International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 3, Issue 2, 36-43, 2022
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V3I2P105

Original Article

Salesforce REST API in Action: A Practical and Research-Based Exploration of Integration Solutions

Laxman Vattam
Independent Researcher,Washington, USA.

Abstract *- Establishing seamless interactions between software systems hinges on robust data exchange and actionable communication protocols. The Salesforce REST API offers a streamlined, RESTful web service interface that provides comprehensive access to Salesforce features through HTTP methods. Developers can utilize this API to perform CRUD operations, execute data queries, access metadata, and retrieve system limits. Supporting both JSON and XML, the REST API stands as a flexible tool for integration efforts. This article presents a comprehensive examination of the Salesforce REST API, highlighting its capabilities for enabling seamless integration between Salesforce and external systems. Rooted in both research and real-world implementation experience, the study explores how RESTful principles are applied to perform common operations such as creating, updating, retrieving, and deleting Salesforce records. Beyond foundational concepts, the article delves into advanced integration approaches using Bulk API, Composite REST API, and Composite Graph API each evaluated through the lens of scalability, performance, and transactional control. Through the discussion of real-time scenarios and implementation patterns, this paper offers practical guidance for architects and developers designing API-driven integrations.*

Keywords *- Salesforce, REST API, Bulk API, Composite REST API, Composite Graph API, Composite Tree, Composite Batch, Integration.*

1. Introduction

The Salesforce REST API serves as a gateway for external applications to integrate with Salesforce platforms using standard HTTP operations in JSON or XML formats. Its lightweight architecture and ease of use make it especially suitable for mobile applications and third-party clients. In addition to core REST API functionalities, Salesforce offers Apex REST, which allows for the creation of custom web services using Apex code on the Force.com platform.

Each REST API resource represents a distinct entity or operation ranging from individual records to complex queries accessible through clearly defined URIs and standard HTTP methods such as GET, POST, PATCH, and DELETE. These endpoints facilitate direct interaction with the Salesforce environment for a variety of operational needs.

1.1 Core REST API Operations

A typical REST request comprises four key elements: the resource URI, an HTTP method, request headers, and an optional request payload. Headers carry metadata, while the payload contains relevant data when necessary.
Key use cases include:

- Accessing available API version details.
- Retrieving metadata about standard and custom Salesforce objects.
- Executing SOQL queries or searches.
- Modifying or deleting records.

1.2 Advantages of REST API

The REST API is widely favored for the following reasons:

- It is designed to be synchronous, supporting near real-time operations.
- It handles both JSON and XML data formats.
- OAuth 2.0 is supported for secure authentication and authorization.
- It is ideal for integrations where data volume is relatively moderate.

1.3 Connected App Framework

A Connected App facilitates the secure connection of external applications to Salesforce via APIs. After initial authorization, it provides access tokens that can be used to interact with Salesforce data based on granted permissions.

2. Handling Large Data Sets with Bulk API

When processing large volumes of records, Salesforce recommends the use of Bulk API, which adheres to RESTful conventions and supports asynchronous data processing. This API is optimized for high-throughput scenarios such as data migration or mass updates. Unlike the synchronous nature of REST and SOAP APIs which are efficient for smaller, real-time tasks Bulk API enables submission of jobs that are processed in the background, supporting operations on hundreds of thousands or even millions of records. Salesforce's Data Loader tool provides a user-friendly interface for utilizing the Bulk API through CSV-based uploads. For advanced needs, custom client applications can be developed.

2.1 Use Cases for Standard REST API

The **Standard REST API** is ideal for straightforward, synchronous CRUD operations and metadata access. It's widely used when building integrations with moderate data volumes or when third-party systems need to interact directly with Salesforce records.

1. Customer Self-Service Portal Integration

- Portals or websites retrieve and update customer data using standard REST API endpoints (e.g., Accounts, Contacts, Cases).
- Enables customers to view/update their profiles or create service requests without exposing custom logic.

2. Real-Time Record Creation from External Systems

- ERP, e-commerce, or booking platforms use REST API to push real-time data such as new Orders, Opportunities, or Leads directly into Salesforce.

3. Sales Intelligence Dashboards

- External BI tools or CRM add-ons query Salesforce in real time using SOQL over REST to pull data such as top opportunities, pipeline metrics, or sales rep activities.

4. Marketing System Integration

- REST API is used to extract campaign member or contact data for external email tools, or to ingest engagement metrics (opens/clicks) back into Salesforce.

5. Mobile or Lightweight Web Clients

- Mobile apps built on React Native or Flutter use the REST API to fetch and display key Salesforce data like Tasks, Events, and Cases for logged-in users.

6. Automated Record Cleanup or Validation Jobs

- Scripts or middleware systems (e.g., Mulesoft or Azure Logic Apps) query Salesforce for outdated or incomplete records, validate them, and perform updates via PATCH or DELETE.

2.2 Supported Operations via REST API

1. Record Manipulation

- **Create**: POST to the sObject Basic Information endpoint with required fields.
- **Update**: PATCH to a specific record's URI with updated data.
- **Delete**: DELETE using the record's ID.

2. Record Retrieval

- Use GET to obtain specific field values from standard or external objects.
- Support exists for retrieving data using both Salesforce IDs and external IDs.

3. Upsert Operations

- Using external IDs, records can be inserted or updated via POST or PATCH.

4. Relationship Traversal

- REST API enables relationship navigation through friendly URLs, simplifying access to related records without manual ID lookups.

5. Auditing and Change Tracking

- Use sObject Get Deleted and sObject Get Updated resources to track data changes over defined time intervals.

2.3 Practical Use Cases for Apex REST API

Apex REST API is ideal when you need **custom business logic**, **fine-grained control**, or **tailored responses**. These use cases typically involve exposing custom endpoints.

1. Custom Lead Ingestion API

- External systems (like a website or third-party CRM) submit leads to Salesforce.
- Apex REST API applies validation logic, maps custom fields, and triggers downstream processes like campaign assignment or scoring.

2. Eligibility Check Service

- Real-time API to check customer eligibility for a product (e.g., loan or insurance) by running business rules in Apex.
- Response includes approval status, reason codes, and recommendations.

3. Custom Quote Generation

- An external system calls an Apex REST API to calculate a quote.
- Apex logic fetches rate cards, applies discounts, and returns a quote summary in real time.

4. Mobile App Integration

- A mobile app uses a lightweight Apex REST API to fetch user-specific data (e.g., active tasks, opportunities, notifications) with custom filtering and formatting.

5. Third-Party System Callbacks

- External systems send callbacks to Salesforce (e.g., payment confirmations or delivery updates), and Apex REST processes and updates corresponding records.

3. Composite REST API

Composite REST API enables the execution of multiple interrelated API operations in a single request. This batch-style processing improves performance and reduces the overhead of multiple HTTP requests.

For instance, a single composite request can:

- Create an Account.
- Use its response to generate a related Contact.
- Query data related to the Account Owner.
- Retrieve metadata conditionally based on modification date.

These subrequests are defined in a composite JSON file and executed in a predefined order using reference IDs.

Execute dependent requests in a single API call

https://MyDomainName.my.salesforce.com/services/data/v62.0/composite/ -H "Authorization: Bearer token -H "Content-Type: application/json" -d "@composite.json"

```
{
   "allOrNone":true,
   "compositeRequest":[
      {
         "method":"POST",
         "url":"/services/data/v51.0/sobjects/Account",
         "referenceId":"MyAccount",
         "body":{
            "Name":"Bear technologies",
            "BillingStreet":"Main street",
            "BillingCity":"San Mateo",
            "BillingStateCode":"CA",
            "Industry":"Consulting"
         }
      },
      {
         "method":"GET",
         "referenceId":"MyAccountInfo",
         "url":"/services/data/v51.0/sobjects/Account/@{MyAccount.id}"
      },
      {
         "method":"POST",
         "referenceId":"MyContact",
         "url":"/services/data/v51.0/sobjects/Contact",
         "body":{
            "lastname":"John Doe",
            "Title":"CTO of @{MyAccountInfo.Name}",
            "MailingStreet":"@{MyAccountInfo.BillingStreet}",
            "MailingCity":"@{MyAccountInfo.BillingAddress.city}",
            "MailingState":"@{MyAccountInfo.BillingState}",
            "AccountId":"@{MyAccountInfo.Id}",
            "Email":"Mike@BeatTech.com",
            "Phone":"9492345678"
         }
      },
      {
         "method":"GET",
         "referenceId":"NewAccountOwner",
         "url":"/services/data/v51.0/sobjects/User/@{NewAccountInfo.OwnerId}?fields=Name,companyName,Title,City,State"
      },
      {
         "method":"GET",
         "referenceId":"AccountMetadata",
         "url":"/services/data/v51.0/sobjects/Account/describe",
         "httpHeaders":{
            "If-Modified-Since":"Tue, 31 May 20XX 18:13:37 GMT"
         }
      }
   ]
}
```

Response body after successfully executing the composite request

```
{
    "compositeResponse" : [{
        "body" : {
            "id" : "001E00000033JJlIAM",
            "success" : true,
            "errors" : [ ]
        },
        "httpHeaders" : {
          "Location" : "/services/data/v51.0/sobjects/Account/001E00000033JJlIAM"
        },
        "httpStatusCode" : 201,
        "referenceId" : "MyAccount"
    },{
        "body" : {
            all the account data
        },
        "httpHeaders" : {
            "ETag" : "\"Jbjuzw7dbhaEG3fd90kJbx6A0ow=\"",
            "Last-Modified" : "Fri, 22 Jun 20XX 20:19:37 GMT"
        },
        "httpStatusCode" : 200,
        "referenceId" : "MyAccountInfo"
    },{
        "body" : {
            "id" : "003R00000056RJKIA2",
            "success" : true,
            "errors" : [ ]
        },
        "httpHeaders" : {
            "Location" : "/services/data/v51.0/sobjects/Contact/003R00000056RJKIA2"
        },
        "httpStatusCode" : 201,
        "referenceId" : "MyContact"
    },{
    },{
        "body" : null,
        "httpHeaders" : {
            "ETag" : "\"f2293620\"",
            "Last-Modified" : "Fri, 22 Jul 2016 18:45:56 GMT"
        },
        "httpStatusCode" : 304,
        "referenceId" : "AccountMetadata"
    }]
}
```


3.1 Practical Use Cases for Composite REST API

Composite REST API is powerful for minimizing API calls and handling dependent operations in a single request. Useful for client-side batching or orchestrating object relationships.

1. Create Account and Contact Together

- A single API call creates an Account and a related Contact using referenceId, reducing round-trips and ensuring consistency.

2. Quote Update + Validation in One Call

- A composite request updates a Quote and retrieves the latest billing address, status, and validation result—ideal for a "Review & Submit" page in client apps.

3. Service Appointment Booking

- A front-end app sends a composite request to:
    - Create a Case
    - Schedule a Service Appointment
    - Link to a Technician
    - Return confirmation details in one shot

4. Batch Update of Contact Preferences

- A composite batch request updates email, SMS, and phone preferences for 100+ Contacts without hitting governor limits for individual calls.

5. Dynamic Form Save

- A dynamic multi-step form in a web app collects Account, Contact, and Custom Object data.
- All records are created in a single composite call, reducing chances of partial save and improving performance.

Update a record and query its name and billing postal code in a single request:
`https://MyDomainName.my.salesforce.com/services/data/v63.0/composite/batch/ -H "Authorization: Bearer token -H "Content-Type: application/json" -d "@batch.json"`

Request body:

```
{
"batchRequests" : [
    {
    "method" : "PATCH",
    "url" : "v51.0/sobjects/account/001H000000K0fHYTAZ",
    "richInput" : {"Name" : "NewName"}
    },{
    "method" : "GET",
    "url" : "v51.0/sobjects/account/001H000000K0fHYTAZ
      ?fields=Name,BillingPostalCode"
    }]
}
```

Response body after successfully executing the subrequests:

```
{
  "hasErrors": false,
  "results": [
    {
      "statusCode": 204,
      "result": null
    },
    {
      "statusCode": 200,
      "result": {
        "attributes": {
          "type": "Account",
          "url": "/services/data/v51.0/sobjects/Account
            /001H000000K0fHYTAZ"
        },
        "Name": "NewName",
        "BillingPostalCode": "94105",
        "Id": "001H000000K0fHYTAZ"
      }
    }
  ]
}
```

4. Using Composite Graphs

For more complex interactions, Composite Graph API introduces a directed graph structure where each node represents a subrequest. These nodes can reference each other, allowing dependencies between operations.
Key features include:

- Support for up to 500 subrequests, compared to 25 in standard composite API.
- All-or-none execution mode, ensuring atomicity.
- Support for endpoints including standard CRUD operations and actions using external IDs.

Each node in the graph can send and receive data to/from other nodes, making this method particularly useful for batch operations with logical dependencies.

In the context of composite graph operations, the nodes are composite subrequests. For example, a node can be a composite subrequest like this:

```
{
   "url" : "/services/data/v51.0/sobjects/Account/",
   "body" : {
      "name" : "Bear Tech"
   },
   "method" : "POST",
   "referenceId" : "reference_id_acc"
}
```

Composite graph requests support only these URLs:

URL	Supported HTTP Methods
/services/data/vXX.X/sobjects/sObject	POST
/services/data/vXX.X/sobjects/sObject/id	DELETE, GET, PATCH
/services/data/vXX.X/sobjects/sObject/customFieldName/externalId	DELETE, GET, PATCH, POST

A composite graph can be *directed* meaning that some nodes use information from other nodes. For example, a node that creates a Contact can use the ID of an Account node to link the Contact with the Account.

```
{
  "graphs": [
    {
      "graphId": "graphid1",
      "compositeRequest": [
        {
          "body": {
            "name": "Bear Tech"
          },
          "method": "POST",
          "referenceId": "ref_id_acc",
          "url": "/services/data/v51.0/sobjects/Account/"
        },
        {
          "body": {
            "FirstName": "Krish",
            "LastName": "Simon",
            "AccountId": "@{reference_id_acc}"
          },
          "method": "POST",
          "referenceId": "ref_id_cont",
          "url": "/services/data/v51.0/sobjects/Contact/"
        }
      ]
    }
  ]
}
```

5. Conclusion

The Salesforce REST API suite augmented by Bulk, Composite, and Graph variants provides a powerful and flexible set of tools for integrating Salesforce with external systems. Its ability to handle both simple and complex data operations makes it ideal for organizations seeking robust integration solutions. As enterprises continue to prioritize data connectivity and agility, leveraging RESTful APIs will be essential for maintaining scalable and maintainable application ecosystems.

References

[1] Salesforce, Inc. "Salesforce Developers REST API Developer Guide." Salesforce Developers. https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/

[2] Salesforce, Inc. "Apex Developer Guide: Exposing Apex Classes as REST Web Services." Salesforce Developers. https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_rest_intro.htm

[3] MuleSoft, LLC. "Best Practices for Salesforce Integration." MuleSoft Whitepaper. https://www.mulesoft.com/resources/esb/salesforce-integration-best-practices

[4] Salesforce, Inc. "Composite Resources." Salesforce Developers. https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/resources_composite.htm

[5] Vinoski, S. "RESTful Web Services: Principles, Patterns, and Practices." IEEE Internet Computing, vol. 13, no. 2, pp. 90–92, 2009. https://doi.org/10.1109/MIC.2009.34

[6] Salesforce, Inc. "Bulk API Developer Guide." Salesforce Developers. https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/

[7] Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Doctoral dissertation, University of California, Irvine). https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[8] Microsoft. (2019). Choosing Between REST APIs and SOAP APIs. Microsoft Architecture Center. https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design

[9] Salesforce, Inc. "Connected Apps." Salesforce Help. https://help.salesforce.com/s/articleView?id=sf.connected_app_overview.htm

[10] S. Nambiar, K. Mehta, and S. Babu, "A Comparative Study of RESTful APIs and SOAP Web Services in Modern Cloud Applications," *International Journal of Computer Applications*, vol. 182, no. 13, pp. 20-26, 2021.

[11] Pautasso, C., Zimmermann, O., & Leymann, F. (2008). RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision. *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*, pp. 805–814. https://doi.org/10.1145/1367497.1367606

[12] Oracle. (2017). *REST vs. SOAP Web Services: Understanding the Differences*. Oracle Technical Paper. https://www.oracle.com/technical-resources/articles/javase/rest.html

[13] Taibi, D., Lenarduzzi, V., & Pahl, C. (2020). Architectural Patterns for Microservices: A Systematic Mapping Study. *Software: Practice and Experience*, 50(1), 1–39. https://doi.org/10.1002/spe.2764