



Version 30.0: Spring '14

Database.com REST API Developer's Guide



Last updated: May 5, 2014

Table of Contents

GETTING STARTED WITH THE DATABASE.COM REST API.....	1
Chapter 1: Introducing Force.com REST API.....	1
Understanding Force.com REST Resources.....	2
Using Compression.....	3
Using cURL in the REST Examples.....	3
Understanding Authentication.....	4
Defining Connected Apps.....	4
Understanding OAuth Endpoints.....	5
Understanding the Web Server OAuth Authentication Flow.....	5
Understanding the User-Agent OAuth Authentication Flow.....	9
Understanding the Username-Password OAuth Authentication Flow.....	12
Understanding the OAuth Refresh Token Process.....	14
Finding Additional Resources.....	16
QUICK START.....	17
Chapter 2: Step 1: Obtain an Organization.....	17
Chapter 3: Step 2: Create Objects and Fields.....	18
Create Widget Object.....	18
Create Model Object.....	18
Relate the Objects.....	19
Chapter 4: Step 4: Create a Remote Access Application.....	20
Chapter 5: Step 5: Walk Through the Sample Code.....	21
Java Sample Code.....	21
USING REST RESOURCES.....	27
Chapter 6: Using REST API Resources.....	27
Getting Information About My Organization.....	28
List Available REST API Versions.....	28
List Organization Limits.....	29
List Available REST Resources.....	30
Get a List of Objects.....	30
Working with Object Metadata.....	31
Retrieve Metadata for an Object.....	31
Get Field and Other Metadata for an Object.....	32
Get Object Metadata Changes.....	33
Working with Records.....	34
Create a Record.....	34
Update a Record.....	34
Delete a Record.....	35

Get Field Values from Records.....	36
Retrieve a Record Using an External ID.....	36
Insert or Update (Upsert) a Record Using an External ID.....	37
Get a List of Deleted Records Within a Given Timeframe.....	40
Get a List of Updated Records Within a Given Timeframe.....	41
Working with Searches and Queries.....	41
Execute a SOQL Query.....	42
Execute a SOQL Query that Includes Deleted Items.....	43
Get Feedback on Query Performance.....	44
Search for a String.....	45
Managing User Passwords.....	45
Manage User Passwords.....	46
Working with Approval Processes and Process Rules.....	47
Get a List of All Approval Processes.....	47
Submit a Record for Approval.....	48
Approve a Record.....	48
Reject a Record.....	49
Bulk Approvals.....	50
Get a List of Process Rules.....	51
Get a Particular Process Rule.....	51
Trigger Process Rules.....	52
REST API REFERENCE.....	53
Chapter 7: Reference.....	53
Versions.....	55
Resources by Version.....	55
Limits.....	56
Describe Global.....	56
SObject Basic Information.....	57
SObject Describe.....	57
SObject Get Deleted.....	58
SObject Get Updated.....	59
SObject Rows.....	60
SObject Rows by External ID.....	61
SObject ApprovalLayouts.....	62
SObject CompactLayouts.....	63
SObject Layouts.....	64
SObject Quick Actions.....	64
SObject User Password.....	65
AppMenu.....	66
FlexiPage.....	69
Process Approvals.....	71
Process Rules.....	72
Query.....	73

QueryAll.....	75
Quick Actions.....	76
Search.....	76
Headers.....	77
Limit Info Header.....	77
Status Codes and Error Responses.....	78
Index.....	80

GETTING STARTED WITH THE DATABASE.COM REST API

Chapter 1

Introducing Force.com REST API

In this chapter ...

- [Understanding Force.com REST Resources](#)
- [Using Compression](#)
- [Using cURL in the REST Examples](#)
- [Understanding Authentication](#)

[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 Database.com 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 Database.com organization. For example, you can:

- Retrieve summary information about the API versions available to you.
- Obtain detailed information about a custom object.
- Perform a query or search.
- Update or delete records.

Suppose you want to retrieve information about the Database.com version. To do this, submit a request for the [Versions](#) resource (this example uses cURL on the *nal* instance):

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

The output from this request is as follows:

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



Note: Database.com runs on multiple server instances. The examples in this guide use the *nal* 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, `/Widget__c/a01D000000INjVe.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/v30.0/
-H "Authorization: Bearer
00D50000000IehZ\!AQcAQH0dMHZfz972Szmpkb58urFRkgeBGsxL_QJWwYMfAbUeeG7c1E6
LYUfiDUkWe6H34r1AAwOR8B8fLEz6n04NPGRrq0FM"
```

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

```
curl https://instance_name.salesforce.com/services/data/v30.0/
-H 'Authorization: Bearer sessionID'
```

Understanding Authentication

Database.com 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 Database.com.
- 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 Database.com. 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 Connected Apps

To authenticate using OAuth, you must define a connected app in Database.com.

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

Use the following steps to create a new connected app.

1. From Setup, click **Create > Apps** and click **New**.
2. Enter the name of your application.
3. Enter the contact email information, as well as any other information appropriate for your application.

4. Select **Enable OAuth Settings**.
5. 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.
6. Add all supported OAuth scopes to **Selected OAuth Scopes**. These scopes refer to permissions given by the user running the connected app.
7. Enter a URL for `Info URL`. This is where the user can go for more information about your application.
8. 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. See the Database.com online help for more information about connected apps.

Understanding OAuth Endpoints

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

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

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

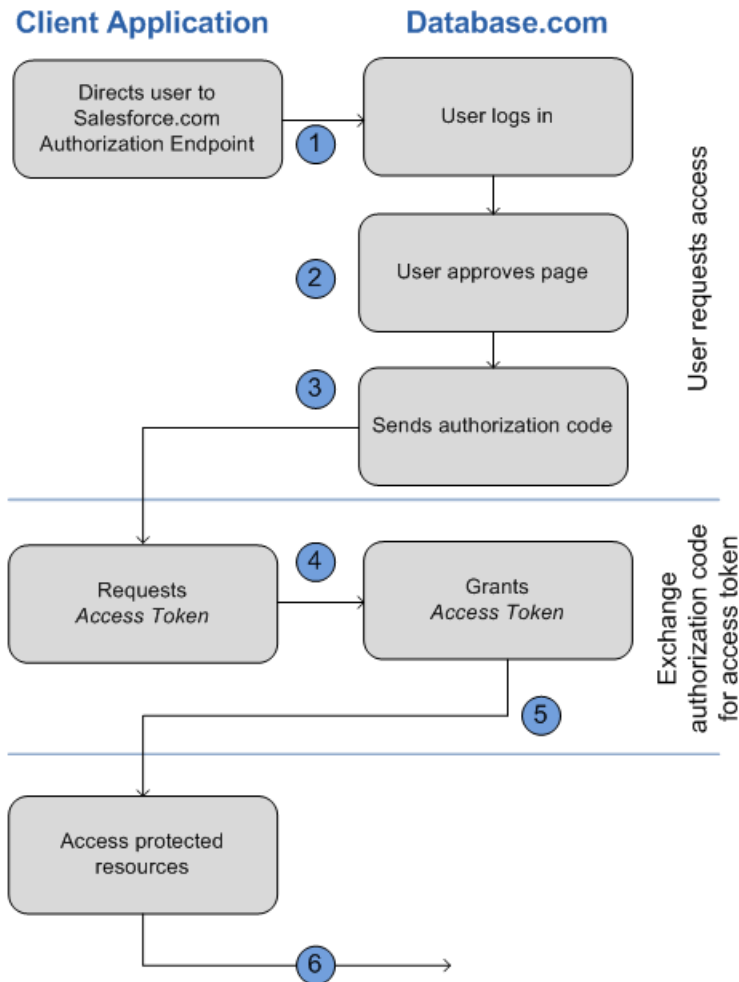
All endpoints require secure HTTP (HTTPS). Each OAuth flow defines which endpoints you need to use and what request data you need to provide.

If you're verifying authentication on a test organization, use "test.database.com" instead of "login.database.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 Database.com authorization endpoint, such as <https://login.database.com/services/oauth2/authorize>. The following parameters are required:

Parameter	Description
<code>response_type</code>	Must be code for this authentication flow.
<code>client_id</code>	The Consumer Key from the remote access application definition.
<code>redirect_uri</code>	The Callback URL from the remote access application definition.

The following parameters are optional:

Parameter	Description
<code>display</code>	<p>Changes the login page's display type. Valid values are:</p> <ul style="list-style-type: none"> <code>page</code>—Full-page authorization screen. This is the default value if none is specified. <code>popup</code>—Compact dialog optimized for modern Web browser popup windows.

Parameter	Description
	<ul style="list-style-type: none"> <code>touch</code>—Mobile-optimized dialog designed for modern smartphones such as Android and iPhone. <code>mobile</code>—Mobile optimized dialog designed for smartphones such as BlackBerry OS 5 that don't support touch screens.
<code>immediate</code>	<p>Determines whether the user should be prompted for login and approval. Values are either <code>true</code> or <code>false</code>. Default is <code>false</code>.</p> <ul style="list-style-type: none"> If set to <code>true</code>, and if the user is currently logged in and has previously approved the application, the approval step is skipped. If set to <code>true</code> and the user is not logged in or has not previously approved the application, the session is immediately terminated with the <code>immediate_unsuccessful</code> error code.
<code>state</code>	Specifies any additional URL-encoded state data to be returned in the callback URL after approval.
<code>scope</code>	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.database.com/services/oauth2/authorize?response_type=code&client_id=
3MVG9lKcPoNINVBIPJjdwlJ9LLM82HnFVVX19KY1uA5mu0QqEWhqKpoW3svG3XHrXDICQjKlmdgAvhCscA
9GE&redirect_uri=https%3A%2F%2Fwww.mysite.com%2Fcode_callback.jsp&state=mystate
```

- The user logs into Database.com 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.
- Once Database.com 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. Database.com appends authorization information to the redirect URL with the following values:

Parameters	Description
<code>code</code>	Authorization code the consumer must use to obtain the access and refresh tokens.
<code>state</code>	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=aWekysIEeqM9PiT
hEfm0Cnr6MoLIfwYRJcQqHdF8f9INökharAS09ia7UNP6RiVScerfhc4w%3D%3D
```

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

Parameter	Description
grant_type	Value must be <code>authorization_code</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	<p>Expected return format. The default is <code>json</code>. Values are:</p> <ul style="list-style-type: none"> <code>urlencoded</code> <code>json</code> <code>xml</code> <p>The return format can also be specified in the header of the request using one of the following:</p> <ul style="list-style-type: none"> <code>Accept: application/x-www-form-urlencoded</code> <code>Accept: application/json</code> <code>Accept: application/xml</code>

An example access token POST request might look something like:

```
POST /services/oauth2/token HTTP/1.1
Host: login.database.com
grant_type=authorization_code&code=aPrxsmIEeqM9PiQroGEWx1UiMQd95_5JUZ
VEhsOFhS8EVvbfYBBJli2W5fn3zbo.8hojaNW_lg%3D%3D&client_id=3MVG9lKcPoNI
NVBIPJjdw1J9LLM82HnFVVX19KY1uA5mu0QqEWhqKpoW3svG3XhRXDiCQjKlmdgAvhCs
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.

Parameters	Description
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 Database.com 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:

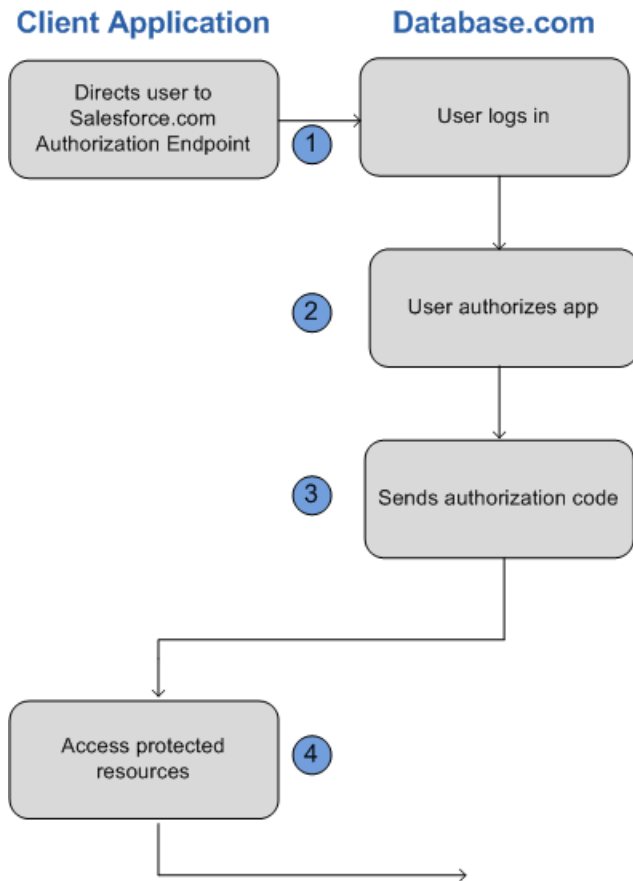
```
{
  "id": "https://login.database.com/id/00Dx0000000BV7z/005x00000012Q9P",
  "issued_at": "1278448101416",
  "refresh_token": "5Aep8614iLM.Dq661ePDmPEgaAW9Oh_L3JKkDpB4xReb54_pZebnUG0h6Sb4KUVDPntWEofWM39yg=",
  "instance_url": "https://na1.salesforce.com",
  "signature": "CMJ41+CCaPQiKjoOEwEig9H4wqhpuLSk4J2urAe+fVg=",
  "access_token": "00Dx0000000BV7z!AR8AQP0jITN80ESESj5EbaZTFG0RNBA1cyWk7TrqoDjoNIWQ2ME_sTZzBjfmOE6zMHq6y8PIW4eWze9JksNEkWU1.Cju7m4"
}
```

- 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 Database.com authorization endpoint, such as <https://login.database.com/services/oauth2/authorize>. The following parameters are required:

Parameter	Description
<code>response_type</code>	Must be <code>token</code> for this authentication flow
<code>client_id</code>	The Consumer Key from the remote access application definition.
<code>redirect_uri</code>	The Callback URL from the remote access application definition.

The following parameters are optional:


Parameter	Description
<code>display</code>	<p>Changes the login page's display type. Valid values are:</p> <ul style="list-style-type: none"> <code>page</code>—Full-page authorization screen. This is the default value if none is specified. <code>popup</code>—Compact dialog optimized for modern Web browser popup windows. <code>touch</code>—Mobile-optimized dialog designed for modern smartphones such as Android and iPhone.

Parameter	Description
	<ul style="list-style-type: none"> <code>mobile</code>—Mobile optimized dialog designed for smartphones such as BlackBerry OS 5 that don't support touch screens.
<code>scope</code>	Specifies what data your application can access. See “Scope Parameter Values” in the online help for more information.
<code>state</code>	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.database.com/services/oauth2/authorize?response_type=token&
client_id=3MVG9lKcPoNINVBIPJjdwlJ9LLJbP_pgwoJYyuisjQhr_LLurNDv7AgQvDTZwCoZuD
ZrXcPCmBv4o.8ds.5iE&redirect_uri=https%3A%2F%2Fwww.mysite.com%2Fuser_callback.jsp&
state=mystate
```

- The user logs into Database.com with their credentials. The user interacts with the authorization endpoint directly, so the application never sees the user's credentials.
- 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. Database.com appends access token information to the redirect URL with the following values:

Parameters	Description
<code>access_token</code>	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.
<code>expires_in</code>	Amount of time the access token is valid, in seconds.
<code>refresh_token</code>	<p>Token that can be used in the future to obtain new access tokens.</p> <p> Warning: This value is a secret. You should treat it like the user's password and use appropriate measures to protect it.</p> <p>The refresh token is only returned if the redirect URI is <code>https://login.database.com/services/oauth2/success</code> or used with a custom protocol that is not HTTPS.</p>
<code>state</code>	The state value that was passed in as part of the initial request, if applicable.
<code>instance_url</code>	Identifies the Database.com instance to which API calls should be sent.
<code>id</code>	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.

Parameters	Description
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=00Dx000000BV7z%21AR8
AQBm8J_xr9kLqmZIRyQxZgLCm4HVi41aGtW0qW3JCzf5xdTGGGSovim8FfJkZEqxbjaFbberKGk
8v8AnYrvChG4qJbQo8&refresh_token=5Aep8614iLM.Dq661ePDmPEgaAW9Oh_L3JKkDpB4xR
eb54_pZfVtildPEk8aimw4Hr9ne7VXXVSIQ%3D%3D&expires_in=7200&state=mystate
```

- 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.

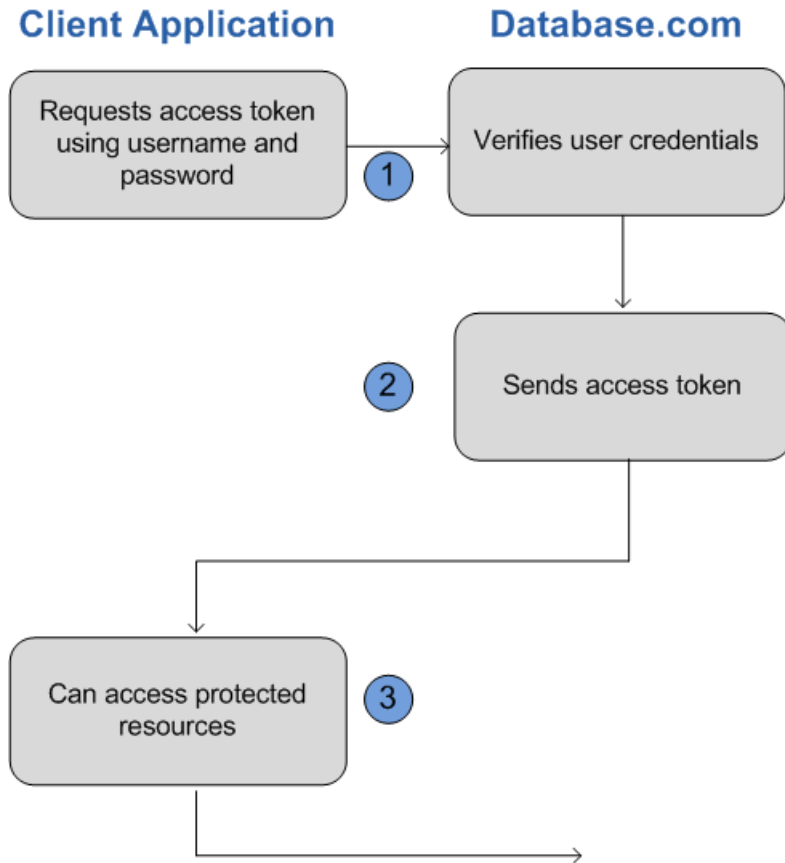
Understanding the Username-Password OAuth Authentication Flow

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.



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 Database.com token request endpoint, such as <https://login.database.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 Database.com. For example, if a user's password is mypassword, and their security token is XXXXXXXXXXXX, then the value provided for this parameter must be mypasswordXXXXXXXXXX. For more 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=3MVG91KcPoNINVBIPJjdwlJ9LLM82Hn
FVVX19KY1uA5mu0QqEWhqKpoW3svG3XHrXDICQjKlmdgAvhCscA9GE&client_secret=
1955279925675241571&username=testuser%40database.com&password=myspassword123456
```

2. Database.com 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 Database.com 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.database.com/id/00Dx0000000BV7z/005x00000012Q9P",
"issued_at":"1278448832702","instance_url":"https://na1.salesforce.com",
"signature":"0CmxinZir53Yex7nE0TD+zMpWIWYGb/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 Database.com 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 Database.com. 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	<p>Expected return format. The default is json. Values are:</p> <ul style="list-style-type: none"> • urlencoded • json • xml <p>The return format can also be specified in the header of the request using one of the following:</p> <ul style="list-style-type: none"> • Accept: application/x-www-form-urlencoded • Accept: application/json • Accept: application/xml <p>This parameter is optional.</p>

An example refresh token POST request might look something like:

```
POST /services/oauth2/token HTTP/1.1
Host: https://login.database.com/
grant_type=refresh_token&client_id=3MVG9lKcPoNINVBIPJjdwlJ9LLM82HnFVVX19KY1uA5mu0
QqEWhqKpoW3svG3XHrXDiCQjKlmdgAvhCscA9GE&client_secret=1955279925675241571
&refresh_token=your token here
```

Once Database.com 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 Database.com 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.

Parameters	Description
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.database.com/id/00Dx0000000BV7z/005x00000012Q9P",
  "issued_at": "1278448384422", "instance_url": "https://nal.salesforce.com",
  "signature": "SSSbLO/gBhmmYNUvN18ODBDFYHzakxOMgqYtu+hDPsc=",
  "access_token": "00Dx0000000BV7z!AR8AQP0jITN80ESESj5EbaZTFG0RNBaT1cyWk7T
  rqoDjoNlWQ2ME_sTZzBjfmOE6zMHq6y8PIW4eWze9JksNEkU1.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 Database.com 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 Database.com:

- [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](#)

QUICK START

Chapter 2

Step 1: Obtain an Organization

If you don't already have an account, go to www.database.com and follow the instructions for joining.

If you already have an 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 Database.com online help.

Chapter 3

Step 2: Create Objects and Fields

In this step you'll create two objects, widget and model, each with a custom field. Then you'll relate the objects to each other with a one-to-many-relationship.

Create Widget Object

Create Model Object

Relate the Objects

Create Widget Object

To create the widget object with a widget cost field:

1. Click **Create > Objects**.
2. Click **New Custom Object**.
3. Enter the information for the widget object:
 - Label: Widget
 - Plural label: Widgets
 - Object name: Widget
 - Record name: Widget Name
 - Data type: Text
4. Leave all other settings as they are and click **Save**.
5. In the Custom Fields & Relationships related list, click **New**.
6. For Data Type, select **Currency** and click **Next**.
7. Enter the custom field details.
 - Field Label: Widget Cost
 - Length: 10
 - Decimal places: 2
 - Field Name: Widget_Cost
8. Leave the remaining settings as they are and click **Next**.
9. Click **Save** to accept the default field-level security settings.

Create Model Object

To create the model object with a model number field:

1. Click **Create > Objects**.

2. Click **New Custom Object**.
3. Enter the information for the model object:
 - Label: `Model`
 - Plural label: `Models`
 - Object name: `Model`
 - Record name: `Model Name`
 - Data type: `Text`
4. Leave all other settings as they are and click **Save**.
5. In the Custom Fields & Relationships related list, click **New**.
6. For Data Type, select `Text` and click **Next**.
7. Enter the custom field details.
 - Field Label: `Model Number`
 - Length: `10`
 - Field Name: `Model_Number`
8. Leave the remaining settings as they are and click **Next**.
9. Click **Save** to accept the default field-level security settings.

Relate the Objects

1. If you aren't already in the Model detail page, click **Create > Objects**, then select the Model object.
2. In the Custom Fields & Relationships related list, click **New**.
3. In the New Custom Field page, select `Master-Detail Relationship` and click **Next**.
4. In the Related To field, select the Widget object and click **Next**.
5. Accept the defaults on the remaining screens by clicking **Next** and then **Save**.

Chapter 4

Step 4: Create a Remote Access Application

External applications use the OAuth protocol to verify both the Database.com user and the external application itself. To provide this functionality to your application, create a remote access application for your organization:

1. Log in to your organization. Logins are checked to ensure they are from a known IP address.
2. Click **Develop** > **Remote Access** to display the Remote Access page.
3. Click **New**.
4. Enter the information for the remote access application:
 - Application: `MyRemoteAccessApplication`
 - Callback URL: `https://no_redirect_uri`
 - Contact Email: `your_email@domain.ext`
5. Click **Save**.

Chapter 5

Step 5: Walk Through the Sample Code

Once you've created your remote application, you can begin building client applications that use the REST API. Use the following samples to create a basic client application. Comments embedded in the sample explain each section of code.

Java Sample Code

This section walks through a sample Java client application that uses the REST API. The purpose of this sample application is to show the required steps for logging into the login server and to demonstrate the invocation and subsequent handling of several REST API calls. This sample application performs the following main tasks:

1. Prompts the user for
 - API version
 - login URL
 - username
 - password
 - OAuth 2.0 consumer key
 - OAuth 2.0 consumer secret
2. Uses the information from the previous step to log in to the single login server and, if the login succeeds:
3. Sends an HTTP GET request to the server URL:
`https://instance.salesforce.com/services/data/v24.0/objects/`. This is equivalent to a calling `describeGlobal()` to retrieve a list of all available objects for the organization's data.
4. Sends an HTTP GET request to the server URL:
`https://instance.salesforce.com/services/data/v24.0/objects/Merchandise__c/describe/`. This is equivalent to a calling `describeSObject()` to retrieve metadata (field list and object properties) for the specified object.
5. Sends an HTTP POST request to the server URL:
`https://instance.salesforce.com/services/data/v24.0/objects/Merchandise__c/` passing a JSON object in the request body. This is equivalent to a calling `create()` to a record corresponding to the JSON object.
6. Sends an HTTP GET request to the server URL:
`https://instance.salesforce.com/services/data/v24.0/query/?q=SELECT+Name+FROM+Merchandise__c`. This is equivalent to a calling `query()`, passing a simple query string ("SELECT Name FROM Merchandise__c"), and iterating through the returned `QueryResult`.

Java Sample Code

Java Sample Code

```
package com.example.sample.rest;

import java.awt.Desktop;
import java.io.BufferedReader;
```

```

import java.io.FileNotFoundException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URLEncoder;

import org.apache.http.Header;
import org.apache.http.HttpResponse;
import org.apache.http.StatusLine;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpParams;

import com.google.gson.Gson;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

public class RestClient extends Object {
    private static BufferedReader reader =
        new BufferedReader(new InputStreamReader(System.in));
    private static String OAUTH_ENDPOINT = "/services/oauth2/token";
    private static String REST_ENDPOINT = "/services/data";
    UserCredentials userCredentials;
    String restUri;
    Header oauthHeader;
    Header prettyPrintHeader = new BasicHeader("X-PrettyPrint", "1");
    Gson gson;
    OAuth2Response oauth2Response;

    public static void main(String[] args) {
        RestClient client = new RestClient();
        client.oauth2Login( client.getUserCredentials() );
        client.testRestData();
    }

    public RestClient() {
        gson = new Gson();
    }

    public HttpResponse oauth2Login(UserCredentials userCredentials) {
        HttpResponse response = null;
        this.userCredentials = userCredentials;
        String loginHostUri = "https://" +
            userCredentials.loginInstanceDomain + OAUTH_ENDPOINT;
        try {
            HttpClient httpClient = new DefaultHttpClient();
            HttpPost httpPost = new HttpPost(loginHostUri);
            StringBuffer requestBodyText =
                new StringBuffer("grant_type=password");
            requestBodyText.append("&username=");
            requestBodyText.append(userCredentials.userName);
            requestBodyText.append("&password=");
            requestBodyText.append(userCredentials.password);
            requestBodyText.append("&client_id=");
            requestBodyText.append(userCredentials.consumerKey);
            requestBodyText.append("&client_secret=");
            requestBodyText.append(userCredentials.consumerSecret);
            StringEntity requestBody =
                new StringEntity(requestBodyText.toString());
            requestBody.setContentType("application/x-www-form-urlencoded");
            httpPost.setEntity(requestBody);

```

```

        httpPost.addHeader(prettyPrintHeader);
        response = httpClient.execute(httpPost);
        if ( response.getStatusLine().getStatusCode() == 200 ) {
            InputStreamReader inputStream = new InputStreamReader(
                response.getEntity().getContent()
            );
            oauth2Response = gson.fromJson( inputStream,
                OAuth2Response.class );
            restUri = oauth2Response.instance_url + REST_ENDPOINT +
                "/v" + this.userCredentials.apiVersion + ".0";
            System.out.println("\nSuccessfully logged in to instance: " +
                restUri);
            oauthHeader = new BasicHeader("Authorization", "OAuth " +
                oauth2Response.access_token);
        } else {
            System.out.println("An error has occurred.");
            System.exit(-1);
        }
    } catch (UnsupportedEncodingException uee) {
        uee.printStackTrace();
    } catch (IOException ioe) {
        ioe.printStackTrace();
    } catch (NullPointerException npe) {
        npe.printStackTrace();
    }
    return response;
}

public void testRestData() {
    String responseBody = restGet(restUri);
    responseBody = restGet(restUri + "/subjects/");
    responseBody = restGet(restUri +
        "/subjects/Merchandise__c/describe/");
    responseBody = restPost(restUri +
        "/subjects/Merchandise__c/", "{ \"Name\" : \"Wee Jet\" }\n\n");
    System.out.println(responseBody);
    JsonParser jsonParser = new JsonParser();
    JsonElement jsonElement = jsonParser.parse(responseBody);
    String id = jsonElement.getAsJsonObject().get("id").getString();
    responseBody = restGet(restUri +
        "/subjects/Merchandise__c/" + id);
    System.out.println(responseBody);
    responseBody = restPost(restUri +
        "/subjects/Merchandise__c/", "{ \"Name\" : \"Zeppelin GmbH\" }\n\n");
    System.out.println(responseBody);
    responseBody = restGet(restUri +
        "/query/?q=SELECT+Name+FROM+Merchandise__c");
    System.out.println(responseBody);
    responseBody = restPatch(restUri +
        "/subjects/Merchandise__c/" + id, "{ \"Name\" : \"Dry Twig.\" }\n\n");
    System.out.println(responseBody);
    responseBody = restGet(restUri +
        "/subjects/Merchandise__c/" + id);
    System.out.println(responseBody);
}

public String restGet(String uri) {
    String result = "";
    printBanner("GET", uri);
    try {
        HttpClient httpClient = new DefaultHttpClient();
        HttpGet httpGet = new HttpGet(uri);
        httpGet.addHeader(oauthHeader);
        httpGet.addHeader(prettyPrintHeader);
        HttpResponse response = httpClient.execute(httpGet);
        result = getBody( response.getEntity().getContent() );
    } catch (IOException ioe) {
        ioe.printStackTrace();
    } catch (NullPointerException npe) {
        npe.printStackTrace();
    }
}

```

```

    }
    return result;
}

public String restPatch(String uri, String requestBody) {
    String result = "";
    printBanner("PATCH", uri);
    try {
        HttpClient httpClient = new DefaultHttpClient();
        HttpPatch httpPatch = new HttpPatch(uri);
        httpPatch.addHeader(oauthHeader);
        httpPatch.addHeader(prettyPrintHeader);
        StringEntity body = new StringEntity(requestBody);
        body.setContentType("application/json");
        httpPatch.setEntity(body);
        HttpResponse response = httpClient.execute(httpPatch);
        result = response.getEntity() != null ?
            getBody( response.getEntity().getContent() ) : "";
    } catch (IOException ioe) {
        ioe.printStackTrace();
    } catch (NullPointerException npe) {
        npe.printStackTrace();
    }
    return result;
}

public String restPatchXml(String uri, String requestBody) {
    String result = "";
    printBanner("PATCH", uri);
    try {
        HttpClient httpClient = new DefaultHttpClient();
        HttpPatch httpPatch = new HttpPatch(uri);
        httpPatch.addHeader(oauthHeader);
        httpPatch.addHeader(prettyPrintHeader);
        httpPatch.addHeader( new BasicHeader("Accept", "application/xml") );
        StringEntity body = new StringEntity(requestBody);
        body.setContentType("application/xml");
        httpPatch.setEntity(body);
        HttpResponse response = httpClient.execute(httpPatch);
        result = getBody( response.getEntity().getContent() );
    } catch (IOException ioe) {
        ioe.printStackTrace();
    } catch (NullPointerException npe) {
        npe.printStackTrace();
    }
    return result;
}

public String restPost(String uri, String requestBody) {
    String result = null;
    printBanner("POST", uri);
    try {
        HttpClient httpClient = new DefaultHttpClient();
        HttpPost httpPost = new HttpPost(uri);
        httpPost.addHeader(oauthHeader);
        httpPost.addHeader(prettyPrintHeader);
        StringEntity body = new StringEntity(requestBody);
        body.setContentType("application/json");
        httpPost.setEntity(body);
        HttpResponse response = httpClient.execute(httpPost);
        result = getBody( response.getEntity().getContent() );
    } catch (IOException ioe) {
        ioe.printStackTrace();
    } catch (NullPointerException npe) {
        npe.printStackTrace();
    }
    return result;
}

/**

```

```

    * Extend the Apache HttpPost method to implement an HttpPost
    * method.
    */
    public static class HttpPatch extends HttpPost {
        public HttpPatch(String uri) {
            super(uri);
        }

        public String getMethod() {
            return "PATCH";
        }
    }

    static class OAuth2Response {
        public OAuth2Response() {
        }
        String id;
        String issued_at;
        String instance_url;
        String signature;
        String access_token;
    }

    class UserCredentials {
        String grantType;
        String userName;
        String password;
        String consumerKey;
        String consumerSecret;
        String loginInstanceDomain;
        String apiVersion;
    }

    // private methods

    private String getUserInput(String prompt) {
        String result = "";
        try {
            System.out.print(prompt);
            result = reader.readLine();
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
        return result;
    }

    private void printBanner(String method, String uri) {
        System.out.println("\n-----\n");

        System.out.println("HTTP Method: " + method);
        System.out.println("REST URI: " + uri);

        System.out.println("\n-----\n");
    }

    private String getBody(InputStream inputStream) {
        String result = "";
        try {
            BufferedReader in = new BufferedReader(
                new InputStreamReader(inputStream)
            );
            String inputLine;
            while ( (inputLine = in.readLine()) != null ) {
                result += inputLine;
                result += "\n";
            }
            in.close();
        } catch (IOException ioe) {

```

```
        ioe.printStackTrace();
    }
    return result;
}

private UserCredentials getUserCredentials() {
    UserCredentials userCredentials = new UserCredentials();
    userCredentials.loginInstanceDomain =
        getUserInput("Login Instance Domain: ");
    userCredentials.apiVersion = getUserInput("API Version: ");
    userCredentials.userName = getUserInput("UserName: ");
    userCredentials.password = getUserInput("Password: ");
    userCredentials.consumerKey = getUserInput("Consumer Key: ");
    userCredentials.consumerSecret = g
        etUserInput("Consumer Secret: ");
    userCredentials.grantType = "password";
    return userCredentials;
}
}
```

USING REST RESOURCES

Chapter 6

Using REST API Resources

In this chapter ...

- [Getting Information About My Organization](#)
- [Working with Object Metadata](#)
- [Working with Records](#)
- [Working with Searches and Queries](#)
- [Managing User Passwords](#)
- [Working with Approval Processes and Process Rules](#)

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>
</Versions>
```

```
<version>26.0</version>
</Version>
</Versions>
```

List Organization Limits



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

Use the [Limits](#) resource to list limits information for your organization.

- Max is the limit total for the organization.
- Remaining is the total number of calls or events left for the organization.

Example usage

```
curl https://instance.salesforce.com/services/data/v29.0/limits/ -H "Authorization:
Bearer token "X-PrettyPrint:1"
```

Example request body

none required

Example response body

```
{
  "DailyApiRequests":
  {
    "Remaining": "4980",
    "Max": "5000"
  },
  "DailyAsyncApexExecutions":
  {
    "Remaining": "250000",
    "Max": "250000"
  },
  "DailyBulkApiRequests":
  {
    "Remaining": "3000",
    "Max": "3000"
  },
  "DailyStreamingApiEvents":
  {
    "Remaining": "1000",
    "Max": "1000"
  },
  "StreamingApiConcurrentClients":
  {
    "Remaining": "10",
    "Max": "10"
  },
  "DataStorageMB":
  {
    "Remaining": 5,
    "Max": 5
  },
  "FileStorageMB":
  {
    "Remaining": 20,
    "Max": 20
  }
}
```

```

    },
    "MassEmail":
    {
        "Remaining":10,
        "Max":10
    },
    "SingleEmail":
    {
        "Remaining":15,
        "Max":15
    }
}

```

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

```

{
  "subjects" : "/services/data/v26.0/subjects",
  "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>
  <subjects>/services/data/v26.0/subjects</subjects>
  <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/subjects/ -H "Authorization: Bearer token"
```

Example request body

none required

Example response body

```
{
  "encoding" : "UTF-8",
  "maxBatchSize" : 200,
  "subjects" : [ {
    "name" : "AggregateResult",
    "label" : "Aggregate Result",
    "keyPrefix" : null,
    "labelPlural" : "Aggregate Result",
    "custom" : false,
    "layoutable" : false,
    "activateable" : false,
    "urls" : {
      "subject" : "/services/data/v26.0/subjects/AggregateResult",
      "describe" : "/services/data/v26.0/subjects/AggregateResult/describe",
      "rowTemplate" : "/services/data/v26.0/subjects/AggregateResult/{ID}"
    },
    "searchable" : false,
    "updateable" : false,
    "createable" : false,
    "deprecatedAndHidden" : false,
    "customSetting" : false,
    "deletable" : false,
    "feedEnabled" : false,
    "mergeable" : false,
    "queryable" : false,
    "replicateable" : false,
    "retrieveable" : false,
    "undeletable" : false,
    "triggerable" : false
  },
  ...
]
```

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 Widget custom object metadata

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

Example request body for retrieving Widget custom object metadata

none required

Example response body for retrieving Widget custom object metadata

```
{
  "objectDescribe" :
  {
    "name" : "Widget__c",
    "updateable" : true,
    "label" : "Widget",
    "keyPrefix" : "a01",
    "custom" : true,

    ...

    "replicateable" : true,
    "retrieveable" : true,
    "undeletable" : true,
    "triggerable" : true
  },
  "recentItems" : [ ]
}
```

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/Widget__c/describe/ -H "Authorization:
Bearer token"
```

Example request body

none required

Example response body

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

```

    "updateable" : true,
    "label" : "Widget",
    "keyPrefix" : "a01",
    "custom" : true,
    ...

    "urls" :
    {
        "uiEditTemplate" : "https://na1.salesforce.com/{ID}/e",
        "subject" : "/services/data/v20.0/subjects/Widget__c",
        "uiDetailTemplate" : "https://na1.salesforce.com/{ID}",
        ...
    },

    "childRelationships" :
    [
        {
            "field" : "FirstPublishLocationId",
            "deprecatedAndHidden" : false,
            ...
        },
        ....
    ],

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

```

Get Object Metadata Changes

Use the SObject Describe resource and the If-Modified-Since HTTP header to determine if object metadata has changed.

If an If-Modified-Since header is provided, along with a date in `EEE, dd MMM yyyy HH:mm:ss z` format, in an [SObject Describe](#) request, response metadata will only be returned if the object metadata has changed since the provided date. If the metadata has not been modified since the provided date, a 304 Not Modified status code is returned, with no response body.

The following example assumes that no changes, such as new custom fields, have been made to the Merchandise__c object after July 3rd, 2013.

Example SObject Describe request

```
/services/data/v29.0/subjects/Merchandise__c/describe
```

Example If-Modified-Since header used with request

```
If-Modified-Since: Wed, 3 Jul 2013 19:43:31 GMT
```

Example response body

No response body returned

Example response status code

```
HTTP/1.1 304 Not Modified
Date: Fri, 12 Jul 2013 05:03:24 GMT
```

If there were changes to `Merchandise__c` made after July 3rd, 2013, the response body would contain the metadata for `Merchandise__c`. See [Get Field and Other Metadata for an Object](#) for an example.

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 `Widget__c` record, with the field values provided in `newwidget.json`.

Example for creating a new Widget

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

Example request body `newwidget.json` file for creating a new Widget

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

Example response body after successfully creating a new Widget

```
{
  "id" : "a01D000000IqhSLIAZ",
  "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 `Widget Cost` within a `Widget` is updated. The updated record information is provided in `patchwidget.json`.

Example for updating a Widget custom object

```
https://na1.salesforce.com/services/data/v20.0/objects/Widget__c/a01D000000INjVe -H "Authorization: Bearer token" -H "Content-Type: application/json" -d @patchwidget.json -X PATCH
```

Example request body `patchwidget.json` file for updating custom field in a Widget custom object

```
{
  "Widget_Cost__c" : 5.75
}
```

Example response body for updating fields in a Widget custom object

none returned

Error response

See [Status Codes and Error Responses](#) on page 78.

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 a Widget record

```
curl https://na1.salesforce.com/services/data/v20.0/subjects/Widget__c/a01D000000INjVe
-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 Name and Widget_Cost__c values are retrieved from a Widget custom object.

Example for retrieving values from fields on a Widget custom object

```
curl https://na1.salesforce.com/services/data/v20.0/subjects/Widget__c/a01D000000INjVe
?fields=Name,Widget_Cost__c -H "Authorization: Bearer token"
```

Example request body

None required

Example response body

```
{
  "attributes" :
  {
    "type" : "Widget__c",
    "url" : "/services/data/v20.0/subjects/Widget__c/a01D000000INjVe"
  }
  "Name" : "Test",
  "Widget_Cost__c" : 5.75,
  "Id" : "a01D000000INjVe"
}
```

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/subjects/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/subjects/Merchandise__c/a00D00000008oWP8IAM"
  },
  "Id" : "a00D00000008oWP8IAM",
  "OwnerId" : "005D00000001KyEIIA0",
  "IsDeleted" : false,
  "Name" : "Example Merchandise",
  "CreatedDate" : "2012-07-12T17:49:01.000+0000",
  "CreatedById" : "005D00000001KyEIIA0",
  "LastModifiedDate" : "2012-07-12T17:49:01.000+0000",
  "LastModifiedById" : "005D00000001KyEIIA0",
  "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 Widget__c. It also assumes that a Widget__c 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/subjects/Widget__c/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" : "A New Widget",
  "Widget_Cost__c" : 6.75
}
```

Response

Successful response:

```
{
  "id" : "a01D0000001pPvHAAU",
  "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 78.

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 Widget__c and a Widget__c 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/subjects/Widget__c/customExtIdField__c/11999
-H "Authorization: Bearer token" -H "Content-Type: application/json" -d @updates.json
-X PATCH
```

Example JSON request body `updates.json` file

```
{
  "Widget_Cost__c" : 5.75
}
```

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 78.

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/subjects/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" : "a02D00000006YUhrIAO",
  "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 78.

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/subjects/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 a List of Deleted Records Within a Given Timeframe

Use the [SObject Get Deleted](#) resource to get a list of deleted records for the specified object. Specify the date and time range within which the records for the given object were deleted. Deleted records are written to a delete log (that is periodically purged), and will be filtered out of most operations, such as SObject Rows or Query (although QueryAll will include deleted records in results).

Example usage for getting a list of `Merchandise__c` records that were deleted between May 5th, 2013 and May 10th, 2013

```
/services/data/v29.0/subjects/Merchandise__c/deleted/
?start=2013-05-05T00%3A00%3A00%2B00%3A00&end=2013-05-10T00%3A00%3A00%2B00%3A00
```

Example request body

None required

JSON example response body

```
{
  "deletedRecords" :
  [
    {
      "id" : "a00D00000008pQRAIA2",
      "deletedDate" : "2013-05-07T22:07:19.000+0000"
    }
  ],
  "earliestDateAvailable" : "2013-05-03T15:57:00.000+0000",
  "latestDateCovered" : "2013-05-08T21:20:00.000+0000"
}
```

XML example response body

```
<?xml version="1.0" encoding="UTF-8"?>
<Merchandise__c>
  <deletedRecords>
    <deletedDate>2013-05-07T22:07:19.000Z</deletedDate>
    <id>a00D00000008pQRAIA2</id>
  </deletedRecords>
  <earliestDateAvailable>2013-05-03T15:57:00.000Z</earliestDateAvailable>
  <latestDateCovered>2013-05-08T21:20:00.000Z</latestDateCovered>
</Merchandise__c>
```

Get a List of Updated Records Within a Given Timeframe

Use the [SObject Get Updated](#) resource to get a list of updated (modified or added) records for the specified object. Specify the date and time range within which the records for the given object were updated.

Example usage for getting a list of Merchandise__c records that were updated between May 6th, 2013 and May 10th, 2013

```
/services/data/v29.0/subjects/Merchandise__c/updated/
?start=2013-05-06T00%3A00%3A00%2B00%3A00&end=2013-05-10T00%3A00%3A00%2B00%3A00
```

Example request body

None required

JSON example response body

```
{
  "ids" :
  [
    "a00D00000008pQR5IAM",
    "a00D00000008pQRGIA2",
    "a00D00000008pQRFIA2"
  ],
  "latestDateCovered" : "2013-05-08T21:20:00.000+0000"
}
```

XML example response body

```
<?xml version="1.0" encoding="UTF-8"?>
<Merchandise__c>
  <ids>a00D00000008pQR5IAM</ids>
  <ids>a00D00000008pQRGIA2</ids>
  <ids>a00D00000008pQRFIA2</ids>
  <latestDateCovered>2013-05-08T21:20:00.000Z</latestDateCovered>
</Merchandise__c>
```

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 [Database.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 Widget records.

Example usage for executing a query

```
curl https://na1.salesforce.com/services/data/v20.0/query/?q=SELECT+name+from+Widget__c
-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" : "Widget__c",
        "url" : "/services/data/v20.0/objects/Widget__c/a01D000000IRFmaIAH"
      },
      "Name" : "Test 1"
    },
    {
      "attributes" :
      {
        "type" : "Widget__c",
        "url" : "/services/data/v20.0/objects/Widget__c/a01D000000IomazIAB"
      },
      "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" : [...]
}
```

Execute a SOQL Query that Includes Deleted Items

Use the [QueryAll](#) resource to execute a SOQL query that includes information about records that have been deleted because of a merge or delete. Use `QueryAll` rather than `Query`, because the `Query` resource will automatically filter out items that have been deleted.

The following query requests the value from the `Name` field from all deleted `Merchandise__c` records, in an organization that has one deleted `Merchandise__c` record. The same query using `Query` instead of `QueryAll` would return no records, because `Query` automatically filters out any deleted record from the result set.

Example usage for executing a query for deleted `Merchandise__c` records

```
/services/data/v29.0/queryAll/?q=SELECT+Name+from+Merchandise__c+WHERE+isDeleted+=+TRUE
```

Example request body for executing a query

none required

Example response body for executing a query

```
{
  "done" : true,
  "totalSize" : 1,
  "records" :
  [
    {
      "attributes" :
      {
        "type" : "Merchandise__c",
        "url" : "/services/data/v29.0/objects/Merchandise__c/a00D0000008pQRAIX2"
      },
      "Name" : "Merchandise 1"
    }
  ]
}
```

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/v29.0/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.

Note that even though `nextRecordsUrl` has `query` in the URL, it will still provide remaining results from the initial `QueryAll` request. The remaining results will include deleted records that matched the initial query.

Example usage for retrieving the remaining results

```
/services/data/v29.0/query/01gD0000002HU6KIAW-2000
```

Example request body for retrieving the remaining results

none required

Example response body for retrieving the remaining results

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

Get Feedback on Query Performance



Note: Using `explain` with the REST API query resource is a pilot feature. There is no support associated with this pilot feature. For more information, contact salesforce.com, inc.

Use the [Query](#) resource along with the `explain` parameter to get feedback on how Database.com will execute your query. Database.com analyzes each query to find the optimal approach to obtain the query results. Depending on the query and query filters, an index or internal optimization might get used. You use the `explain` parameter to return details on how Database.com will optimize your query, without actually running the query. Based on the response, you can decide whether to fine-tune the performance of your query by making changes like adding filters to make the query more selective.

The response will contain one or more query execution plans that, sorted from most optimal to least optimal. The most optimal plan is the plan that's used when the query is executed.

See the `explain` parameter in [Query](#) for more details on the fields returned when using `explain`. See “More Efficient SOQL Queries” in the *Apex Code Developer's Guide* for more information on making your queries more selective.

Example usage for getting performance feedback on a query that uses `Merchandise__c`

```
/services/data/v30.0/query/?explain=
SELECT+Name+FROM+Merchandise__c+WHERE+CreatedDate+=+TODAY+AND+Price__c+>+10.0
```

Example response body for executing a performance feedback query

```
{
  "plans" : [ {
```

```

    "cardinality" : 1,
    "fields" : [ "CreatedDate" ],
    "leadingOperationType" : "Index",
    "relativeCost" : 0.0,
    "subjectCardinality" : 3,
    "subjectType" : "Merchandise__c"
  }, {
    "cardinality" : 1,
    "fields" : [ ],
    "leadingOperationType" : "TableScan",
    "relativeCost" : 0.65,
    "subjectCardinality" : 3,
    "subjectType" : "Merchandise__c"
  } ]
}

```

This response indicates that Database.com found two possible execution plans for this query. The first plan uses the CreatedDate index field to improve the performance of this query. The second plan scans all records without using an index. The first plan will be used if the query is actually executed.

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" : "Widget__c",
      "url" : "/services/data/v20.0/subjects/Widget__c/a01D000000IqhSLIAZ"
    },
    "Id" : "a01D000000IqhSLIAZ"
  },
  {
    "attributes" :
    {
      "type" : "Widget__c",
      "url" : "/services/data/v20.0/subjects/Widget__c/a01D000000IomazIAB"
    },
    "Id" : "a01D000000IomazIAB"
  }
]

```

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.

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/subjects/User/005D00000001KyEIIA0/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/subjects/User/005D00000001KyEIIA0/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/subjects/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>
```

Working with Approval Processes and Process Rules

The examples in this section use REST API resources to work with approval processes and process rules.

Get a List of All Approval Processes

Use the [Process Approvals](#) resource to get information about approvals.

Example usage

```
curl https://na1.salesforce.com/services/data/v30.0/process/approvals/ -H "Authorization:
Bearer token"
```

Example request body

none required

Example JSON response body

```
{
  "approvals" : {
    "Account" : [ {
      "description" : null,
      "id" : "04aD00000008Py9",
      "name" : "Account Approval Process",
      "object" : "Account",
      "sortOrder" : 1
    } ]
  }
}
```

Submit a Record for Approval

Use the [Process Approvals](#) resource to submit a record or a collection of records for approval. Each call takes an array of requests. The entity must support an approval process and an approval process must have already been defined.

Example usage

```
curl https://na1.salesforce.com/services/data/v30.0/process/approvals/ -H "Authorization: Bearer token" -H "Content-Type: application/json" -d @submit.json"
```

Example request body submit.json file

```
{
  "requests" : [{
    "actionType" : "Submit",
    "contextId" : "001D000000I8mIm",
    "nextApproverIds" : ["005D00000015rY9"],
    "comments" : " this is a test"}]
}
```

Example JSON response body

```
[ {
  "actorIds" : [ "005D00000015rY9IAI" ],
  "entityId" : "001D000000I8mImIAJ",
  "errors" : null,
  "instanceId" : "04gD0000000Cvm5IAC",
  "instanceStatus" : "Pending",
  "newWorkitemIds" : [ "04iD0000000Cw6SIAS" ],
  "success" : true } ]
```

Approve a Record

Use the [Process Approvals](#) resource to approve a record or a collection of records. Each call takes an array of requests. The current user must be an assigned approver.

Example usage

```
curl https://na1.salesforce.com/services/data/v30.0/process/approvals/ -H "Authorization: Bearer token" -H "Content-Type: application/json" -d @approve.json"
```

Example request body `approve.json` file

```
{
  "requests" : [{
    "actionType" : "Approve",
    "contextId" : "04iD00000000Cw6SIAS",
    "nextApproverIds" : ["005D000000015rY9"],
    "comments" : "this record is approved"}]
}
```

Example JSON response body

```
[ {
  "actorIds" : null,
  "entityId" : "001D0000000I8mImIAJ",
  "errors" : null,
  "instanceId" : "04gD00000000CvmAIAS",
  "instanceStatus" : "Approved",
  "newWorkitemIds" : [ ],
  "success" : true
} ]
```

Reject a Record

Use the [Process Approvals](#) resource to reject a record or a collection of records. Each call takes an array of requests. The current user must be an assigned approver.

Example usage

```
curl https://na1.salesforce.com/services/data/v30.0/process/approvals/ -H "Authorization: Bearer token" -H "Content-Type: application/json" -d @reject.json"
```

Example request body `reject.json` file

```
{
  "requests" : [{
    "actionType" : "Reject",
    "contextId" : "04iD00000000Cw6cIAC",
    "comments" : "This record is rejected."}]
}
```

Example JSON response body

```
[ {
  "actorIds" : null,
  "entityId" : "001D0000000I8mImIAJ",
  "errors" : null,
}
```

```

"instanceId" : "04gD0000000CvmFIAS",
"instanceStatus" : "Rejected",
"newWorkitemIds" : [ ],
"success" : true
} ]

```

Bulk Approvals

Use the [Process Approvals](#) resource to do bulk approvals. You can specify a collection of different Process Approvals requests to have them all executed in bulk.

Example usage

```

curl https://na1.salesforce.com/services/data/v30.0/process/approvals/ -H "Authorization: Bearer token" -H "Content-Type: application/json" -d @bulk.json"

```

Example request body `bulk.json` file

```

{
  "requests" :
  [{
    "actionType" : "Approve",
    "contextId" : "04iD0000000Cw6r",
    "comments" : "approving an account"
  }, {
    "actionType" : "Submit",
    "contextId" : "001D0000000JRWBd",
    "nextApproverIds" : ["005D000000015rY9"],
    "comments" : "submitting an account"
  }, {
    "actionType" : "Submit",
    "contextId" : "003D0000000QBZ08",
    "comments" : "submitting a contact"
  }]
}

```

Example JSON response body

```

[ {
  "actorIds" : null,
  "entityId" : "001D0000000I8mImIAJ",
  "errors" : null,
  "instanceId" : "04gD0000000CvmZIAS",
  "instanceStatus" : "Approved",
  "newWorkitemIds" : [ ],
  "success" : true
}, {
  "actorIds" : null,
  "entityId" : "003D0000000QBZ08IAH",
  "errors" : null,
  "instanceId" : "04gD0000000CvmeIAC",
  "instanceStatus" : "Approved",
  "newWorkitemIds" : [ ],
  "success" : true
}, {
  "actorIds" : [ "005D000000015rY9IAI" ],
  "entityId" : "001D0000000JRWBdIAP",
  "errors" : null,
  "instanceId" : "04gD0000000CvmfIAC",

```

```

"instanceStatus" : "Pending",
"newWorkitemIds" : [ "04iD0000000Cw6wIAC" ],
"success" : true
} ]

```

Get a List of Process Rules

Use the [Process Rules](#) resource to get information about process rules.

Example usage

```

curl https://na1.salesforce.com/services/data/v30.0/process/rules/ -H "Authorization: Bearer token"

```

Example request body

none required

Example JSON response body

```

{
  "rules" : {
    "Account" : [ {
      "actions" : [ {
        "id" : "01VD0000000D2w7",
        "name" : "ApprovalProcessTask",
        "type" : "Task"
      } ],
      "description" : null,
      "id" : "01QD0000000APli",
      "name" : "My Rule",
      "namespacePrefix" : null,
      "object" : "Account"
    } ]
  }
}

```

Get a Particular Process Rule

Use the [Process Rules](#) resource and specify the *SObjectName* and *workflowRuleId* of the rule you want to get the metadata for.

Example usage

```

curl https://na1.salesforce.com/services/data/v30.0/process/rules/Account/01QD0000000APli -H "Authorization: Bearer token"

```

Example request body

none required

Example JSON response body

```

{

```

```

"actions" : [ {
  "id" : "01VD0000000D2w7",
  "name" : "ApprovalProcessTask",
  "type" : "Task"
} ],
"description" : null,
"id" : "01QD0000000APli",
"name" : "My Rule",
"namespacePrefix" : null,
"object" : "Account"
}

```

Trigger Process Rules

Use the [Process Rules](#) resource to trigger process rules. All rules associated with the specified ID will be evaluated, regardless of the evaluation criteria. All IDs must be for records on the same object.

Example usage

```

curl https://na1.salesforce.com/services/data/v30.0/process/rules/ -H "Authorization: Bearer token" -H "Content-Type: application/json" -d @rules.json

```

Example request body `rules.json` file

```

{
  "contextIds" : [
    "001D000000JRWBd",
    "001D000000I8mIm",
    "001D000000I8aaf"
  ]
}

```

Example JSON response body

```

{
  "errors" : null,
  "success" : true
}

```

REST API REFERENCE

Chapter 7

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 Database.com instance you are using, or a [custom domain](#). The domain is returned as part of the Authentication response, or via the Console at **Domain Management** > **My Domain**. For example, to retrieve basic information about a Widget custom object in version 20.0:
`http://na1.salesforce.com/services/data/v20.0/objects/Widget__c/`.

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

Resource Name	URI	Description
Versions	/	Lists summary information about each Database.com 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.
Limits	/vXX.X/limits/	Lists information about limits in your organization.
Describe Global	/vXX.X/objects/	Lists the available objects and their metadata for your organization's data.
SObject Basic Information	/vXX.X/objects/ 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/objects/ SObject /describe/	Completely describes the individual metadata at all levels for the specified object.
SObject Get Deleted	/vXX.X/objects/ SObject /deleted/ ?start= startDateAndTime &end= endDateAndTime	Retrieves the list of individual records that have been deleted within the given timespan for the specified object.
SObject Get Updated	/vXX.X/objects/ SObject /updated/ ?start= startDateAndTime &end= endDateAndTime	Retrieves the list of individual records that have been updated (added or changed) within the given timespan for the specified object.

Resource Name	URI	Description
SObject Rows	/vXX.X/subjects/ <i>SObject</i> / <i>id</i> /	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/subjects/ <i>SObject</i> / <i>fieldName</i> / <i>fieldValue</i>	Creates new records or updates existing records (upserts records) based on the value of a specified external ID field.
SObject ApprovalLayouts	/vXX.X/subjects/ <i>SObjectName</i> /describe/approvalLayouts/	Returns a list of approval layouts for a specified object.
SObject CompactLayouts	/vXX.X/subjects/ <i>Object</i> /describe/compactLayouts/	Returns a list of compact layouts for a specific object.
SObject Layouts	/vXX.X/subjects/global/describe/layouts/ /vXX.X/subjects/ <i>object</i> /describe/layouts/	Returns a list of layouts and descriptions, including for publisher actions.
SObject Quick Actions	/vXX.X/subjects/ <i>object</i> /quickActions/ /vXX.X/subjects/ <i>object</i> /quickActions/{ <i>action name</i> } /vXX.X/subjects/ <i>object</i> /quickActions/{ <i>action name</i> }/describe/ services/data/vXX.X/subjects/ <i>object</i> /quickActions/{ <i>action name</i> }/defaultValues/ vXX.X/subjects/ <i>object</i> /quickActions/{ <i>action name</i> }/defaultValues/{ <i>parent id</i> }	Returns a list of publisher actions and details.
SObject User Password	/vXX.X/subjects/User/ <i>user id</i> /password	Set, reset, or get information about a user password.
Process Approvals	/vXX.X/process/approvals/	Returns a list of all approval processes. Can also be used to submit a particular record if that entity supports an approval process and one has already been defined. Records can be approved and rejected if the current user is an assigned approver.
Process Rules	/vXX.X/process/rules/	Returns a list of all workflow rules. If a rule has actions, the actions will be listed under the rule. Can also be used to trigger all workflow rules that are associated with a specified record. The actions for a rule are only fired if the rule's criteria is met.
Query	/vXX.X/query/?q= <i>soql</i>	Executes the specified SOQL query.
QueryAll	/vXX.X/queryAll/?q= <i>soql</i>	Executes the specified SOQL query. Results can include deleted, merged and archived records.
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.

Resource Name	URI	Description
Search	<code>/vXX.X/search/?s=<i>sos1</i></code>	Executes the specified SOSL search. The search string must be URL-encoded.

Versions

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

URI

/

Formats

JSON, XML

HTTP Method

GET

Authentication

none

Parameters

none

Example

See [List Available REST API Versions](#) on page 28.

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

Limits



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

Lists information about limits in your organization. This resource is available in REST API version 29.0 and later for API users with the “View Setup and Configuration” permission. The resource returns these limits:

- Daily API calls
- Daily Batch Apex and future method executions
- Daily Bulk API calls
- Daily Streaming API events
- Streaming API concurrent clients
- Daily number of single emails sent to external email addresses using Apex or Force.com APIs
- Daily number of mass emails sent to external email addresses using Apex or Force.com APIs

The resource also returns these limits if the API user has the “Manage Users” permission.

- Data storage (MB)
- File storage (MB)

URI

`/vXX.X/limits/`

Formats

JSON, XML

HTTP Method

GET

Authentication

Authorization: Bearer ***token***

Example

See [List Organization Limits](#).

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/objects/`

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 78.

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 a custom object using the GET method, or create a new custom object using the POST method.

URI

/vXX.X/subjects/*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 31.
- For an example of creating a new record using POST, see [Create a Record](#) on page 34.

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 a custom object.

The If-Modified-Since header can be used with this resource, with a date format of `EEE, dd MMM yyyy HH:mm:ss z`. When this header is used, if the object metadata has not changed since the provided date, a 304 Not Modified status code is returned, with no response body.

URI

/vXX.X/subjects/*SObjectName*/describe/

Formats

JSON, XML

HTTP Method

GET

AuthenticationAuthorization: Bearer *token***Parameters**

none required

Example

See [Get Field and Other Metadata for an Object](#) on page 32. For an example that uses the If-Modified-Since HTTP header, see [Get Object Metadata Changes](#) on page 33.

SObject Get Deleted

Retrieves the list of individual records that have been deleted within the given timespan for the specified object. SObject Get Deleted is available in API version 29.0 and later.

This resource is commonly used in data replication applications. Note the following considerations:

- Deleted records are written to a delete log which this resource accesses. A background process that runs every two hours purges records that have been in an organization's delete log for more than two hours if the number of records is above a certain limit. Starting with the oldest records, the process purges delete log entries until the delete log is back below the limit. This is done to protect Database.com from performance issues related to massive delete logs
- Information on deleted records are returned only if the current session user has access to them.
- Results are returned for no more than 15 days previous to the day the call is executed (or earlier if an administrator has purged the Recycle Bin).

See “Data Replication” in the [SOAP API Developer's Guide](#) for additional details on data replication and data replication limits.

URI

/vXX.X/subjects/*SObjectName*/deleted/?start=*startDateAndTime*&end=*endDateAndTime*

Formats

JSON, XML

HTTP Method

GET

AuthenticationAuthorization: Bearer *token***Parameters**

Parameter	Description
start	Starting date/time (Coordinated Universal Time (UTC)—not local— timezone) of the timespan for which to retrieve the data. The API ignores the seconds portion of the specified dateTime value (for example, 12:30:15 is interpreted as 12:30:00 UTC). The date and time should be provided in ISO 8601 format: <i>YYYY-MM-DDThh:mm:ss+hh:mm</i> . The date/time value for start must chronologically precede end. This parameter should be URL-encoded.

Parameter	Description
end	Ending date/time (Coordinated Universal Time (UTC)—not local— timezone) of the timespan for which to retrieve the data. The API ignores the seconds portion of the specified dateTime value (for example, 12:35:15 is interpreted as 12:35:00 UTC). The date and time should be provided in ISO 8601 format: <code>YYYY-MM-DDThh:mm:ss+hh:mm</code> . This parameter should be URL-encoded

Response format

Property	Type	Description
deletedRecords	array	Array of deleted records that satisfy the start and end dates specified in the request. Each entry contains the record ID and the date and time the record was deleted in ISO 8601 format, using Coordinated Universal Time (UTC) timezone.
earliestDateAvailable	String	ISO 8601 format timestamp (Coordinated Universal Time (UTC)—not local— timezone) of the last physically deleted object.
latestDateCovered	String	ISO 8601 format timestamp (Coordinated Universal Time (UTC)—not local— time zone) of the last date covered in the request.

Example

For an example of getting a list of deleted items, see [Get a List of Deleted Records Within a Given Timeframe](#) on page 40.

SObject Get Updated

Retrieves the list of individual records that have been updated (added or changed) within the given timespan for the specified object. SObject Get Updated is available in API version 29.0 and later.

This resource is commonly used in data replication applications. Note the following considerations:

- Results are returned for no more than 30 days previous to the day the call is executed.
- Your client application can replicate any objects to which it has sufficient permissions. For example, to replicate all data for your organization, your client application must be logged in with “View All Data” access rights to the specified object. Similarly, the objects must be within your sharing rules.
- There is a limit of 200,000 IDs returned from this resource. If more than 200,000 IDs would be returned, `EXCEEDED_ID_LIMIT` is returned. You can correct the error by choosing start and end dates that are closer together.

See “Data Replication” in the [SOAP API Developer's Guide](#) for additional details on data replication and data replication limits.

URI

`/vXX.X/objects/SObjectName/updated/?start=startDateAndTime&end=endDateAndTime`

Formats

JSON, XML

HTTP Method

GET

Authentication

Authorization: Bearer **token**

Parameters

Parameter	Description
start	Starting date/time (Coordinated Universal Time (UTC) time zone—not local—timezone) of the timespan for which to retrieve the data. The API ignores the seconds portion of the specified dateTime value (for example, 12:30:15 is interpreted as 12:30:00 UTC). The date and time should be provided in ISO 8601 format: <code>YYYY-MM-DDThh:mm:ss+hh:mm</code> . The date/time value for start must chronologically precede end. This parameter should be URL-encoded
end	Ending date/time (Coordinated Universal Time (UTC) time zone—not local—timezone) of the timespan for which to retrieve the data. The API ignores the seconds portion of the specified dateTime value (for example, 12:35:15 is interpreted as 12:35:00 UTC). The date and time should be provided in ISO 8601 format: <code>YYYY-MM-DDThh:mm:ss+hh:mm</code> . This parameter should be URL-encoded

Response format

Property	Type	Description
ids	array	Array of updated records that satisfy the start and end dates specified in the request. Each entry contains the record ID.
latestDateCovered	String	ISO 8601 format timestamp (Coordinated Universal Time (UTC)—not local—time zone) of the last date covered in the request.

Example

For an example of getting a list of updated deleted items, see [Get a List of Updated Records Within a Given Timeframe](#) on page 41.

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/subjects/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 35.

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/subjects/*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 ApprovalLayouts

Returns a list of approval layouts for a specified object. Specify a particular approval process name to limit the return value to one specific approval layout. This resource is available in REST API version 30.0 and later.

Syntax

URI

To get an approval layout description for a specified object, use
`/vXX.X/objects/SObjectName/describe/approvalLayouts/`

To get an approval layout description for a particular approval process, use
`/vXX.X/objects/SObjectName/describe/approvalLayouts/approvalProcessName`

Formats

JSON, XML

HTTP methods

HEAD, GET

Authentication

Authorization: Bearer **token**

Request parameters

None required

Example

Getting all approval layouts for an sObject

```
curl
https://na1.salesforce.com/services/data/v30.0/objects/Account/describe/approvalLayouts/
-H "Authorization: Bearer token"
```

Example JSON Response body

```
{
  "approvalLayouts" : [ {
    "id" : "04aD00000008Py9IAE",
    "label" : "MyApprovalProcessName",
    "layoutItems" : [...],
    "name" : "MyApprovalProcessName"
  }, {
    "id" : "04aD00000008Q0KIAU",
    "label" : "Process1",
    "layoutItems" : [...],
    "name" : "Process1"
  } ]
}
```

If you haven't defined any approval layouts for an object, the response is `{"approvalLayouts" : []}`.

Getting the approval layout for a particular approval process

```
curl
https://na1.salesforce.com/services/data/v30.0/subjects/Account/describe/approvalLayouts/MyApprovalProcessName
-H "Authorization: Bearer token"
```

Example JSON Response body

```
{
  "approvalLayouts" : [ {
    "id" : "04aD00000008Py9IAE",
    "label" : "MyApprovalProcessName",
    "layoutItems" : [...],
    "name" : "MyApprovalProcessName"
  } ]
}
```

SObject CompactLayouts

Returns a list of compact layouts for a specific object. This resource is available in REST API version 29.0 and later.

Syntax

URI

For a compact layout description for a specific object, use `/vXX.X/subjects/Object/describe/compactLayouts/`

Formats

JSON, XML

HTTP methods

HEAD, GET

Authentication

Authorization: Bearer **token**

Request parameters

None required

Example

Getting compact layouts

```
curl
https://na1.salesforce.com/services/data/v29.0/subjects/Object/describe/compactLayouts/
-H "Authorization: Bearer token"
```

Example JSON Response body

```
{
  "compactLayouts" : [ ],
  "defaultCompactLayoutId" : "0AH000000000000GAA",
  "recordTypeCompactLayoutMappings" : [ {
    "recordTypeId" : "012000000000000AAA",
    "compactLayoutId" : "Compact Layout ID"
  } ]
}
```

```
} ]
}
```

If you haven't defined any compact layouts for an object, the `compactLayoutId` returns as `Null`.

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 or publisher actions.

URI

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

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

Formats

JSON, XML

HTTP Method

HEAD, GET

Authentication

Authorization: Bearer *token*

Parameters

None required

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 or publisher actions.

URI

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

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

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

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

In API version 28.0, to evaluate the default values for an action, use

`vXX.X/subjects/object/quickActions/{action name}/defaultValues/{parent id}`

In API version 29.0 and greater, to evaluate the default values for an action, use
`vXX.X/subjects/object/quickActions/{action name}/defaultValues/{context id}`

This returns the default values specific to the {context id} object.

Formats

JSON, XML

HTTP Method

HEAD, GET, POST

Authentication

Authorization: Bearer *token*

Parameters

None required

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/subjects/User/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 46.

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, Database.com will reset the user password to an auto-generated password, which will be returned in the response.

AppMenu

Returns a list of items in either the Salesforce app drop-down menu or the Salesforce1 navigation menu.

Syntax

URI

To return a list of the Salesforce app drop-down menu items, the URI is: `/vXX.X/appMenu/AppSwitcher/`

To return a list of the Salesforce1 navigation menu items, the URI is: `/vXX.X/appMenu/Salesforce1/`

Available since release

29.0

Formats

JSON, XML

HTTP methods

GET, HEAD

Authentication

Authorization: Bearer *token*

Request body

None

Request parameters

None required

Example

Getting appMenu types

```
curl https://na1.salesforce.com/services/data/v29.0/appMenu/ -H "Authorization: Bearer token"
```

Example response body for `/vXX.X/appMenu/AppSwitcher/`

```
{
  "appMenuItems" : [ {
    "type" : "Tabset",
    "content" : null,
    "icons" : null,
    "colors" : null,
    "label" : "Sales",
    "url" : "/home/home.jsp?tsid=02uxx00000056Sq"
  }, {
    "type" : "Tabset",
    "content" : null,
    "icons" : null,
    "colors" : null,
    "label" : "Call Center",
    "url" : "/home/home.jsp?tsid=02uxx00000056Sr"
  }
]
```

```

    }, {
      "type" : "Tabset",
      "content" : null,
      "icons" : null,
      "colors" : null,
      "label" : "Marketing",
      "url" : "/home/home.jsp?tsid=02uxx00000056St"
    }, {
      "type" : "Tabset",
      "content" : null,
      "icons" : null,
      "colors" : null,
      "label" : "Salesforce Chatter",
      "url" : "/home/home.jsp?tsid=02uxx00000056Su"
    }, {
      "type" : "Tabset",
      "content" : null,
      "icons" : null,
      "colors" : null,
      "label" : "Community",
      "url" : "/home/home.jsp?tsid=02uxx00000056Sw"
    }, {
      "type" : "Tabset",
      "content" : null,
      "icons" : null,
      "colors" : null,
      "label" : "App Launcher",
      "url" : "/app/mgmt/applauncher/appLauncher.apexp?tsid=02uxx00000056Sx"
    } ]
  }
}

```

Example response body for /vxx.x/appMenu/Salesforce1/

```

{
  "appMenuItems" : [ {
    "type" : "Standard.Search",
    "content" : null,
    "icons" : null,
    "colors" : null,
    "label" : "Smart Search Items",
    "url" : "/search"
  }, {
    "type" : "Standard.MyDay",
    "content" : null,
    "icons" : null,
    "colors" : null,
    "label" : "Today",
    "url" : "/myDay"
  }, {
    "type" : "Standard.Tasks",
    "content" : null,
    "icons" : null,
    "colors" : null,
    "label" : "Tasks",
    "url" : "/tasks"
  }, {
    "type" : "Standard.Dashboards",
    "content" : null,
    "icons" : null,
    "colors" : null,
    "label" : "Dashboards",
    "url" : "/dashboards"
  }, {
    "type" : "Tab.flexiPage",
    "content" : "MySampleFlexiPage",
    "icons" : [ {
      "contentType" : "image/png",

```

```

        "width" : 32,
        "height" : 32,
        "theme" : "theme3",
        "url" : "http://myorg.com/img/icon/custom51_100/bell32.png"
    }, {
        "contentType" : "image/png",
        "width" : 16,
        "height" : 16,
        "theme" : "theme3",
        "url" : "http://myorg.com/img/icon/custom51_100/bell16.png"
    }, {
        "contentType" : "image/svg+xml",
        "width" : 0,
        "height" : 0,
        "theme" : "theme4",
        "url" : "http://myorg.com/img/icon/t4/custom/custom53.svg"
    }, {
        "contentType" : "image/png",
        "width" : 60,
        "height" : 60,
        "theme" : "theme4",
        "url" : "http://myorg.com/img/icon/t4/custom/custom53_60.png"
    }, {
        "contentType" : "image/png",
        "width" : 120,
        "height" : 120,
        "theme" : "theme4",
        "url" : "http://myorg.com/img/icon/t4/custom/custom53_120.png"
    } ],
    "colors" : [ {
        "context" : "primary",
        "color" : "FC4F59",
        "theme" : "theme4"
    }, {
        "context" : "primary",
        "color" : "FC4F59",
        "theme" : "theme3"
    } ],
    "label" : "My App Home Page",
    "url" : "/servlet/servlet.Integration?lid=01rxx0000000Vsd&ic=1"
}, {
    "type" : "Tab.apexPage",
    "content" : "/apex/myapexpage",
    "icons" : [ {
        "contentType" : "image/png",
        "width" : 32,
        "height" : 32,
        "theme" : "theme3",
        "url" : "http://myorg.com/img/icon/cash32.png"
    }, {
        "contentType" : "image/png",
        "width" : 16,
        "height" : 16,
        "theme" : "theme3",
        "url" : "http://myorg.com/img/icon/cash16.png"
    }, {
        "contentType" : "image/svg+xml",
        "width" : 0,
        "height" : 0,
        "theme" : "theme4",
        "url" : "http://myorg.com/img/icon/t4/custom/custom41.svg"
    }, {
        "contentType" : "image/png",
        "width" : 60,
        "height" : 60,
        "theme" : "theme4",
        "url" : "http://myorg.com/img/icon/t4/custom/custom41_60.png"
    }, {
        "contentType" : "image/png",

```

```

        "width" : 120,
        "height" : 120,
        "theme" : "theme4",
        "url" : "http://myorg.com/img/icon/t4/custom/custom41_120.png"
    } ],
    "colors" : [ {
        "context" : "primary",
        "color" : "3D8D8D",
        "theme" : "theme4"
    }, {
        "context" : "primary",
        "color" : "3D8D8D",
        "theme" : "theme3"
    } ],
    "label" : "label",
    "url" : "/servlet/servlet.Integration?lid=01rxx0000000Vyb&ic=1"
} ]
}

```

FlexiPage

Returns a list of Flexible Pages and their details. Information returned includes Flexible Page regions, the components within each region, and each component's properties, as well as any associated QuickActions. This resource is available in API version 29.0 and later.

Syntax

URI

To return all the details of a Flexible Page, use `/vXX.X/flexiPage/ID of Flexible page`.

Formats

JSON, XML

HTTP methods

HEAD, GET

Authentication

Authorization: Bearer *token*

Parameters

None required

Example

Getting root Flexible Page resource

```
curl https://na1.salesforce.com/services/data/v29.0/flexiPage/ -H "Authorization: Bearer token"
```

Getting a Flexible Page whose ID is 0M0xx0000000001CAA

```
curl https://na1.salesforce.com/services/data/v29.0/flexiPage/0M0xx0000000001CAA -H "Authorization: Bearer token"
```

Example request body for /vXX.X/flexiPage/

none required

Example response body for /vXX.X/flexiPage/

```
{
  "urls" : {
    "flexiPage" : "/services/data/v29.0/flexiPage",
    "rowTemplate" : "/services/data/v29.0/flexiPage/{ID}"
  }
}
```

Example request body for /vXX.X/flexiPage/{ID of FlexiPage}

none required

Example response body for /vXX.X/flexiPage/{ID of FlexiPage}

Note: This code example contains quickActionList information. To find out more about quick actions—also known as publisher actions—in the REST API, see [Quick Actions](#) and [Object Quick Actions](#).

```
{
  "name": "DeveloperNameOfFlexiPage",
  "id": "0M0xx0000000001CAA",
  "label": "FlexiPage Label",
  "quickActionList": {
    "quickActionListItems": [
      {
        "type": "Post",
        "label": "Post",
        "quickActionName": "FeedItem.TextPost",
        "targetSobjectType": null,
        "iconUrl": null,
        "miniIconUrl": null
      },
      {
        "type": "Create",
        "label": "testFlexiQuickAction",
        "quickActionName": "flexiAction",
        "targetSobjectType": "Contact",
        "iconUrl": "http://{SALESFORCE-APPSERVER-DOMAIN}/img/icon/contacts32.png",
        "miniIconUrl":
"http://{SALESFORCE-APPSERVER-DOMAIN}/img/icon/contacts16.png"
      }
    ]
  },
  "regions": [
    {
      "name": "main",
      "components": [
        {
          "properties": [
            {
              "name": "entityName",
              "value": "Account"
            },
            {
              "name": "filterName",
              "value": "MyAccounts"
            }
          ],
          "typeName": "filterList",
          "typeNamespace": "force"
        }
      ]
    }
  ]
}
```

```

    }
  ]
}

```

In the code sample above:

- `name`—the name of the region
- `components`—an array of Aura components in the region
- `properties`—an array of properties for the component
- `typeName`—the name of the Aura component
- `typeNamespace`—the namespace of the Aura component

Process Approvals

Returns a list of all approval processes. Can also be used to submit a particular record if that entity supports an approval process and one has already been defined. Records can be approved and rejected if the current user is an assigned approver. When using a POST request to do bulk approvals, the requests that succeed are committed and the requests that don't succeed send back an error.

Syntax

URI

To return a list of the approvals, the URI is: `/vXX.X/process/approvals/`

Available since release

30.0

Formats

JSON, XML

HTTP methods

GET, HEAD, POST

Authentication

Authorization: Bearer *token*

Request parameters

None required

Request body

The request body contains an array of process requests that contain the following information:

Name	Type	Description
<code>actionType</code>	string	Represents the kind of action to take: <i>Submit</i> , <i>Approve</i> , or <i>Reject</i> .
<code>contextId</code>	ID	The ID of the item that is being acted upon.
<code>comments</code>	string	The comment to add to the history step associated with this request.
<code>nextApproverIds</code>	ID[]	If the process requires specification of the next approval, the ID of the user to be assigned the next request.

Response body

The response contains an array of process results that contain the following information:

Name	Type	Description
actorIds	ID[]	IDs of the users who are currently assigned to this approval step.
entityId	ID	The object being processed.
errors	Error[]	The set of errors returned if the request failed.
instanceId	ID	The ID of the ProcessInstance associated with the object submitted for processing.
instanceStatus	string	The status of the current process instance (not an individual object but the entire process instance). The valid values are “Approved,” “Rejected,” “Removed,” or “Pending.”
newWorkItemIds	ID[]	Case-insensitive IDs that point to ProcessInstanceWorkitem items (the set of pending approval requests)
success	boolean	true if processing or approval completed successfully.

Examples

- See [Get a List of All Approval Processes](#).
- See [Submit a Record for Approval](#).
- See [Approve a Record](#).
- See [Reject a Record](#).
- See [Bulk Approvals](#).

Process Rules

Returns a list of all workflow rules. If a rule has actions, the actions will be listed under the rule. Can also be used to trigger all workflow rules that are associated with a specified record. The actions for a rule are only fired if the rule’s criteria is met. When using a POST request, if anything fails, the whole transaction is rolled back.

Cross-object workflow rules cannot be invoked using the REST API. When doing POST method invocation, the `contextID` in the request body must use the 15-character ID.

Syntax

URI

To get a list of the workflow rules or to trigger one or more workflow rules, the URI is: `/vXX.X/process/rules/`

To get the rules for a particular object: `/vXX.X/process/rules/SObjectName`

To get the metadata for a particular rule: `/vXX.X/process/rules/SObjectName/workflowRuleId`

Available since release

30.0

Formats

JSON, XML

HTTP methods

HEAD, GET, POST

AuthenticationAuthorization: Bearer *token***Request parameters**

None required

Request body

The request body contains an array of context IDs:

Name	Type	Description
contextId	ID	The ID of the item that is being acted upon.

Examples

- See [Get a List of Process Rules](#).
- See [Get a Particular Process Rule](#).
- See [Trigger Process Rules](#).

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 in the `nextRecordsUrl` field of the response. The identifier can be used in an additional request to retrieve the next batch.

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

For retrieving query performance feedback without executing the query:

`/vXX.X/query/?explain=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

AuthenticationAuthorization: Bearer *token*

Parameters

Parameter	Description
<code>q</code>	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”
<code>explain</code>	<p>A SOQL query to get performance feedback on. Use <code>explain</code> instead of <code>q</code> to get a response that details how Database.com will process your query. You can use this feedback to further optimize your queries.</p> <p>The <code>explain</code> parameter is available in API version 30.0 and later.</p> <div>  <p>Note: Using <code>explain</code> with the REST API query resource is a pilot feature. There is no support associated with this pilot feature. For more information, contact salesforce.com, inc.</p> </div>

Response body

For a query using the `q` parameter, the response contains an array of query result records. For a query using the `explain` parameter, the response contains one or more query plans that can be used to execute the query, sorted from most optimal to least optimal. Each plan has the following information:

Name	Type	Description
<code>cardinality</code>	number	The estimated number of records the query would return, based on index fields, if any.
<code>fields</code>	string[]	The index fields used for the query, if the leading operation type is <code>Index</code> , otherwise null.
<code>leadingOperationType</code>	string	<p>The primary operation type that will be used to optimize the query. This can be one of these values:</p> <ul style="list-style-type: none"> • <code>Index</code>—The query will use an index on the query object. • <code>Other</code>—The query will use optimizations internal to Database.com. • <code>Sharing</code>—The query will use an index based on the user’s sharing rules. If there are sharing rules that limit which records are visible to the current user, those rules can be used to optimize the query. • <code>TableScan</code>—The query will scan all records for the query object, and won’t use an index.
<code>relativeCost</code>	number	The cost of this query compared to the SOQL selective query threshold. A value greater than 1.0 means the query won’t be selective. See “More Efficient SOQL Queries” in the <i>Apex Code Developer’s Guide</i> for more information on selective queries.
<code>subjectCardinality</code>	number	The approximate count of all records in your organization for the query object.
<code>subjectType</code>	string	The name of the query object, such as <code>Merchandise__c</code> .

Example

For an example of making a query and retrieving additional query results using the query identifier, see [Execute a SOQL Query](#) on page 42.

For an example using the `explain` parameter to get feedback on a query, see [Get Feedback on Query Performance](#) on page 44.

For more information on SOQL see the [Database.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*.

QueryAll

Executes the specified SOQL query. Unlike the `Query` resource, `QueryAll` will return records that have been deleted because of a merge or delete. `QueryAll` will also return information about archived Task and Event records. `QueryAll` is available in API version 29.0 and later.

If the query results are too large, the response contains the first batch of results and a query identifier in the `nextRecordsUrl` field of the response. The identifier can be used in an additional request to retrieve the next batch. Note that even though `nextRecordsUrl` has `query` in the URL, it will still provide remaining results from the initial `QueryAll` request. The remaining results will include deleted records that matched the initial query.

URI

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

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

`/vXX.X/queryAll/query identifier`

Formats

JSON, XML

HTTP Method

GET

Authentication

Authorization: Bearer **token**

Parameters

Parameter	Description
<code>q</code>	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 that includes deleted items, see [Execute a SOQL Query that Includes Deleted Items](#) on page 43
- For an example of a query that retrieves additional results using the query identifier, see [Retrieving the Remaining SOQL Query Results](#) on page 44

For more information on SOQL see the [Database.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 or publisher 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"
```

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 45.

For more information on SOSL see the [Database.com SOQL and SOSL Reference](#).

Headers

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

- [Limit Info Header](#)

Limit Info Header

Header Field Name and Values

The Limit Info header is a response header that's returned from each call to the REST API. This header returns limit information for the organization. Use this header to monitor your API limits as you make calls against the organization.

Field name

Sforce-Limit-Info

Field values

- `api-usage`—Specifies the API usage for the organization against which the call was made in the format `nn/nnnn`. The first number is the number of API calls used, and the second number is the API limit for the organization.

Example

Sforce-Limit-Info: api-usage=14/5000

This is an example of a response to a REST request for a Merchandise record.

```
HTTP/1.1 200 OK
Date: Mon, 20 May 2013 22:21:46 GMT
Sforce-Limit-Info: api-usage=18/5000
Last-Modified: Mon, 20 May 2013 20:49:32 GMT
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked

{
  "attributes" : {
    "type" : "Merchandise__c",
    "url" : "/services/data/v30.0/subjects/Merchandise__c/a00D0000008pQSNIA2"
  },
  "Id" : "a00D0000008pQSNIA2",
  "OwnerId" : "005D0000001QX8WIAW",
  "IsDeleted" : false,
  "Name" : "Phone Case - iPhone 4/4S",
  "CreatedDate" : "2013-05-20T20:49:32.000+0000",
  "CreatedById" : "005D0000001QX8WIAW",
  "LastModifiedDate" : "2013-05-20T20:49:32.000+0000",
  "LastModifiedById" : "005D0000001QX8WIAW",
```

```

"SystemModstamp" : "2013-05-20T20:49:32.000+0000",
"LastActivityDate" : null,
"LastViewedDate" : "2013-05-20T22:19:56.000+0000",
"LastReferencedDate" : "2013-05-20T22:19:56.000+0000",
"Description__c" : "Phone Case for iPhone 4/4S",
"Price__c" : 16.99,
"Stock_Price__c" : 12.99,
"Total_Inventory__c" : 108.0
}

```

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.
304	The request content has not changed since a specified date and time. The date and time is provided in a <code>If-Modified-Since</code> header. See Get Object Metadata Changes for an example.
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 <code>message</code> and <code>errorCode</code> .
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" : [ ],
  "message" : "malformed id a01900K0001pPuOAAU",

```

```
"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

A

AppMenu [66](#)
 ApprovalLayouts [62](#)
 Approvals [71](#)
 Authentication
 Additional resources [16](#)
 OAuth [4–5](#), [9](#), [12](#), [14](#)
 OAuth endpoints [5](#)
 Remote access applications [4](#)

B

base URI [2](#)
 Bulk approval [50](#)

C

CompactLayouts [63](#)
 Compression
 deflate [3](#)
 gzip [3](#)
 Create [34](#)
 cURL [3](#)

D

date-time [2](#)
 Delete record [35](#)
 Describe Global [30](#), [56](#)

E

Error responses [78](#)

F

Field values
 retrieving values [36](#)
 FlexiPage [69](#)

H

Headers
 If-Modified-Since [33](#)
 Limit Info [77](#)

I

If-Modified-Since Header [33](#)

J

JSON [2](#)

L

Layouts [64](#)
 Limit Info Header [77](#)
 Limits [29](#), [56](#)
 List REST resources [30](#)

O

OAuth
 Additional resources [16](#)
 OAuth 2.0 [2](#)
 Refresh token [14](#)
 User-agent OAuth flow [9](#)
 Username-password OAuth flow [12](#)
 Web server OAuth flow [5](#)
 Object metadata retrieval [31](#)

P

Password management [46](#), [65](#)
 PATCH
 creating records with [34](#)
 Process [71–72](#)
 Process approvals [47–48](#)
 Process rule metadata [51](#)
 Process rules [51–52](#)
 Publisher Quick Actions
 actions [64](#), [76](#)
 Layouts [64](#)
 QuickActions [64](#), [76](#)

Q

Query
 explain parameter [44](#)
 Query that includes deleted items [43](#)
 QueryAll [43](#), [75](#)
 QuickActions [64](#), [76](#)

R

Reject approval [49](#)
 Resource list by version [55](#)
 Resources
 SOject upsert [61](#)
 upsert [37](#)
 REST
 architecture [2](#)
 cache [2](#)
 Examples for approval processes and process rules [47](#)
 Examples for getting object metadata [31](#)
 Examples for getting organization information [28](#)
 Examples for managing user passwords [45](#)

REST (*continued*)

- Examples for searching and queries [41](#)

- Examples for working with records [34](#)

- Examples of using resources [27](#)

- gateway [2](#)

- proxy server [2](#)

- resources [2](#)

- stateless [2](#)

REST API [1](#)

REST resources

- list of REST resources [53](#)

REST resources list [30](#)

- Retrieve object metadata [31](#)

- Retrieving field values [36](#)

- Retrieving records using external IDs [36](#)

S

- Search [45](#), [76](#)

SObject

- ApprovalLayouts [62](#)

- CompactLayouts [63](#)

- QuickActions [64](#)

- SObject Basic Information [57](#)

- SObject Describe [32–33](#), [57](#)

- SObject Get Deleted [40](#), [58](#)

- SObject Get Updated [41](#), [59](#)

- SObject Row [34](#), [60](#)

- SObject upsert [61](#)

- SObject user

- password [65](#)

- Status codes [78](#)

U

- Upsert [37](#)

- Upsert, sObject [61](#)

V

- Versions [28](#), [55](#)

W

- Workflow rules [72](#)

X

- XML [2](#)