Accelerating Salesforce Development: Leveraging Composite Requests for Scalable Solution

Kalpana Puli

kalpanapuli@gmail.com
Independent Researcher, Texas, USA

Abstract
Salesforce Composite Requests have revolutionized API development by providing a streamlined approach to execute multiple API calls within a single transaction. This method not only optimizes API usage but also ensures data consistency and simplifies workflows for developers. By utilizing this capability, businesses can scale applications efficiently and reduce API costs. This paper explores the principles of Composite Requests, emphasizing their advantages, implementation methodologies, and practical applications in the Salesforce ecosystem. Through real-world examples and advanced use cases, this study demonstrates how Composite Requests enhance application performance and scalability.

Keywords: Salesforce, Composite Requests, API Optimization, Workflow Automation, Application Development, Scalability

I. Introduction

In the modern era of cloud computing, APIs have become a cornerstone for building scalable and integrated applications. Salesforce, as a leading cloud-based platform, offers various APIs to support business automation and integration needs. Among these, Composite Requests stand out for their ability to consolidate multiple REST API calls into a single transaction. Composite Requests address common challenges in API development, such as efficiency, transaction management, and API usage limits. For instance, organizations using Salesforce can handle up to 25 sub requests per transaction, enabling developers to optimize API consumption while maintaining operational integrity. This paper delves into the mechanics, benefits, and real-world applications of Composite Requests, showcasing their significance in building robust and scalable applications.

II. Understanding Composite Requests

Composite Requests in Salesforce's REST API provide a streamlined approach to manage multiple operations within a single API call. By consolidating multiple actions, developers can enhance the efficiency of their interactions with the Salesforce platform. This feature significantly reduces the overhead associated with making multiple discrete API calls, which is particularly beneficial in high-latency environments or for complex workflows that require several actions. Through Composite Requests, users can process various records, including creating, updating, or querying multiple Salesforce objects in one seamless request, ensuring smoother data handling and a reduction in response time.
One of the key advantages of Composite Requests is their support for dynamic data referencing across different sub requests. This capability allows the output of one sub request, such as the creation of a new record, to be used as input for subsequent operations. For example, after creating an account and receiving its unique ID as a response, this ID can be referenced in a follow-up request to create related records, such

as contacts or opportunities. This dynamic referencing eliminates the need for developers to manually handle IDs or make additional calls to retrieve and pass data, simplifying complex workflows and ensuring data integrity across multiple operations.

Moreover, Composite Requests also improve resource utilization by minimizing the number of HTTP requests and maximizing the use of batch processing. Instead of executing multiple, isolated calls that would each require individual server resources and network bandwidth, a Composite Request can send multiple sub requests within one payload, optimizing both server-side and client-side resource usage. By reducing the number of round-trips between the client and server, this feature leads to faster processing times, reduced network congestion, and better overall system performance. This efficiency is crucial for scaling API interactions in larger enterprise environments where resource optimization directly impacts operational costs.

Example Use Case: Creating an order and linking it to a newly created customer account in real-time without additional API calls

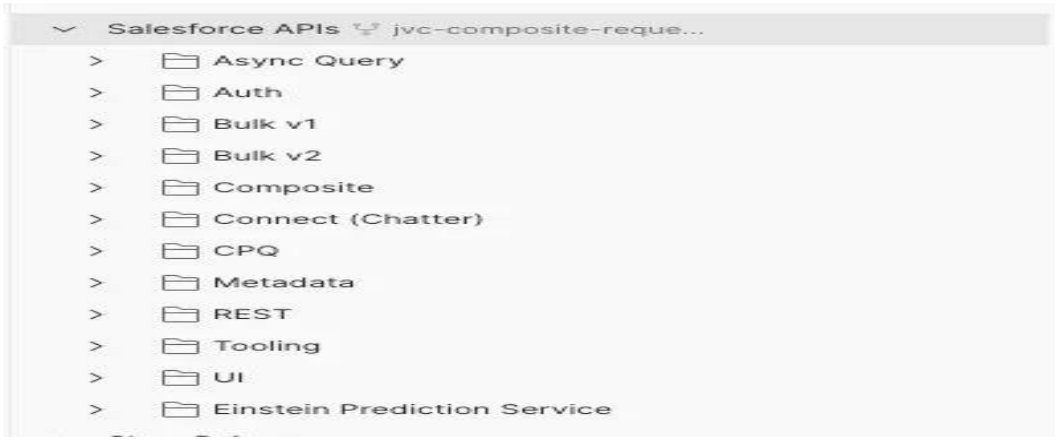


FIG I: API'S [1]

Consider a scenario where a business needs to create an account and multiple associated contacts. Traditionally, this would require separate API calls for each operation. With Composite Requests, this process can be condensed as shown below:

Sample Payload:

```
{
 "compositeRequest": [
  {
   "method":"POST",
   "url":\"/services/data/v52.0/sobjects/Account",
   "referenceId": "refAccount",
   "body": { "Name": "Acme Corporation" }
  },
  {
   "method": "POST",
   "url": "/services/data/v52.0/sobjects/Contact",
   "referenceId": "refContact",
   "body": {
    "FirstName": "Jane",
    "LastName": "Doe",
    "AccountId": "@{refAccount.id}"
```

```
      }
    }
  ]}
```

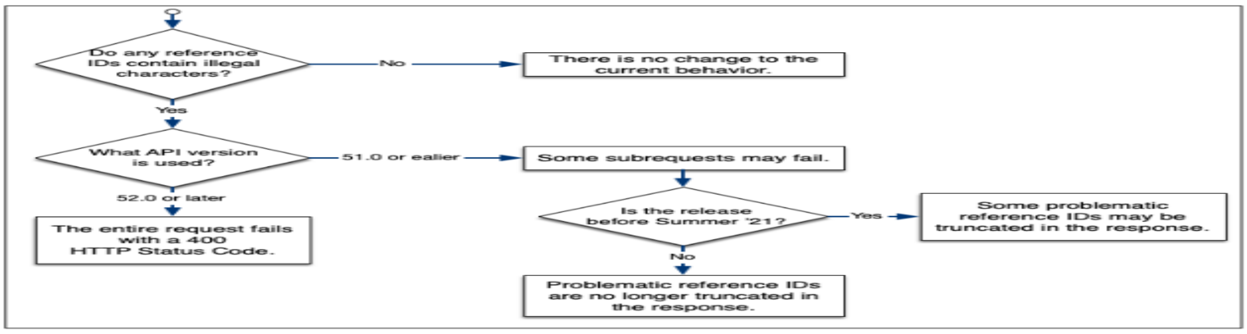
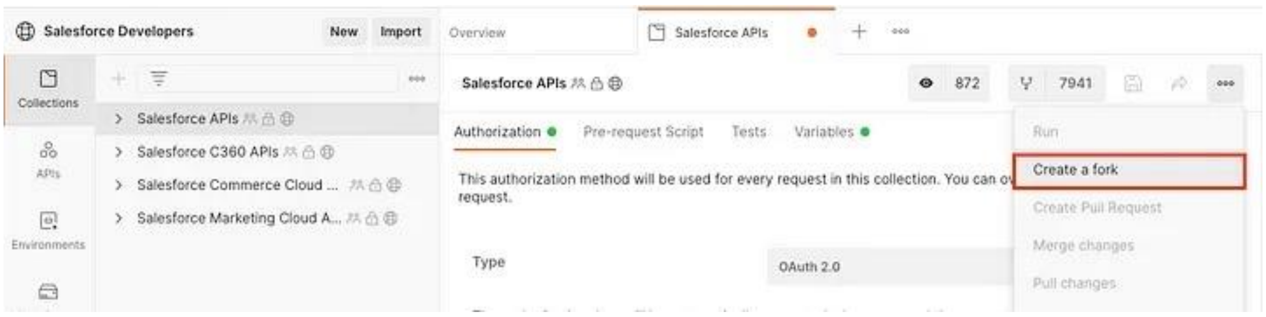

Do any reference IDs contain illegal characters?
No
There is no change to the current behavior.
Yes
What API version is used?
51.0 or ealier
Some subrequests may fail.
52.0 or later
The entire request fails with a 400 HTTP Status Code.
Is the release before Summer '21?
Yes
Some problematic reference IDs may be truncated in the response.
No
Problematic reference IDs are no longer truncated in the response.

FIG II :REST API Developer Guide



This authorization method will be used for every request in this collection. You can o
request.

FIG III : Selection Of API[1]

III. Advantages of Composite Requests

By combining multiple operations, Composite Requests reduce the number of API calls, making them especially beneficial for organizations with strict API usage limits. Developers can streamline workflows and minimize the need for additional code to handle sequential API calls. With transaction management features, Composite Requests ensure that operations are either fully executed or completely rolled back. Capable of handling up to 25 sub requests in one transaction, Composite Requests empower businesses to build scalable solutions.

IV. Implementing Composite Requests

A Salesforce Developer Org serves as an essential environment for testing API functionalities and experimenting with various features offered by Salesforce. It provides a safe, isolated space where developers can create and modify records, test API calls, and explore the full capabilities of the Salesforce platform without impacting production data. By using a Developer Org, developers can work with realistic data structures, integrations, and workflows to simulate real-world use cases and validate their API logic before deploying to a live environment. This ensures that any changes or new features are thoroughly tested and do not affect the integrity of the production system.
Postman or similar API testing tools are invaluable in crafting and testing requests within the Salesforce ecosystem. These tools allow developers to easily construct HTTP requests, define headers, set request bodies, and handle authentication procedures without writing custom code. Postman, for example, provides an intuitive user interface that simplifies the process of building complex requests and inspecting responses. When working with Salesforce APIs, Postman can be used to test Composite Requests, check the responses for accuracy, and troubleshoot potential issues, all while avoiding the need for creating a fully integrated

environment for each test scenario. These tools speed up the development process and make debugging and optimization more efficient.
OAuth 2.0 authentication is a critical component when working with Salesforce APIs, as it ensures secure, authorized access to Salesforce resources. To interact with Salesforce's REST API, developers must first obtain an access token through the OAuth 2.0 flow, which is part of the authentication and authorization process. This token serves as proof of identity and permission, granting access to specific data and services within Salesforce. Once the access token is obtained, it can be used in the headers of API requests to authenticate and ensure that the interaction is secure. The correct setup and handling of OAuth 2.0 are fundamental for ensuring the integrity and security of data when executing API calls, including Composite Requests.

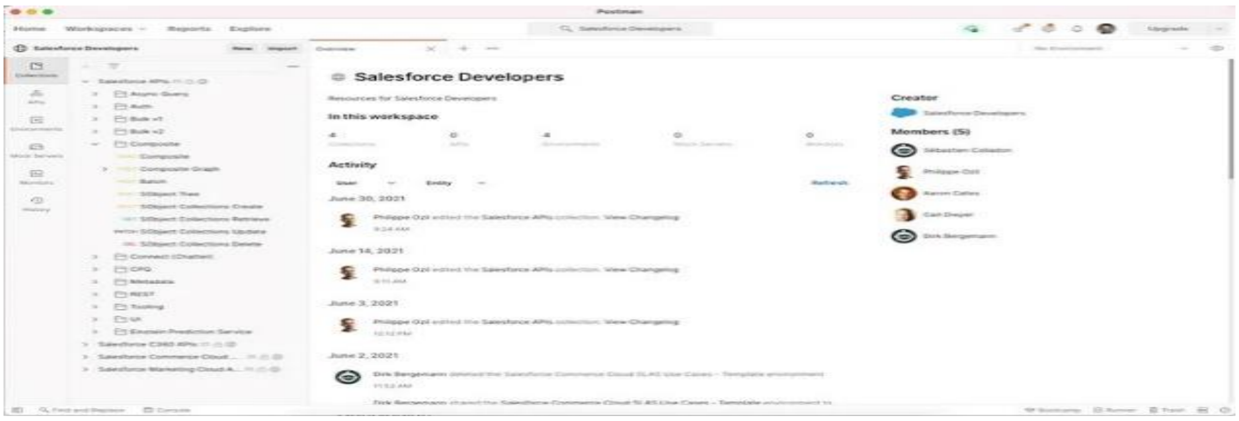
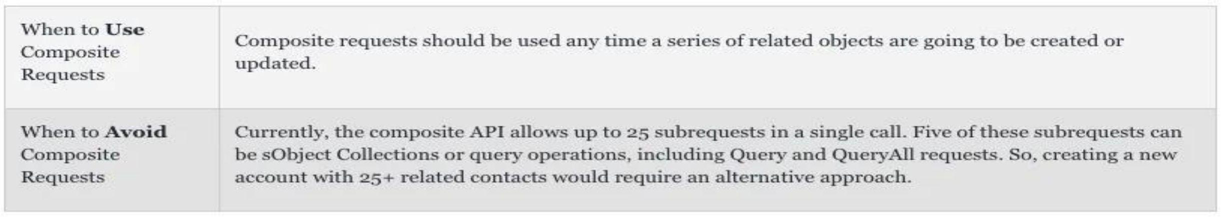


FIG IV: Configuring Postman[1]

V. Real-World Applications

A financial institution streamlined its customer onboarding process by bundling account creation, contact addition, and lead assignment into a single Composite Request. This reduced API consumption by 40% and decreased onboarding time from 3 days to 1 day. An e-commerce platform implemented Composite Requests to automate order processing, inventory updates, and customer notifications. This approach improved order handling efficiency by 30%.



When to **Use** Composite Requests	Composite requests should be used any time a series of related objects are going to be created or updated.
When to **Avoid** Composite Requests	Currently, the composite API allows up to 25 subrequests in a single call. Five of these subrequests can be sObject Collections or query operations, including Query and QueryAll requests. So, creating a new account with 25+ related contacts would require an alternative approach.

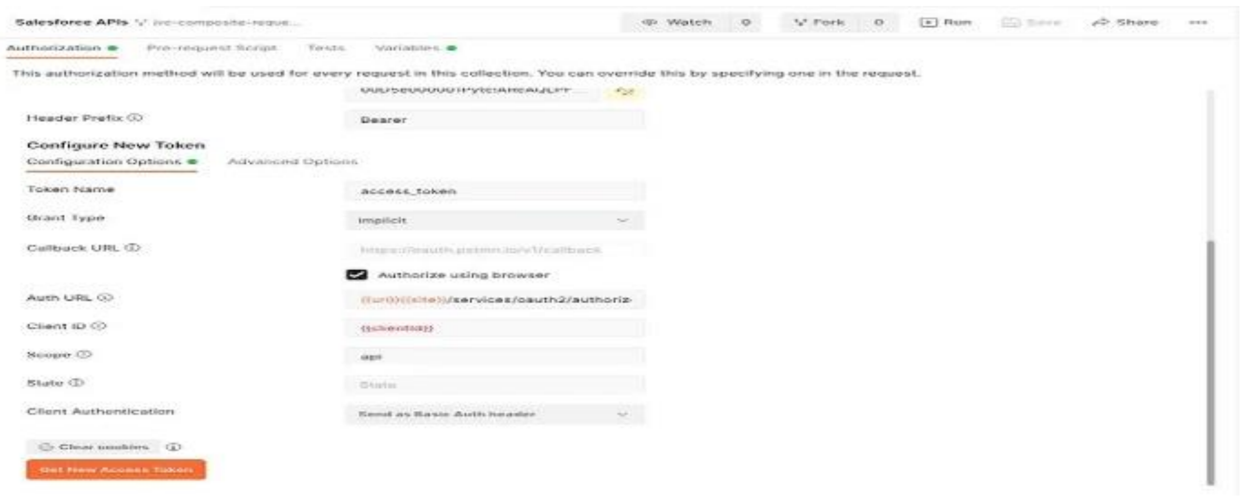
FIG V: Simple Composite Request[1]

VI. Challenges and Considerations

While Salesforce Composite Requests support up to 25 sub requests, managing dependencies between them and debugging issues can become increasingly complex as the number of operations grows. Each sub request may depend on the output of a previous one, and handling these interdependencies requires careful planning to ensure data integrity. Moreover, when the Allor None parameter is set to false, partial failures can occur, where some sub requests may succeed while others fail. Developers must implement robust error-handling mechanisms to capture and manage these partial failures effectively. This can include validating the responses of each sub request, logging errors for later analysis, and providing fallback strategies to ensure the overall process does not break down.

To maintain optimal performance, it is also crucial for developers to regularly evaluate the impact of Composite Requests on application response times. Although Composite Requests are designed to reduce the overhead of multiple API calls, executing many complex sub requests within a single call can still negatively affect performance, especially in large-scale applications or when handling large datasets. Developers should monitor API response times, resource consumption, and potential bottlenecks to ensure that the use of Composite Requests does not compromise the overall user experience. By evaluating performance regularly and optimizing sub request structures, developers can balance functionality with efficiency, ensuring that the system remains responsive even as the complexity of the requests increases.

.


Salesforce APIs / rvc-composite-reque...

This authorization method will be used for every request in this collection. You can override this by specifying one in the request.

Header Prefix: Bearer

Configure New Token

Token Name: access_token
Grant Type: implicit
Callback URL: https://oauth.pstmn.io/v1/callback
Authorize using browser
Auth URL: {{url}}{{site}}/services/oauth2/authoriz
Client ID: {{clientid}}
Scope: api
State: State
Client Authentication: Send as Basic Auth header

FIG VI: Logging[1]

Problems with Conventional API Calls: Increased usage of the API, using allotted limits more quickly.

VIII. A Comparative Study of Conventional API Calls and Composite Requests

A. Conventional API Method Conventional API calls necessitate separate transactions for every activity, which raises the complexity and resource consumption. For instance, establishing an account and linking several contacts may require distinct calls for every entity.
Sequential dependencies must be handled explicitly in code. Increased delay as a result of several HTTP queries. By combining operations, Approach Composite Requests overcome these constraints. As the last example showed, developers gain from more efficient workflows and a smaller API footprint. Composite requests come with built-in tools for managing dependencies through dynamic data referencing (e.g., @{refAccount.id}).Time-sensitive workflows function better when there is less network overhead.

VIII. A Comparative Study of Conventional API Calls and Composite Requests

A. Conventional API Method Conventional API calls necessitate separate transactions for every activity, which raises the complexity and resource consumption. For instance, establishing an account and linking several contacts may require distinct calls for every entity. Increased usage of the API, using allotted limits more quickly. Sequential dependencies must be handled explicitly in code. Increased delay because of several HTTP queries. By combining operations, Approach Composite Requests overcome these constraints. As the last example showed, developers gain from more efficient workflows and a smaller API footprint. Composite requests come with built-in tools for managing dependencies through dynamic data referencing (e.g., @{refAccount.id}).Time-sensitive workflows function better when there is less network overhead. Intermediate client-side processing is not necessary since responses from earlier activities can easily feed later processes.

IX. The Best Ways to Implement Composite Requests

Reduce the payload's size as much as possible to avoid exceeding request size restrictions. Keep your structures brief and steer clear of superfluous fields. Use the Debug Logs in Salesforce to monitor in real time. Make sure every operation is working by validating each Subrequest separately before bundling. Prevent unwanted access by using OAuth tokens with the proper scopes. Change access credentials on a regular basis and keep an eye out for unusual activity in API logs. Keep an eye on API response times using Salesforce Event Monitoring. Examine usage trends for APIs to find areas for improvement.

X. Upcoming Development Trends for APIs

Additional platforms are following suit.to Composite Requests, suggesting a more widespread trend in the industry toward the efficiency of API bundling. For instance, batch processing tools have been added to platforms like as Google Cloud APIs and Microsoft Dynamics 365.AI models are being created to forecast and recommend optimizations for API calls.AI might, for instance, automatically spot chances to combine related tasks into Composite Requests.GraphQL offers an alternative method of retrieving and altering relevant data in a single query, whereas Composite Requests optimize RESTful APIs. By contrasting different technologies, one can learn when to employ each strategy.

XI. Composite Requests Future Prospects

Salesforce's API ecosystem is always changing. Future features could consist of:Higher Limits on Subrequests Permitting a Composite Request to contain more than 25 Subrequeststo make room for more extensive workflows. Improved debugging tools to show performance effects and dependencies. Composite Requests are directly integrated into Salesforce's Flow Builder, making installation easier for non-technical users. Tools for real-time performance analytics that track and dynamically improve Composite Request performance.

XII. Conclusion

Salesforce Composite Requests provide a transformative approach to API development, offering optimized usage, transactional integrity, and simplified workflows. As businesses increasingly rely on API-driven architectures, leveraging Composite Requests can significantly enhance application performance and scalability. Future advancements in Salesforce APIs are expected to further empower developers and organizations in building robust solutions.

References

[1] Miro, “Process,” *Medium*. https://miro.medium.com/v2/resize:fit:640/format:webp/1*XVb4KDIU2FLL2qKYUiXubw.png (Accessed June. 23, 2022).

[2] P.Ozil, “Streamlining development with composite requests,” *Salesforce Developers Blog*, 2023. [Online]. Available: https://developer.salesforce.com/blogs/2022/12/processing-large-amounts-of-data-with-apis-part-2-of-2. (Accessed July. 23, 2022).

[3] Gartner, “API management trends,” 2024. [Online]. Available: https://www.gartner.com/en/documents/5551595. [Accessed: Aug. 15, 2022].

[4] Forrester, “The role of API optimization in modern applications,” 2024. [Online]. Available: https://reprint.forrester.com/reports/the-forrester-wave-tm-api-management-software-q3-2022-8039c3ee/index.html. [Accessed: Sep. 15, 2022].

[5] Salesforce, “Best practices for composite API usage,” [Online]. Available: https://salesforce.com/best-practices-composite-api. [Accessed: Oct. 23, 2022].