REST API Architecture

REST API follows the standard RESTful principles and characteristics.

Client-server

Client applications are independent from Salesforce REST API, meaning each is managed and updated independently.

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 URIs, 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 HTTPS.

Named resources

All resources are named using a base URI that follows your Lightning Platform endpoint. See REST Resources and Requests for details and examples.

Layered components

Intermediaries, such as proxy servers and gateways, are allowed between the client and the resources.

In addition to the standard RESTful principles, REST API includes other key characteristics in its architecture that are important to understand and consider as you develop your applications.

Authentication

REST API supports OAuth 2.0 (an open protocol to allow secure API authorization). See Authorize Apps with OAuth in *Salesforce Help* for more details.

Support for JSON and XML

JSON requests are supported in UTF-8 and are the default. XML requests are supported in UTF-8 and UTF-16. XML responses are provided in UTF-8. Use the `HTTP ACCEPT` header to specify either JSON or XML.

In versions 57.0 and earlier, it's possible to append `json` or `xml` to the URI. For example, `/Account/001D000000INjVe.json`. We recommend using the `HTTP ACCEPT` header to specify JSON or XML instead.

In versions 58.0 and later, appending JSON or XML to the URI isn't supported.

Compression

Compression reduces bandwidth loads by compressing the messages sent between REST API and your client. REST API supports compression with gzip and deflate, as defined by the HTTP 1.1 specification. See Compression Headers.

Conditional Requests

Response caching is supported by conditional request headers that follow the standards defined by the HTTP 1.1 specification, with a few exceptions. See Conditional Request Headers.

Cross-Origin Resource Sharing

Cross-Origin Resource Sharing (CORS) enables web browsers to request resources from origins other than their own. For example, using CORS, JavaScript code at https://www.example.com could request a resource from https://www.salesforce.com. To access supported Salesforce APIs, Apex REST resources, and Lightning Out from JavaScript code in a web browser, add the origin serving the code to a Salesforce CORS allowlist. See Perform Cross-Origin Requests from Web Browsers.

Salesforce ID Length

Salesforce IDs in response bodies are always 18-character IDs. In request bodies, you can use either 15 character IDs or 18 character IDs.

Method Overriding

To override an HTTP method if you use an HTTP library that doesn't allow overriding or setting an arbitrary HTTP method name, use the request parameter `_HttpMethod`.

```
POST https://instance_name/services/data/v65.0/chatter/
/chatter/users/me/conversations/03MD0000000008KMAQ
?_HttpMethod=PATCH&read=true
```

Note

The `_HttpMethod` parameter is case-sensitive. Use the correct case for all values.

HTTPS

All communication between client and server is over HTTPS.