified object ID. Retrieves, updates, or deletes records. This resource can also be used to retrieve field values.
SObject Rows by External ID	`/vXX.X/sobjects/`***`SObject`***`/`***`fieldName`***`/`***`fieldValue`***	Creates new records or updates existing records (upserts records) based on the value of a specified external ID field.
SObject Layouts	`/vXX.X/sobjects/global/describe/layouts/` `/vXX.X/sobjects/`***`object`***`/describe/layouts/`	Returns a list of layouts and descriptions, including for publisher actions.
SObject Blob Retrieve	`/vXX.X/sobjects/`***`SObject`***`/`***`id`***`/`***`blobField`***	Retrieves the specified blob field from an individual record.

Resource Name	URI	Description
SObject Quick Actions	/vXX.X/sobjects/***object***/quickActions/ /vXX.X/sobjects/***object***/quickActions/***{action name}*** /vXX.X/sobjects/***object***/quickActions/***{action name}***/describe/ services/data/vXX.X/sobjects/***object***/quickActions/***{action name}***/defaultValues/ vXX.X/sobjects/***object***/quickActions/{action name}/defaultValues/{parent id}	Returns a list of publisher actions and details.
SObject User Password	/vXX.X/sobjects/User/***user id***/password /vXX.X/sobjects/SelfServiceUser/***self service user id***/password	Set, reset, or get information about a user password.
Query	/vXX.X/query/?q=***soql***	Executes the specified SOQL query.
Quick Actions	/vXX.X/quickActions/	Return a list of global publisher actions and their types, as well as custom fields and objects that appear in the Chatter feed.
Search	/vXX.X/search/?s=***sosl***	Executes the specified SOSL search. The search string must be URL-encoded.
Search Scope and Order	/vXX.X/search/scopeOrder	Returns an ordered list of objects in the default global search scope of a logged-in user. Global search keeps track of which objects the user interacts with and how often and arranges the search results accordingly. Objects used most frequently appear at the top of the list.
Search Result Layouts	/vXX.X/searchlayout/?q=***Comma delimited object list***	Returns search result layout information for the objects in the query string. For each object, this call returns the list of fields displayed on the search results page as columns, the number of rows displayed on the first page, and the label used on the search results page.
Recently Viewed Items	/vXX.X/recent	Gets the most recently accessed items that were viewed or referenced by the current user.

Versions

Lists summary information about each Salesforce 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

See List Available REST API Versions on page 29.

Resources by Version

Lists available resources for the specified API version, including resource name and URI.

URI

`/vXX.X/`

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: Bearer` ***`token`***

Parameters

none

Example

See List Available REST Resources. on page 30

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 on encoding, see Internationalization and Character Sets.

URI

`/vXX.X/sobjects/`

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: Bearer` ***`token`***

Parameters

none required

Example

See Get a List of Objects on page 30.

Error responses

See Status Codes and Error Responses on page 69.

SObject Basic Information

Describes the individual metadata for the specified object. Can also be used to create a new record for a given object. For example, this can be used to retrieve the metadata for the Account object using the GET method, or create a new Account object using the POST method.

URI

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

Formats

JSON, XML

HTTP Method

GET, POST

Authentication

`Authorization: Bearer` ***`token`***

Parameters

none required

Examples

- For an example of retrieving metadata for an object, see Retrieve Metadata for an Object on page 32.
- For an example of creating a new record using POST, see Create a Record on page 34.
- For an example of create a new record along with providing blob data for the record, see Insert or Update Blob Data on page 41.

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: Bearer` ***`token`***

Parameters

none required

Example

See Get Field and Other Metadata for an Object on page 32.

SObject Layouts

Returns a list of layouts and descriptions, including for publisher actions. The list of fields and the layout name are returned. This resource is available in REST API version 28.0 and later. When working with publisher actions, also refer to Quick Actions.

Note: In the application, QuickActions are referred to as actions.

URI

To return descriptions of global layouts, the URI is: `/vXX.X/sobjects/Global/describe/layouts/`

For a layout description for a specific object, use `/vXX.X/sobjects/`***`Object`***`/describe/layouts/`

Formats

JSON, XML

HTTP Method

HEAD, GET, POST

Authentication

`Authorization: Bearer` ***`token`***

Parameters

None required

Example for getting layouts

```
curl https://na1.salesforce.com/services/data/v28.0/sobjects/global/describe/layouts/
-H "Authorization: Bearer token"
```

Example for describing a layout

```
curl
https://na1.salesforce.com/services/data/v28.0/sobjects/global/describe/layouts/ContactLayout
 -H 'Authorization: Bearer access_token -H "Content-Type: application/json" -d
@contactlayout.json'
```

Example JSON Response body `contactlayout.json` file

```
{

   "Name" : { "Joe Smith" },
   "Description" : "This is the company owner and purchasing decision maker."
   "url":"/services/data/v26.0/sobjects/Site/describe"
}
```

```
{

   "label" : { "Joe Smith" },
   "Description" : "This is the company owner and purchasing decision maker."
   "url":"/services/data/v26.0/sobjects/Site/describe"
}
```

```
[ { "name" : "contactlayout",
    "searchColumns" : [ { "field" : "Account.Name",
          "format" : null,
          "label" : "Account Name",
          "name" : "Name"
        },
        { "field" : "Account.Site",
          "format" : null,
          "label" : "Account Site",
          "name" : "Site"
        },
        { "field" : "Account.Phone",
          "format" : null,
          "label" : "Phone",
          "name" : "Phone"
        },
        { "field" : "User.Alias",
          "format" : null,
          "label" : "Account Owner Alias",
          "name" : "Owner.Alias"
        }
      ]
  },
  { "label" : "Search Results",
    "limitRows" : 25,
    "searchColumns" : [ { "field" : "Contact.Name",
          "format" : null,
          "label" : "Name",
          "name" : "Name"
        },
        { "field" : "Account.Name",
          "format" : null,
          "label" : "Account Name",
          "name" : "Account.Name"
        },
        { "field" : "Account.Site",
          "format" : null,
          "label" : "Account Site",
          "name" : "Account.Site"
        },
        { "field" : "Contact.Phone",
          "format" : null,
          "label" : "Phone",
          "name" : "Phone"
        },
        { "field" : "Contact.Email",
          "format" : null,
          "label" : "Email",
          "name" : "Email"
        },
        { "field" : "User.Alias",
          "format" : null,
          "label" : "Contact Owner Alias",
          "name" : "Owner.Alias"
        }
      ]
  },
  { "label" : "Search Results",
    "limitRows" : 25,
    "searchColumns" : [ { "field" : "Lead.Name",
          "format" : null,
          "label" : "Name",
          "name" : "Name"
        },
        { "field" : "Lead.Title",
          "format" : null,
          "label" : "Title",
          "name" : "Title"
        },
        { "field" : "Lead.Phone",
          "format" : null,
          "label" : "Phone",
          "name" : "Phone"
        },
        { "field" : "Lead.Company",
          "format" : null,
          "label" : "Company",
          "name" : "Company"
        },
        { "field" : "Lead.Email",
          "format" : null,
          "label" : "Email",
          "name" : "Email"
        },
        { "field" : "Lead.Status",
          "format" : null,
          "label" : "Lead Status",
          "name" : "toLabel(Status)"
        },
        { "field" : "Name.Alias",
          "format" : null,
          "label" : "Owner Alias",
          "name" : "Owner.Alias"
        }
        ]
    },
]
```

SObject Rows

Accesses records based on the specified object ID. Retrieves, updates, or deletes records. This resource can also be used to retrieve field values. Use the GET method to retrieve records or fields, the DELETE method to delete records, and the PATCH method to update records.

To create new records, use the SObject Basic Information resource.

URI

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

Formats

JSON, XML

HTTP Method

GET, PATCH, DELETE

Authentication

`Authorization: Bearer` ***`token`***

Parameters

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

Examples

- For an example of retrieving field values using GET, see Get Field Values from Records on page 36.
- For an example of updating a record using PATCH, see Update a Record on page 34.
- For an example of deleting a record using DELETE, see Delete a Record on page 36.
- For an example of updating the blob data for an object, see Insert or Update Blob Data on page 41.

SObject Rows by External ID

Creates new records or updates existing records (upserts records) based on the value of a specified external ID field.

- If the specified value doesn't exist, a new record is created.
- If a record does exist with that value, the field values specified in the request body are updated.
- If the value is not unique, the REST API returns a 300 response with the list of matching records.

Note: Do not specify `Id` or an external ID field in the request body or an error is generated.

URI

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

Formats

JSON, XML

HTTP Method

HEAD, GET, PATCH, DELETE

Authentication

`Authorization: Bearer` ***`token`***

Parameters

None

Examples

- For an example of retrieving a record based on an external ID, see Retrieve a Record Using an External ID on page 36.
- For examples of creating and updating records based on external IDs, see Insert or Update (Upsert) a Record Using an External ID on page 37.

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: Bearer` ***`token`***

Parameters

none required

Example

For an example of retrieving the blob data from an Attachment or Document, see Get Attachment Content from a Record on page 40.

Error responses

See Status Codes and Error Responses on page 69.

SObject Quick Actions

Returns a list of publisher actions and details. This resource is available in REST API version 28.0 and later. When working with publisher actions, also refer to Quick Actions.

Note: In the application, QuickActions are referred to as actions.

URI

To return a specific object's actions as well as global actions, use: `/vXX.X/sobjects/`***`object`***`/quickActions/`

To return a specific action, use `/vXX.X/sobjects/`***`object`***`/quickActions/`***`{action name}`***

To return a specific action's descriptive detail, use `/vXX.X/sobjects/`***`object`***`/quickActions/`***`{action name}`***`/describe/`

To return a specific action's default values, including default field values, use `services/data/vXX.X/sobjects/`***`object`***`/quickActions/`***`{action name}`***`/defaultValues/`

To evaluate the default values for an action, use `vXX.X/sobjects/`***`object`***`/quickActions/{action name}/defaultValues/{parent id}`

Formats

JSON, XML

HTTP Method

HEAD, GET, POST

Authentication

`Authorization: Bearer` ***`token`***

Parameters

None required

Example for getting Account actions

```
curl https://na1.salesforce.com/services/data/v28.0/sobjects/Account/quickActions -H
"Authorization: Bearer token"
```

Example for creating a contact on Account using an action

```
curl
https://na1.salesforce.com/services/data/v28.0/sobjects/Account/quickActions/CreateContact
 -H 'Authorization: Bearer access_token -H "Content-Type: application/json" -d
@newcontact.json'
```

Example JSON request body `newcontact.json` file

```
{

   "parentId" : "001D000000JRSGf",
   "record" : { "LastName" : "Smith" }

}
```

Considerations

- The resources return all actions—both global and standard—in addition to the ones requested.

SObject User Password

Set, reset, or get information about a user password. This resource is available in REST API version 24.0 and later.

URI

`/vXX.X/sobjects/User/`***`user ID`***`/password`

For managing passwords for self-service users, the URI is:

`/vXX.X/sobjects/SelfServiceUser/`***`self service user ID`***`/password`

Formats

JSON, XML

HTTP Method

HEAD, GET, POST, DELETE

Authentication

`Authorization: Bearer `***`token`***

Parameters

None required

Example

For examples of getting password information, setting a password, and resetting a password, see Manage User Passwords on page 50.

Considerations

- If the session does not have permission to access the user information, an INSUFFICIENT_ACCESS error will be returned.
- When using this resource to set a new password, the new password must conform to the password policies for the organization, otherwise you will get an INVALID_NEW_PASSWORD error response.
- You can only set one password per request.
- When you use the DELETE method of this resource, Salesforce will reset the user password to an auto-generated password, which will be returned in the response.

Query

Executes the specified SOQL query. If the query results are too large, the response contains the first batch of results and a query identifier. The identifier can be used in an additional request to retrieve the next batch.

URI

`/vXX.X/query/?q=`***`SOQL query`***

For retrieving additional query results if the initial results are too large:

`/vXX.X/query/`***`query identifier`***

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: Bearer `***`token`***

Parameters

Parameter	Description
`q`	A SOQL query. Note that you will need to replace spaces with "+" characters in your query string to create a valid URI. An example query parameter string might look like: "SELECT+Name+FROM+MyObject"

Example

For an example of making a query and retrieving additional query results, see Execute a SOQL Query on page 45

For more information on SOQL see the *Force.com SOQL and SOSL Reference*. For more information on query batch sizes, see Changing the Batch Size in Queries in the *SOAP API Developer's Guide*.

Quick Actions

Returns a list of global publisher actions and standard actions. This resource is available in REST API version 28.0 and later. When working with publisher actions, also refer to SObject Quick Actions.

Note: In the application, QuickActions are referred to as actions.

URI

`/vXX.X/quickActions/`

Formats

JSON, XML

HTTP Method

HEAD, GET, POST

Authentication

`Authorization: Bearer token`

Parameters

None required

Example usage for getting global quick actions

```
curl https://na1.salesforce.com/services/data/v28.0/quickActions/ -H "Authorization:
Bearer token"
```

Example for creating a contact using an action

```
curl https://na1.salesforce.com/services/data/v28.0/quickActions/CreateContact -H
'Authorization: Bearer access_token -H "Content-Type: application/json" -d
@newcontact.json'
```

Example JSON request body `newcontact.json` file

```
{

   "record" : { "LastName" : "Smith" }

}
```

Search

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

URI

`/vXX.X/search/?q=SOSL search string`

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: Bearer token`

Parameters

Parameter	Description
`q`	A SOSL statement that is properly URL-encoded.

Example

See Search for a String on page 46.

For more information on SOSL see the *Force.com SOQL and SOSL Reference*.

Search Scope and Order

Returns an ordered list of objects in the default global search scope of a logged-in user. Global search keeps track of which objects the user interacts with and how often and arranges the search results accordingly. Objects used most frequently appear at the top of the list.

The returned list reflects the object order in the user's default search scope, including any pinned objects on the user's search results page. This call is useful if you want to implement a custom search results page using the optimized global search scope. The search string must be URL-encoded.

URI

`/vXX.X/search/scopeOrder`

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: Bearer` ***`token`***

Example

See Get the Default Search Scope and Order.

Search Result Layouts

Returns search result layout information for the objects in the query string. For each object, this call returns the list of fields displayed on the search results page as columns, the number of rows displayed on the first page, and the label used on the search results page. This call supports bulk fetch for up to 100 objects in a query.

URI

`/vXX.X/``search/layout/?q=`***`Comma delimited object list`***

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: Bearer token`

Example

Get Search Result Layouts for Objects

Recently Viewed Items

Gets the most recently accessed items that were viewed or referenced by the current user.

This resource only accesses most recently used item information. If you want to modify the list of recently viewed items, you'll need to access the RecentlyViewed object directly by using a SOQL Query with a FOR VIEW or FOR REFERENCE clause.

URI

`/vXX.X/recent`

Formats

JSON, XML

HTTP Method

GET

Authentication

`Authorization: Bearer token`

Parameters

Parameter	Description
`limit`	An optional maximum number of results that will be returned. If this parameter is not specified, the default maximum number of records returned is the maximum number of entries in RecentlyViewed, which is 200.

Example

Example response JSON data for a request of the form `services/data/v28.0/recents/?limit=2`:

```
{
    "attributes" :
    {
        "type" : "Merchandise__c",
        "url" : "/services/data/v27.0/sobjects/Merchandise__c/a06U000000CelH0IAJ"
    },
    "Id" : "a06U000000CelH0IAJ",
    "Name" : "Extras"
},
{
    "attributes" :
    {
        "type" : "Merchandise__c",
        "url" : "/services/data/v27.0/sobjects/Merchandise__c/a06U000000CelGvIAJ"
    },
    "Id" : "a06U000000CelGvIAJ",
    "Name" : "Merch1"
}
```

See Also:

RecentlyViewed
SOQL FOR VIEW Clause
SOQL FOR REFERENCE Clause

Headers

This section lists custom HTTP request and response headers used for REST API

- Assignment Rule Header

Assignment Rule Header

The Assignment Rule header is a request header applied when creating or updating Cases or Leads. If enabled, the active assignment rules are used. If disabled, the active assignment rules are not applied. If a valid AssignmentRule ID is provided, the AssignmentRule is applied. If the header is not provided with a request, REST API defaults to using the active assignment rules.

Header Field Name and Values

Field name

`SForce-Auto-Assign`

Field values

- `TRUE`. Active assignment rules are applied for created or updated Cases or Leads. This is the default if the header is not provided in the request.
- `FALSE`. Active assignment rules are not applied for created or updated Cases or Leads.
- Valid AssignmentRule ID. The given AssignmentRule is applied for created Cases or Leads.

The `TRUE` and `FALSE` values are not case-sensitive.

Example

```
SForce-Auto-Assign: FALSE
```

Status Codes and Error Responses

Either when an error occurs or when a response is successful, the response header contains an HTTP code, and the response body usually contains:

- The HTTP response code
- The message accompanying the HTTP response code
- The field or object where the error occurred (if the response returns information about an error)

HTTP response code	Description
200	“OK” success code, for GET or HEAD request.
201	“Created” success code, for POST request.
204	“No Content” success code, for DELETE request.
300	The value returned when an external ID exists in more than one record. The response body contains the list of matching records.
400	The request couldn’t be understood, usually because the JSON or XML body contains an error.
401	The session ID or OAuth token used has expired or is invalid. The response body contains the `message` and `errorCode`.
403	The request has been refused. Verify that the logged-in user has appropriate permissions.
404	The requested resource couldn’t be found. Check the URI for errors, and verify that there are no sharing issues.
405	The method specified in the Request-Line isn’t allowed for the resource specified in the URI.
415	The entity in the request is in a format that’s not supported by the specified method.
500	An error has occurred within Force.com, so the request couldn’t be completed. Contact salesforce.com Customer Support.

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 doesn’t exist, for example, if you try to create a record using a misspelled object name

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

Index