Introduction

For making multiple REST API calls for one task and improving the performance of the application, Salesforce is having the composite resources to batch up multiple calls in a single call. We can use the composite resources to make multiple requests using a single REST API call. It will simplify our code, reduce network overhead and improve our application performance.

The Composite Resources

Using REST API composite resources, we can improve our application's performance by minimizing the number of round-trips between the client and the server. Salesforce having three types of composite resources.

composite/batch/: Execute a set of subrequests in a single request. Subrequests are executed independently and information can't be passed between subrequest calls.

composite/tree/: Create one or more sObject trees (a collection of nested, parent-child records with a single root record) in a single request. All tree root records must be the same type.

composite/: Execute a series of REST API subrequests in a single call. The output of one subrequest can be used as input to a subsequent subrequest.

Note: The composite resources shouldn't be used in place of existing APIs that are designed to handle asynchronous uploads or modifications of large batches of records, like the Bulk API.

Composite Batch

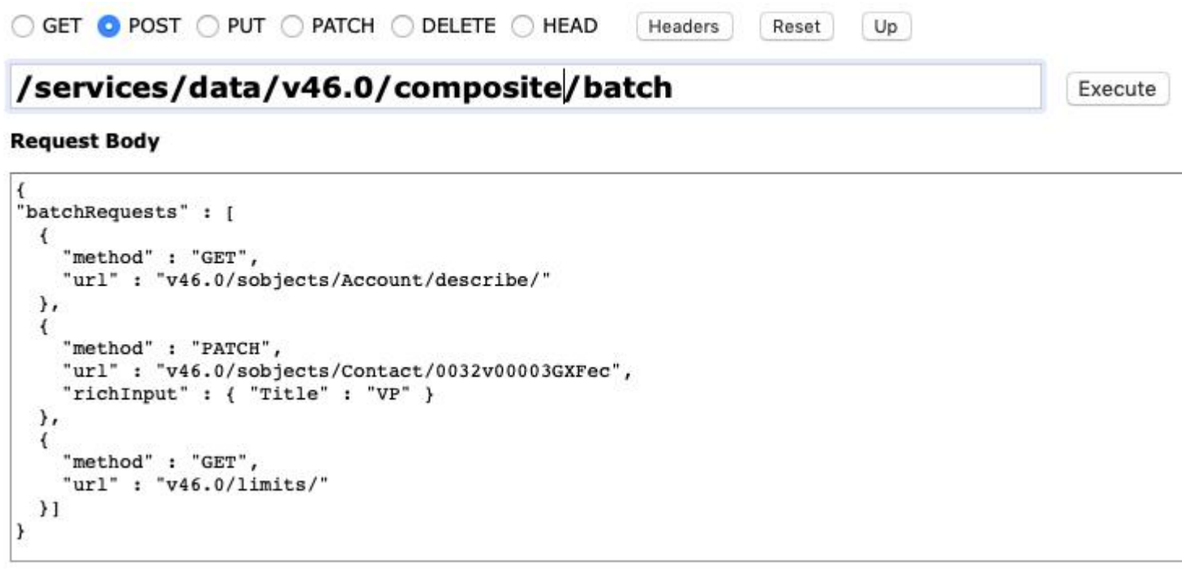
Using the batch resource, we can execute a bunch of independent REST API calls in a single REST API request. Set of requests in a batch are called as subrequests and all subrequests are executed in the context of the same user. Subrequests are can't pass information between each of them. The single response body will contain the statuses and response bodies of the subrequests in the batch. Each subrequest counts against rate limits. Subrequests execute serially in their order specified in the request body. If a subrequest executes successfully, it will commit its data. If a subrequest fails, commits made by previous subrequests are not rolled back. Also If a batch request will not complete within 10 minutes, the batch times out and the remaining subrequests are not executed.

To make a batch resource call, issue a POST to ***instance endpoint*/services/data/*API version*/composite/batch/** with a request body that contains an array of REST API subrequests. For each subrequest in the array, you specify the HTTP method (GET, POST, PATCH, etc), the resource URL, and other optional request attributes as needed.

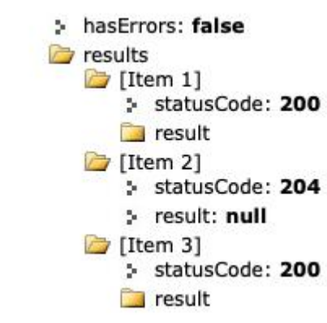
URI	/vXX.X/composite/batch
Formats	JSON, XML
Method	POST
Authentication	Authorization: Bearer token
Request body	The request body contains a batch request collection that includes subrequests to execute.
Response body	Collection of subrequest results. if the errors are having the value true if at least one of the results in the result set is an HTTP status code in the 400 or 500 range; false otherwise.

Example

This example contains a describe call on Account, followed by an update to change a particular Contact's Title field, followed by a request to get the current API limits:



Response body after successfully creating records:



Composite SObject Tree

Use the tree resource to create multiple object records of the same type, along with child records, in a single call. A sObject tree is a collection of nested, parent-child records with a single root record. In the request data, you supply the record hierarchies, required and optional field values, each record's type, and a reference ID for each record.

Upon success, the response contains the IDs of the created records. If an error occurs while creating a record, the entire request fails. In this case, the response contains only the reference ID of the record that caused the error and the error information.

The request can contain the following:

1. Up to a total of 200 records across all trees
2. Up to five records of different types
3. SObject trees up to five levels deep

URI	vXX.X/composite/tree/SObjectName
Formats	JSON, XML
Method	POST
Authentication	Authorization: Bearer token
Request body	Collection of record trees to create. Each tree's root record type must correspond to the sObject specified in the SObject Tree URI.
Response body	Upon success, results contain the reference ID of each requested record and its new record ID. Upon failure, it contains only the reference ID of the record that caused the error, error status code, error message, and fields related to the error. In the case of duplicate reference IDs, results contain one item for each instance of the duplicate ID.

Example

Create Multiple Records

In a single request, we can create up to 200 records. In the request data, we need to supply the required and optional field values for each record, each record's type, and a reference ID for each record, and then use the POST method of the resource. The response body will contain the IDs of the created records if the request is successful. Otherwise, the response contains only the reference ID of the record that caused the error and the error information.

Here's an example request body that uses two single-record SObject trees to create two unrelated Account records, with each Account record having no child records:

Choose an HTTP method to perform on the REST API service URI below:

GET POST PUT PATCH DELETE HEAD Headers Reset Up

/services/data/v46.0/composite/tree/Account Execute

Request Body

```
{
"records" :[
  {
    "attributes" : {"type" : "Account", "referenceId" : "ref1"},
    "name" : " Create Multiple Records Sample Account1",
    "phone" : "9876543210",
    "website" : "www.salesforce1.com"
  },{
    "attributes" : {"type" : "Account", "referenceId" : "ref2"},
    "name" : " Create Multiple Records Sample Account2",
    "phone" : "9876543211",
    "website" : "www.salesforce2.com"
  }]
}
```

Response body after successfully creating records:

```
{
  "hasErrors" : false,
  "results" : [ {
    "referenceId" : "ref1",
    "id" : "0012v00002feELFAA2"
  }, {
    "referenceId" : "ref2",
    "id" : "0012v00002feELGAA2"
  } ]
}
```

Create Nested Records

Using the SObject Tree resource to create nested records that share a root record type. Once the request is processed, the records are created and parents and children are automatically linked by ID. In the request data, you supply the record hierarchies, required and optional field values, each record's type, and a reference ID for each record, and then use the POST method of the resource. The response body will contain the IDs of the created records if the request is successful. Otherwise, the response contains only the reference ID of the record that caused the error and the error information.

Here's an example request body that uses one SObject tree to create a single Account record along with two child Contact records:

GET POST PUT PATCH DELETE HEAD Headers Reset Up

/services/data/v46.0/composite/tree/Account Execute

Request Body

```
{
"records" :[
  {
    "attributes" : {"type" : "Account", "referenceId" : "ref1"},
    "name" : "TreeAccountWithContacts",
    "phone" : "9876543212",
    "website" : "www.hoffensoft.com",
    "Contacts" : {
      "records" : [{
        "attributes" : {"type" : "Contact", "referenceId" : "ref2"},
        "lastname" : "John",
        "Title" : "Manager",
        "email" : "sample@hoffensoft.com"
      },{
        "attributes" : {"type" : "Contact", "referenceId" : "ref3"},
        "lastname" : "Peter",
        "title" : "Lead",
        "email" : "sample@hoffensoft.com"
      }]
    }
  }]
}
```

Response body after successfully creating records:

```
{
  "hasErrors" : false,
  "results" : [ {
    "referenceId" : "ref1",
    "id" : "0012v00002feLaNAAU"
  }, {
    "referenceId" : "ref2",
    "id" : "0032v00003GWtJeAAL"
  }, {
    "referenceId" : "ref3",
    "id" : "0032v00003GWtJfAAL"
  } ]
}
```

Composite

Execute a series of REST API subrequests in a single call. The output of one subrequest can be used as input to a subsequent subrequest. The requests in a composite call are called subrequests. In a subrequest's body, you specify a reference ID that maps to the subrequest's response. You can then refer to the ID in the URL or body fields of later subrequests by using a JavaScript-like reference notation. Subrequests can be calls to SObject and Query/QueryAll resources. You can have up to 25 subrequests in a single call. Up to 10 of these subrequests can be query operations.

URI	/vXX.X/composite
Formats	JSON
Method	GET (lists other available composite resources), POST
Authentication	Authorization: Bearer token
Request body	The request body contains an allOrNone flag that specifies how to roll back errors and a composite request collection that includes subrequests to execute.
Response body	Collection of subrequest results

Example

The below example uses the Composite resource to execute several dependent requests in a single call. First, it creates an account and retrieves its information. Next, it uses the account data and the Composite resource's reference ID functionality to create contact and populate its fields based on the account data

GET POST PUT PATCH DELETE HEAD Headers Reset Up

/services/data/v46.0/composite Execute

Request Body

```
{
    "allOrNone" : true,
    "compositeRequest" : [{
        "method" : "POST",
        "url" : "/services/data/v46.0/sobjects/Account",
        "referenceId" : "NewAccount",
        "body" : {
            "Name" : "Composite Request New Account",
            "BillingStreet" : "Landmark @ 1 Market Street",
            "BillingCity" : "San Francisco",
            "BillingState" : "California",
             "Industry" : "Technology"
        }
    },{
        "method" : "GET",
        "referenceId" : "NewAccountInfo",
        "url" : "/services/data/v46.0/sobjects/Account/@{NewAccount.id}"
    },{
        "method" : "POST",
        "referenceId" : "NewContact",
        "url" : "/services/data/v46.0/sobjects/Contact",
        "body" : {
            "lastname" : "Prakash",
            "Title" : "CTO of @{NewAccountInfo.Name}",
            "MailingStreet" : "@{NewAccountInfo.BillingStreet}",
            "MailingCity" : "@{NewAccountInfo.BillingAddress.city}",
            "MailingState" : "@{NewAccountInfo.BillingState}",
            "AccountId" : "@{NewAccountInfo.Id}",
            "Email" : "vp@hoffensoft.com",
            "Phone" : "1234567890"
        }

    }]
}
```

Response body after successfully creating records:

- compositeResponse
  - [Item 1]
    - body
      - id: **0012v00002fehiOAAQ**
      - success: **true**
      - errors
    - httpHeaders
      - Location: **/services/data/v46.0/sobjects/Account/0012v00002fehiOAAQ**
    - httpStatusCode: **201**
    - referenceId: **NewAccount**
  - [Item 2]
    - body
    - httpHeaders
      - ETag: **"gTI7lF2oYMlDxM+gTW1a62nzqxfxxihRqAygUNh9DPs="**
      - Last-Modified: **Wed, 13 Nov 2019 14:18:01 GMT**
    - httpStatusCode: **200**
    - referenceId: **NewAccountInfo**
  - [Item 3]
    - body
      - id: **0032v00003GXFecAAH**
      - success: **true**
      - errors
    - httpHeaders
    - httpStatusCode: **201**
    - referenceId: **NewContact**