# Bulk API 2.0

Version 51.0, Spring '21

'21

@salesforcedocs

Last updated: December 16, 2020

# CONTENTS

# CHAPTER 1 Bulk API 2.0

In this chapter ...



Bulk API 2.0 provides a simple interface to quickly load large amounts of data into your Salesforce org and to perform bulk queries on your org data.

Bulk API 2.0 is available in API version 41.0 and later. Query jobs in Bulk API 2.0 are available in API version 47.0 and later.

The API includes the following features.

**Support for OAuth 2.0**
Bulk API 2.0 supports all OAuth flows supported by other Salesforce REST APIs.

**Automatic File Batching**
Bulk API 2.0 simplifies uploading large amounts of data by breaking the data into batches automatically. All you have to do is upload a CSV file with your record data and check back when the results are ready.

**Daily Limits Based on Number of Records**
Instead of limiting the amount of data uploaded daily by the quantity of jobs, Bulk API 2.0 uses a limit of total records uploaded. See Limits and Allocations.

## API End-of-Life

Salesforce is committed to supporting each API version for a minimum of three years from the date of first release. In order to mature and improve the quality and performance of the API, versions that are more than three years old might cease to be supported.

When an API version is to be deprecated, advance notice is given at least one year before support ends. Salesforce will directly notify customers using API versions planned for deprecation.

# Process for Creating Bulk API 2.0 Requests

Use the Bulk API 2.0 to process large amounts of data by defining a job, providing data for that job, and monitoring the results.

Create a new Bulk API 2.0 request by doing the following steps.

1. Create a job. The job defines a specific Bulk API 2.0 operation, such as inserting new records, or updating records.
2. Upload data for the job. For data sets under 100,000 characters, upload the data as part of a multipart request when you create the job. For larger data sets, up to 150MB (after base64 encoding), upload the data by creating an upload data request. Don't delete your local CSV data until you've confirmed that Salesforce has processed all records successfully.
3. If you're using a separate request to upload data, once you're done uploading data, close the job by setting the job state to `UploadComplete`. Salesforce will start processing the job at this point.
4. Check the status of the job. When a job's state is `JobComplete`, you can retrieve its results.

   Note: You must use the same API version to get query results that you used to create the query. Otherwise, the call returns a 409 error.

5. Retrieve the results of the completed job by checking the successful results, failed results, and unprocessed records.

# How Requests Are Processed

Whether you create an ingest or query job, Salesforce automatically optimizes your request to process the job as quickly as possible, and to minimize timeouts or other failures.

When you create job requests with Bulk API 2.0, Salesforce provides a job "state" to describe the progress or outcome of the job. You can manually Check the status of the job, or you can view job state from within the Salesforce UI. From Setup, in the Quick Find box, enter *`Bulk Data Load Jobs`*, and then select **Bulk Data Load Jobs**. The following table summarizes Bulk API 2.0 job states during job creation and processing.

| Job Phase | State | Description |
|---|---|---|
| Creation | `Open` | An ingest job was created and is open for data uploads. |
| Creation | `UploadComplete` | (*Ingest*) All job data has been uploaded. The job is queued and ready to be processed.<br>(*Query*) Salesforce has put the query job in the queue. |
| Processing | `InProgress` | The job is being processed by Salesforce. Operations include automatic, optimized batching or chunking of job data, and processing of job operations. |
| Outcome | `JobComplete` | The job was processed. |
| Outcome | `Failed` | The job couldn't be processed successfully. |
| Outcome | `Aborted` | The job was canceled by the job creator, or by a user with the "Manage Data Integrations" permission. |

### Ingest Jobs

While processing ingest jobs, Salesforce Bulk API 2.0 automatically divides your job's data into multiple batches to improve performance.

Salesforce creates a separate batch for every 10,000 records in your job data, up to a daily maximum of 150,000,000 records. If the limit is exceeded while processing your job data, the remaining data isn't processed. The ingest job is marked as having failed.

Just as a job can fail, so can an individual batch. If Salesforce can't process all the records in a batch within 10 minutes, the batch fails. Salesforce automatically retries failed batches up to a maximum of 10 times. If the batch still can't be processed after 10 retries, the entire ingest job is moved to the `Failed` state and remaining job data isn't processed.

Note: If there's a failure, create a new ingest job to process the records that weren't processed.

To determine what records weren't processed and what errors occurred, use the Failed Record Results and Unprocessed Record Results resources.

**Query Jobs**

Bulk API 2.0 query jobs enable asynchronous processing of SOQL queries. They're designed to handle queries that return large amounts of data (10,000 records or more).

The daily limit for query jobs is 10,000 jobs per day.

The API automatically handles retries. If you receive a message that the API retried more than 15 times, apply a filter criteria and try again.

Instead of configuring batches, Bulk API 2.0 query jobs automatically determine the best way to divide your query job into smaller chunks, helping to avoid failures or timeouts. Bulk API 2.0 is optimized to chunk large query jobs with the following objects:

- Account
- AccountContactRelation
- AccountTeamMember
- AiVisitSummary
- Asset
- B2BMktActivity
- B2BMktProspect
- Campaign
- CampaignMember
- CandidateAnswer
- Case
- CaseArticle
- CaseComment
- Claim
- ClaimParticipant
- Contact
- ContractLineItem
- ConversationEntry
- CustomerProperty
- EinsteinAnswerFeedback
- EmailMessage
- EngagementScore
- Event
- EventRelation
- FeedItem

- Individual
- InsurancePolicy
- InsurancePolicyAsset
- InsurancePolicyParticipant
- Lead
- LeadInsight
- LiveChatTranscript
- LoginHistory
- LoyaltyLedger
- LoyaltyMemberCurrency
- LoyaltyMemberTier
- LoyaltyPartnerProduct
- LoyaltyProgramMember
- LoyaltyProgramPartner
- Note
- ObjectTerritory2Association
- Opportunity
- OpportunityContactRole
- OpportunityHistory
- OpportunityLineItem
- OpportunitySplit
- OpportunityTeamMember
- Pricebook2
- PricebookEntry
- Product2
- ProductConsumed
- ProductRequired
- QuickText
- Quote
- QuoteLineItem
- ReplyText
- ScoreIntelligence
- ServiceContract
- Task
- TermDocumentFrequency
- TransactionJournal
- User
- UserRole
- VoiceCall
- WorkOrder

- WorkOrderLineItem

This optimization also includes custom objects, and any Sharing and History tables that support standard objects.

# Limits and Allocations

The following limits and allocations and limits apply to Bulk API 2.0.

## Limits for Ingest Jobs

| Item | Limit |
|---|---|
| Maximum number of records uploaded per rolling 24-hour period | 150,000,000 |
| Maximum number of times a batch can be retried. See How Requests Are Processed. | The API automatically handles retries. If you receive a message that the API retried more than 10 times, use a smaller upload file and try again. |
| Maximum file size. | 150 MB |
| Maximum number of characters in a field. | 32,000 |
| Maximum number of fields in a record. | 5,000 |
| Maximum number of characters in a record. | 400,000 |

## Limits for Query Jobs

| Item | Limit |
|---|---|
| The maximum number of query jobs per 24-hour rolling window.<br>The current number can be seen in the `DailyBulkV2QueryJobs` value in the response to the `/vXX.X/limits/` REST API method. | 10,000 |
| The total query results that can be generated per 24 hour rolling window.<br>The current size can be seen in the `DailyBulkV2QueryFileStorageMB` value in the response to the `/vXX.X/limits/` REST API method. | 1 TB |
| The maximum number of times a query job is retried before failure. | The API automatically handles retries. If you receive a message that the API retried more than 15 times, apply a filter criteria and try again. |
| Results lifespan. | Jobs that are older than 7 days are deleted if they are in a terminal state (completed or failed), |

## Batch Allocations

In Bulk API 2.0, you do not have to manage batches. The API creates and handles them for you transparently in the background. This lets you focus on managing the jobs instead of the batches.

And in Bulk API 2.0, only *ingest* jobs consume batches. Query jobs do not. For details, see How Requests Are Processed.

The batch allocation (15,000 batches per day) is shared between Bulk API and Bulk API 2.0, so every batch that is processed in Bulk API or Bulk API 2.0 counts towards this allocation.

**Note:** In Bulk API 2.0, each ingest job creates an initial batch with zero records. This empty batch does not contribute to the daily allocation limit.

## See Also

For a comparison with Bulk API limits, see the *Salesforce Developer Limits and Allocations Quick Reference*.

# CHAPTER 2 Bulk Uploading and Upserting

In this chapter ...



Bulk API 2.0 enables you to upload and upsert large amounts of data into your Salesforce org.

# Examples

Use the following examples to understand how to do tasks with Bulk API 2.0

## Walkthrough for Bulk Insert

This walkthrough guides you through the steps for creating a job to insert new records, uploading data for the job, checking status, and retrieving the results.

- Read through all the steps in this walkthrough. You might also want to review the rest of this document to get a quick overview of Bulk API 2.0 calls and concepts.
- Familiarize yourself with Workbench. This walkthrough uses Workbench to issue Bulk API 2.0 calls, but you can use any tool or development environment that can make REST requests.

1. Create a CSV file containing your data.

   Using your preferred spreadsheet tool, create a CSV file that contains the records you want to insert. The first row of the CSV file lists the field names for the object you're working with. Each subsequent row corresponds to a record that you want to insert.

   For a simple example list of Contacts that can be used to create a CSV file, see: Bulk API Sample Contact Data

   For additional information on preparing CSV files, such as delimiter options and valid date/time formats, see Data Files.

2. Create a job.

   To do any Bulk API 2.0 task, such as inserting or updating records, you first create a job. The job specifies the type of object, such as Contact, that you're loading. The job also specifies the operation you're performing, such as insert or delete. Once you've created a job, you can use the resulting job ID in Bulk API 2.0 requests to close or abort (cancel) the job.

   a. In Workbench, log in to your org.

   b. Go to **Utilities** > **REST Explorer**. Use Bulk API 2.0 to create a job by issuing a POST request with the following details:

      **Request Headers:**

      ```
      Content-Type: application/json; charset=UTF-8
      Accept: application/json
      ```

      **URI:**

      ```
      /services/data/v***XX.X***/jobs/ingest/
      ```

      **Request Body:**

      ```
      {
          "object" : "Contact",
          "contentType" : "CSV",
          "operation" : "insert",
          "lineEnding": "CRLF"
      }
      ```

      If you're inserting an object other than Contact, replace "Contact" in the request body with your object name.

      When you're using Workbench, set `lineEnding` to `CRLF`. If you're using another tool, set it to match the line endings in your data file.

      You should get a response that includes the job ID, with a job state of `Open`. You'll use the job ID in Bulk API 2.0 calls in the next steps. You'll also use the URL in the `contentUrl` field of the response in the next step when you upload your data.

3. Upload your CSV data.

   After creating a job, you're ready to upload your data. You provide record data using the CSV file you created earlier.

   In Workbench's REST Explorer, use Bulk API 2.0 to create a job data request with the CSV data. Issue a PUT request with the following details:

   **Request Headers:**

   ```
   Content-Type: text/csv
   Accept: application/json
   ```

   **URI:**

   Use the URI provided in the `contentUrl` field of the response from step 1. The URI has a format similar to the following example.

   ```
   /services/data/vXX.X/jobs/ingest/jobId/batches/
   ```

   **Request Body:**

   ```
   (Content of your CSV file)
   ```

   For the Request Body, copy and paste the CSV data into Workbench's Request Body text field. With Bulk API 2.0, you can submit CSV data that does not in total exceed 150 MB in size (after base64 encoding).

   You should get a response that includes the job ID still in the `Open` state. You could cancel the job at this point by using a PATCH to set the job state to `Aborted`.

4. Close the job.

   Once you're done submitting data, you can inform Salesforce that the job is ready for processing by closing the job.

   In Workbench's REST Explorer, issue a PATCH request with the following details:

   **Request Headers:**

   ```
   Content-Type: application/json; charset=UTF-8
   Accept: application/json
   ```

   **URI:**

   ```
   /services/data/vXX.X/jobs/ingest/jobId/
   ```

   **Request Body:**

   ```
   {
       "state" : "UploadComplete"
   }
   ```

5. Check the job status and results.

   To get basic status information on a job, such as the overall job state or the number of records processed, use a GET request with the following details:

   **Request Headers:**

   ```
   Content-Type: application/json; charset=UTF-8
   Accept: application/json
   ```

   **URI:**

   ```
   /services/data/vXX.X/jobs/ingest/jobId/
   ```

Once a job has been completed and is in the `JobComplete` state (or `Failed` state), you can get details of which job data records were successfully processed by issuing a GET request with the following details:

**Request Headers:**

```
Content-Type: application/json; charset=UTF-8
Accept: text/csv
```

```
/services/data/vXX.X/jobs/ingest/jobId/successfulResults/
```

You get a response that contains CSV data, with each row containing a record ID and information on whether that record was successfully processed or not. To get details on records that encountered an error during processing, use a GET request using the `failedResults` resource instead of the `successfulResults` resource.

# Walkthrough for Bulk Insert With A Multipart Request

This walkthrough guides you through the steps for creating a job to insert new records, creating batches for the job, checking status, and retrieving the results. This walkthrough uses a single multipart request to create the job and upload job data.

- Read through all the steps in this walkthrough. You might also want to review the rest of this document to get a quick overview of Bulk API 2.0 calls and concepts.
- Familiarize yourself with Workbench. This walkthrough uses Workbench to issue Bulk API 2.0 calls, but you can use any tool or development environment that can make REST requests.

1. Create a CSV file containing your data.

   Using your preferred spreadsheet tool, create a CSV file that contains the records you want to insert. The first row of the CSV file lists the field names for the object you're working with. Each subsequent row corresponds to a record that you want to insert.

   For a very simple example list of Contacts that can be used to create a CSV file, see: Bulk API Sample Contact Data

   For additional information on preparing CSV files, such as delimiter options and valid date/time formats, see Data Files.

   For this walkthrough, since we'll use a multipart request to create the job and upload the data, the total size of the CSV data should not exceed 100,000 characters.

2. Create a job.

   To do any Bulk API 2.0 task, such as inserting or updating records, you first create a job. The job specifies the type of object, such as Contact, that you're loading. The job also specifies the operation you're performing, such as insert or delete. In this walkthrough we'll also include job data in the request that creates the job.

   Once you've created a job, you can use the resulting job ID in Bulk API 2.0 requests to close or abort (cancel) the job.

   a. In Workbench, log in to your org.

   b. Go to **Utilities** > **REST Explorer**. Use Bulk API 2.0 to create a new job and upload job data by issuing a POST request with the following details:

      **Request Headers:**

      ```
      Content-Type: multipart/form-data; boundary=BOUNDARY
      Accept: application/json
      ```

      **URI:**

      ```
      /services/data/vXX.X/jobs/ingest/
      ```

**Request Body:**

```
--BOUNDARY
Content-Type: application/json
Content-Disposition: form-data; name="job"

{
  "object":"Contact",
  "contentType":"CSV",
  "operation": "insert",
  "lineEnding": "CRLF"
}

--BOUNDARY
Content-Type: text/csv
Content-Disposition: form-data; name="content"; filename="content"

FirstName,LastName,MailingCity
Astro,Nomical,San Francisco
Hootie,McOwlface,San Francisco
Appy,Camper,San Francisco
Earnie,Badger,San Francisco
--BOUNDARY--
```

In this example, "BOUNDARY" is used to mark the request body boundaries between the job details and the job CSV data.

When you're using Workbench, set `lineEnding` to `CRLF`. If you're using another tool, set it to match the line endings in your data file.

You should get a response that includes the job ID, with a job state of `UploadComplete`. You'll use the job ID in Bulk API 2.0 calls in the next steps.

3. Check the job status and results.

   To get basic status information on a job, such as the overall job state or the number of records processed, use a GET request with the following details:

   **Request Headers:**

   ```
   Content-Type: application/json; charset=UTF-8
   Accept: application/json
   ```

   **URI:**

   ```
   /services/data/v***XX.X***/jobs/ingest/***jobId***/
   ```

   Once a job has been completed and is in the `JobComplete` state (or `Failed` state), you can get details of which job data records were successfully processed by issuing a GET request with the following details:

   **Request Headers:**

   ```
   Content-Type: application/json; charset=UTF-8
   Accept: text/csv
   ```

   ```
   /services/data/v***XX.X***/jobs/ingest/***jobId***/successfulResults/
   ```

You'll get a response that contains CSV data, with each row containing a record ID and information on whether that record was successfully processed or not. To get details on records that encountered an error during processing, use a GET request using the `failedResults` resource instead of the `successfulResults` resource.

# Walkthrough for Bulk Upsert

This walkthrough guides you through the steps for creating a job to upsert records, uploading data for the job, checking status, and retrieving the results.

- Read through all the steps in this walkthrough. You might also want to review the rest of this document to get a quick overview of Bulk API 2.0 calls and concepts.
- Familiarize yourself with Workbench. This walkthrough uses Workbench to issue Bulk API 2.0 calls, but you can use any tool or development environment that can make REST requests.

1. Confirm that your object is using an external ID field.

   Upserting records requires an external ID field on the object involved in the job. This field is used to identify existing records in the org. Bulk API 2.0 uses the external ID field to decide if a record in the bulk job data will be used to update an existing record, or create a new record.

2. Create a CSV file containing your data.

   Using your preferred spreadsheet tool, create a CSV file that contains the records you want to upsert. The first row of the CSV file lists the field names for the object you're working with. Each subsequent row corresponds to a record that you want to insert.

   One of the columns in the CSV must correspond to the external ID field identified in step 1.

   For additional information on preparing CSV files, such as delimiter options and valid date/time formats, see Data Files.

3. Create a job.

   To do any Bulk API 2.0 task, such as inserting or updating records, you first create a job. The job specifies the type of object, such as Contact, that you're loading. The job also specifies the operation you're performing, such as insert or delete. Once you've created a job, you can use the resulting job ID in Bulk API 2.0 requests to close or abort (cancel) the job.

   a. In Workbench, log in to your org.

   b. Go to **Utilities** > **REST Explorer**. Use Bulk API 2.0 to create a new job by issuing a POST request with the following details:

      **Request Headers:**

      ```
      Content-Type: application/json; charset=UTF-8
      Accept: application/json
      ```

      **URI:**

      ```
      /services/data/vXX.X/jobs/ingest/
      ```

      **Request Body:**

      ```
      {
          "object" : "Contact",
          "externalIdFieldName" : name of external id field
          "contentType" : "CSV",
          "operation" : "upsert"
      }
      ```

Note that you need to provide the name of the external ID field when you create the upsert job, using the `externalIdFieldName` property. If you're inserting an object other than Contact, replace "Contact" in the request body with your object name.

You should get a response that includes the job ID, with a job state of `Open`. You'll use the job ID in Bulk API 2.0 calls in the next steps. You'll also use the URL in the `contentUrl` field of the response in the next step when you upload your data.

4. Upload your CSV data.

   After creating a new job, you're ready to upload your data. You'll provide the record data using the CSV file you created earlier.

   In Workbench's REST Explorer, use Bulk API 2.0 to create a new job data request with the CSV data. Issue a PUT request with the following details:

   **Request Headers:**

   ```
   Content-Type: text/csv
   Accept: application/json
   ```

   **URI:**

   Use the URI provided in the `contentUrl` field of the response from step 3. The URI will have a format similar to the following example.

   ```
   /services/data/vXX.X/jobs/ingest/jobId/batches/
   ```

   **Request Body:**

   ```
   (Content of your CSV file)
   ```

   For the Request Body, copy and paste the CSV data into Workbench's Request Body text field. With Bulk API 2.0, you can submit CSV data that does not in total exceed 150MB in size (after base64 encoding).

   You should get a response that includes the job ID still in the `Open` state. You could cancel the job at this point by using a PATCH to set the job state to `Aborted`.

5. Close the job.

   Once you're done submitting data, you can inform Salesforce that the job is ready for processing by closing the job.

   In Workbench's REST Explorer, issue a PATCH request with the following details:

   **Request Headers:**

   ```
   Content-Type: application/json; charset=UTF-8
   Accept: application/json
   ```

   **URI:**

   ```
   /services/data/vXX.X/jobs/ingest/jobId/
   ```

   **Request Body:**

   ```
   {
       "state" : "UploadComplete"
   }
   ```

6. Check the job status and results.

   To get basic status information on a job, such as the overall job state or the number of records processed, use a GET request with the following details:

**Request Headers:**

```
Content-Type: application/json; charset=UTF-8
Accept: application/json
```

**URI:**

```
/services/data/vXX.X/jobs/ingest/jobId/
```

Once a job has been completed and is in the `JobComplete` state (or `Failed` state), you can get details of which job data records were successfully processed by issuing a GET request with the following details:

**Request Headers:**

```
Content-Type: application/json; charset=UTF-8
Accept: text/csv
```

```
/services/data/vXX.X/jobs/ingest/jobId/successfulResults/
```

You'll get a response that contains CSV data, with each row containing a record ID and information on whether that record was successfully processed or not. To determine which records were created and which records were updated during the upsert job, examine the `Created` field of the successful results. To get details on records that encountered an error during processing, use a GET request using the `failedResults` resource instead of the `successfulResults` resource.

# Data Files

Use comma-separated value (CSV) data when submitting data rows for Bulk API 2.0 jobs. Bulk API 2.0 supports several formatting options with CSV data, such as supporting multiple field delimiter characters and line ending characters.

## Prepare CSV Files

The first row in a CSV file lists the field names for the object that you're processing. Each subsequent row corresponds to a record in Salesforce.

All the records in a CSV file must be for the same object. You specify this object in the job associated with the batch.

Note the following when working with CSV files with Bulk API 2.0:

- You must include all required fields when you create a record. You can optionally include any other field for the object.
- Each field-name header in the file must be the same as the field's Field Name (for standard fields) or API Name (for custom fields). Results only include columns that are a match.
- If you're updating a record, any fields that aren't defined in the CSV file are ignored during the update.
- Files must be in UTF-8 format. Files are converted to base64 when received by Salesforce. This conversion can increase the data size by approximately 50%. To account for the base64 conversion increase, upload data that does not exceed 100 MB.
- Bulk API 2.0 supports several field delimiter characters: backquote (`), caret (^), comma, pipe (|), semicolon, and tab. The default delimiter is comma. Specify the delimiter to use when you create your job, using the `columnDelimiter` request field.
- Bulk API 2.0 supports several line ending formats: linefeed, and carriage-return plus linefeed. The default line ending is linefeed. Specify the line ending to use when you create your job, using the `lineEnding` request field.
- Use double-quotes to escape characters in field values that would otherwise get interpreted as field delimiters or line endings. For example, if a field value includes a comma, and comma is the current column delimiter for the job, you must wrap the field value in double-quotes in the CSV data, like "Director, Marketing".

- Field values aren't trimmed. A space before or after a delimiter is included in the field value. A space before or after a double quote generates an error for the row. For example, `John,Smith` is valid; `John, Smith` is valid, but the second value is `" Smith"`; `."John", "Smith"` is not valid.
- Empty field values are ignored when you update records. To set a field value to `null`, use a field value of `#N/A`.
- Fields with a `double` data type can include fractional values. Values can be stored in scientific notation if the number is large enough (or, for negative numbers, small enough), as indicated by the W3C XML Schema Part 2: Datatypes Second Edition specification.

Note: The header row can contain up to 32,000 characters.

To find the name of a field, you can:

- Use the `describeSObjects()` call in the *SOAP API Developer Guide*, or the `sObject Describe` resource in the *REST API Developer Guide*.
- Use Salesforce Setup.
- Look up the object in *Object Reference*, which lists the field names, types, and descriptions by object.

## Use Salesforce Setup to Find Field names

To find an object's field name in Salesforce Setup:

1. From **Setup**, in the **Quick Find** box, enter *`Object Manager`*. Click **Object Manager**.
2. Click on the object in the list.
3. From the object's management settings, click on **Fields & Relationships**.
4. Click the field under `Field Label` to find the field name.

For a standard field, use the `Field Name` value as the field column header in your CSV file.

For a custom field, use the `API Name` value as the field column header in a CSV file or the field name identifier in an XML or JSON file. (To find the API Name, click the field name.)

# Valid Date Format in Records

Specify the right format for `dateTime` and `date` fields.

## dateTime

Use the *`yyyy-MM-ddTHH:mm:ss.SSS+/-HHmm`* or *`yyyy-MM-ddTHH:mm:ss.SSSZ`* formats to specify `dateTime` fields.

- `yyyy` is the four-digit year
- `MM` is the two-digit month (01-12)
- `dd` is the two-digit day (01-31)
- 'T' is a separator indicating that time-of-day follows
- `HH` is the two-digit hour (00-23)
- `mm` is the two-digit minute (00-59)
- `ss` is the two-digit seconds (00-59)
- `SSS` is the optional three-digit milliseconds (000-999)
- `+/-HHmm` is the Zulu (UTC) time zone offset
- 'Z' is the reference UTC timezone

When a timezone is added to a UTC `dateTime`, the result is the date and time in that timezone. For example, 2002-10-10T12:00:00+05:00 is 2002-10-10T07:00:00Z and 2002-10-10T00:00:00+05:00 is 2002-10-09T19:00:00Z. See W3C XML Schema Part 2: DateTime Datatype.

### date

Use the *yyyy-MM-dd* format to specify `date` fields.

Note: Specifying an offset for `date` is not supported.

## Relationship Fields in a Header Row

Many objects in Salesforce are related to other objects. For example, Account is a parent of Contact. You can add a reference to a related object in a CSV file by representing the relationship in a column header. When you're processing records in the Bulk API, you use ***RelationshipName.IndexedFieldName*** syntax in a CSV column header to describe the relationship between an object and its parent, where *RelationshipName* is the relationship name of the field and *IndexedFieldName* is the indexed field name that uniquely identifies the parent record. Use the `describeSObjects()` call in the SOAP-based SOAP API to get the `relationshipName` property value for a field.

Some objects also have relationships to themselves. For example, the `Reports To` field for a contact is a reference to another contact. If you're inserting a contact, you could use a `ReportsTo.Email` column header to indicate that you're using a contact's `Email` field to uniquely identify the `Reports To` field for a contact. The `ReportsTo` portion of the column header is the `relationshipName` property value for the `Reports To` field. The following CSV file uses a relationship:

```
FirstName,LastName,ReportsTo.Email
Tom,Jones,buyer@salesforcesample.com
```

Note the following when referencing relationships in CSV header rows:

- You can use a child-to-parent relationship, but you can't use a parent-to-child relationship.
- You can use a child-to-parent relationship, but you can't extend it to use a child-to-parent-grandparent relationship.
- You can only use indexed fields on the parent object. A custom field is indexed if its `External ID` field is selected. A standard field is indexed if its `idLookup` property is set to `true`. See the Field Properties column in the field table for each standard object.

### Relationship Fields for Custom Objects

Custom objects use custom fields to track relationships between objects. Use the relationship name, which ends in `__r` (underscore-underscore-r), to represent a relationship between two custom objects. You can add a reference to a related object by representing the relationship in a column header.

If the child object has a custom field with an `API Name` of `Mother_Of_Child__c` that points to a parent custom object and the parent object has a field with an `API Name` of `External_ID__c`, use the column header `Mother_Of_Child__r.External_ID__c` to indicate that you're using the parent object's `External ID` field to uniquely identify the `Mother Of Child` field. To use a relationship name in a column header, replace the `__c` in the child object's custom field with `__r`. For more information about relationships, see Understanding Relationship Names in the *Salesforce SOQL and SOSL Reference Guide* at `www.salesforce.com/us/developer/docs/soql_sosl/index.htm`.

The following CSV file uses a relationship:

```
Name,Mother_Of_Child__r.External_ID__c
CustomObject1,123456
```

## Relationships for Polymorphic Fields

A polymorphic field can refer to more than one type of object as a parent. For example, either a contact or a lead can be the parent of a task. In other words, the `WhoId` field of a task can contain the ID of either a contact or a lead. Since a polymorphic field is more flexible, the syntax for the column header has an extra element to define the type of the parent object. The syntax is ***`ObjectType:RelationshipName.IndexedFieldName`***. The following sample includes two reference fields:

1. The `WhoId` field is polymorphic and has a `relationshipName` of `Who`. It refers to a lead and the indexed `Email` field uniquely identifies the parent record.
2. The `OwnerId` field is not polymorphic and has a `relationshipName` of `Owner`. It refers to a user and the indexed `Id` field uniquely identifies the parent record.

```
Subject,Priority,Status,Lead:Who.Email,Owner.Id
Test Bulk API polymorphic reference field,Normal,Not
Started,lead@salesforcesample.com,005D0000001AXYz
```

Warning: The ***`ObjectType:`*** portion of a field column header is only required for a polymorphic field. You get an error if you omit this syntax for a polymorphic field. You also get an error if you include this syntax for a field that is not polymorphic.

# Sample CSV Files

The following examples demonstrate different ways to use CSV data with Bulk API 2.0.

For simplicity, all the following examples only show a few records.

## Simple CSV

This example contains three Account records and specifies the Name, Description, and NumberOfEmployees fields for each record.

```
Name,Description,NumberOfEmployees
TestAccount1,Description of TestAccount1,30
TestAccount2,Another description,40
TestAccount3,Yet another description,50
```

A job that uses this CSV data might be defined with the following job properties.

```
{
  "object" : "Account",
  "contentType" : "CSV",
  "operation" : "insert"
}
```

## CSV with Alternate Line Ending

This example contains two Contact records and specifies three fields for each record. The data was created on a Windows platform, and each line ends with a carriage return and line feed. The carriage return is displayed as "^M" in this example.

```
FirstName,LastName,Description^M
Tom,Jones,Branding guru^M
Ian,Dury,Fuzzy logic expert^M
```

A job that uses this CSV data and specifies that carriage return + line feed is used as the line ending sequence would use the following job properties.

```
{
  "object" : "Contact",
  "contentType" : "CSV",
  "operation" : "insert",
  "lineEnding" : "CRLF"
}
```

## CSV with Semicolon Delimiter and Escaped Fields

This example contains two Contact records and specifies five fields for each record. The field delimiter is a semicolon instead of a comma. The Description fields contain characters that must be escaped using double-quotes, including a line-break in the second record.

```
FirstName;LastName;Title;Birthdate;Description
Tom;Jones;Senior Director;1940-06-07Z;"Self-described as ""the top"" branding guru"
Ian;Dury;Chief Imagineer;1965-12-11Z;"Expert in fuzzy logic design; Knowledgeable in AI
Influential in technology purchases."
```

A job that uses this CSV data and specifies that semicolon is used as the column delimiter would use the following job properties.

```
{
  "object" : "Contact",
  "contentType" : "CSV",
  "operation" : "insert",
  "columnDelimiter" : "SEMICOLON"
}
```

## CSV with Relationship Field

This example contains two Contact records and specifies FirstName, LastName, and Owner.Email fields for each record. This example assumes a unique User record exists that has an Email value of "mfellow@salesforce.com" and creates a relationship with this record and the Contact records. If the User record doesn't exist, or if there are multiple User records with an Email value of "mfellow@salesforce.com", the relationship can't be created and the job fails.

```
FirstName,LastName,Owner.Email
Joe,User,mfellow@salesforce.com
Jane,User,mfellow@salesforce.com
```

A job that uses this CSV data might be defined with the following job properties.

```
{
  "object" : "Contact",
  "contentType" : "CSV",
  "operation" : "insert"
}
```

### CSV for Upsert Using External IDs

This example contains three Contact records and specifies FirstName, LastName, Phone, and ExternalId__c for each record. This example assumes the custom ExternalId__c external ID field has been added to Contact.

```
FirstName,LastName,Phone,ExternalId__c
Mark,Brown,4155558787,"1001"
Dave,Stillman,4155552212,"1002"
Joe,Smith,2125556363,"5001"
```

A job that uses this CSV data might be an upsert job defined with the following properties.

```
{
  "object" : "Contact",
  "externalIdFieldName" : "ExternalId__c",
  "contentType" : "CSV",
  "operation" : "upsert"
}
```

# Reference

The API reference for Bulk API 2.0 includes all the actions that you can perform with jobs.

## Create a Job

Creates a job, which represents a bulk operation (and associated data) that is sent to Salesforce for asynchronous processing. Provide job data via an **Upload Job Data** request, or as part of a multipart create job request.

**URI**

/services/data/v*XX.X*/jobs/ingest

**Availability**

This resource is available in API version 41.0 and later.

**Formats**

JSON

**HTTP Method**

POST

**Authentication**

`Authorization: Bearer` ***`token`***

**Parameters**

None.

**Headers**

Optionally, use the `Sforce-Call-Options` header to specify a default namespace.

**Request Body**

| Property | Type | Description | Required or Optional |
|---|---|---|---|
| `assignmentRuleId` | string | The ID of an assignment rule to run for a Case or a Lead. The assignment rule can be active or inactive. The ID can | Optional |

| Property | Type | Description | Required or Optional |
|---|---|---|---|
| | | be retrieved by using the Lightning Platform SOAP API or the Lightning Platform REST API to query the AssignmentRule object.<br>This property is available in API version 49.0 and later. | |
| `columnDelimiter` | ColumnDelimiterEnum | The column delimiter used for CSV job data. The default value is `COMMA`. Valid values are:<br>• `BACKQUOTE`—backquote character (`)<br>• `CARET`—caret character (^)<br>• `COMMA`—comma character (,) which is the default delimiter<br>• `PIPE`—pipe character (\|)<br>• `SEMICOLON`—semicolon character (;)<br>• `TAB`—tab character | Optional |
| `contentType` | ContentType | The content type for the job. The only valid value (and the default) is `CSV`. | Optional |
| `externalIdFieldName` | string | The external ID field in the object being updated. Only needed for upsert operations. Field values must also exist in CSV job data. | Required for upsert operations |
| `lineEnding` | LineEndingEnum | The line ending used for CSV job data, marking the end of a data row. The default is `LF`. Valid values are:<br>• `LF`—linefeed character<br>• `CRLF`—carriage return character followed by a linefeed character | Optional |
| `object` | string | The object type for the data being processed. Use only a single object type per job. | Required |
| `operation` | OperationEnum | The processing operation for the job. Valid values are:<br>• `insert`<br>• `delete`<br>• `hardDelete`<br>• `update`<br>• `upsert`<br>Note: When the `hardDelete` value is specified, the deleted records aren't stored in the Recycle Bin. Instead, they become immediately eligible for deletion. The permission for this operation, "Bulk API Hard Delete," is disabled by default and must | Required |

| Property | Type | Description | Required or Optional |
|---|---|---|---|
| | | be enabled by an administrator. A Salesforce user license is required for hard delete. | |

For multipart requests, the request body can also include CSV record data. See Usage Notes on page 22 for more details.

**Response Body**

| Property | Type | Description |
|---|---|---|
| `apiVersion` | string | The API version that the job was created in. |
| `assignmentRuleId` | id | The ID of the assignment rule. This property is only shown if an assignment rule is specified when the job is created. |
| `columnDelimiter` | ColumnDelimiterEnum | The column delimiter used for CSV job data. Values include:<br>• `BACKQUOTE`—backquote character (`)<br>• `CARET`—caret character (^)<br>• `COMMA`—comma character (,) which is the default delimiter<br>• `PIPE`—pipe character (\|)<br>• `SEMICOLON`—semicolon character (;)<br>• `TAB`—tab character |
| `concurrencyMode` | ConcurrencyModeEnum | For future use. How the request was processed. Currently only parallel mode is supported. (When other modes are added, the mode will be chosen automatically by the API and will not be user configurable.) |
| `contentType` | ContentType | The format of the data being processed. Only CSV is supported. |
| `contentUrl` | URL | The URL to use for Upload Job Data on page 23 requests for this job. Only valid if the job is in `Open` state. |
| `createdById` | string | The ID of the user who created the job. |
| `createdDate` | dateTime | The date and time in the UTC time zone when the job was created. |
| `externalIdFieldName` | string | The name of the external ID field for an upsert. |
| `id` | string | Unique ID for this job. |
| `jobType` | JobTypeEnum | The job's type. Values include:<br>• `BigObjectIngest`—BigObjects job<br>• `Classic`—Bulk API 1.0 job<br>• `V2Ingest`—Bulk API 2.0 job |
| `lineEnding` | LineEndingEnum | The line ending used for CSV job data. Values include:<br>• `LF`—linefeed character<br>• `CRLF`—carriage return character followed by a linefeed character |

| Property | Type | Description |
|---|---|---|
| `object` | string | The object type for the data being processed. |
| `operation` | | The processing operation for the job. Values include:<br>• `insert`<br>• `delete`<br>• `hardDelete`<br>• `update`<br>• `upsert` |
| `state` | JobStateEnum | The current state of processing for the job. Values include:<br>• `Open`—The job has been created, and data can be added to the job.<br>• `UploadComplete`—No new data can be added to this job. You can't edit or save a closed job.<br>• `Aborted`—The job has been aborted. You can abort a job if you created it or if you have the "Manage Data Integrations" permission.<br>• `JobComplete`—The job was processed by Salesforce.<br>• `Failed`—Some records in the job failed. Job data that was successfully processed isn't rolled back. |
| `systemModstamp` | dateTime | Date and time in the UTC time zone when the job finished. |

**Usage Notes**

For small amounts of job data (100,000 characters or less), you can create a job and upload all the data for a job using a multipart request. The following example request header and body uses a multipart format to contain both job information and job data.

```
Content-Type: multipart/form-data; boundary=BOUNDARY
```

```
--BOUNDARY
Content-Type: application/json
Content-Disposition: form-data; name="job"

{
  "object":"Contact",
  "contentType":"CSV",
  "operation":"insert"
}

--BOUNDARY
Content-Type: text/csv
Content-Disposition: form-data; name="content"; filename="content"

(Content of your CSV file)
--BOUNDARY--
```

# Upload Job Data

Uploads data for a job using CSV data you provide.

**URI**

/services/data/v*XX.X*/jobs/ingest/*jobID*/batches

**Availability**

This resource is available in API version 41.0 and later.

**Formats**

text/csv

**HTTP Method**

PUT

**Authentication**

`Authorization: Bearer token`

**Parameters**

None.

**Request Body**

CSV file with record data.

**Response Body**

None. Returns a status code of 201 (Created), which indicates that the job data was successfully received by Salesforce.

**Usage Notes**

The resource URL is provided in the `contentUrl` field in the response from Create a Job, or the response from a Job Info request on an open job.

A request can provide CSV data that does not in total exceed 150 MB of base64 encoded content. When job data is uploaded, it is converted to base64. This conversion can increase the data size by approximately 50%. To account for the base64 conversion increase, upload data that does not exceed 100 MB.

Don't delete your local CSV data until you've confirmed that all records were successfully processed by Salesforce. If a job fails, use the successful results, failed results, and unprocessed records resources to determine what records from your CSV data you need to resubmit.

# Close or Abort a Job

Closes or aborts a job. If you close a job, Salesforce queues the job and uploaded data for processing, and you can't add any additional job data. If you abort a job, the job does not get queued or processed.

**URI**

/services/data/v*XX.X*/jobs/ingest/*jobID*

**Availability**

This resource is available in API version 41.0 and later.

**Formats**

JSON

**HTTP Method**

PATCH

**Authentication**

`Authorization: Bearer token`

**Parameters**

None.

**Request Body**

| Property | Type | Description | Required or Optional |
|---|---|---|---|
| `state` | JobStateEnum | The state to update the job to. Use `UploadComplete` to close a job, or `Aborted` to abort a job. | Required |

**Response Body**

| Property | Type | Description |
|---|---|---|
| `apiVersion` | string | The API version that the job was created in. |
| `assignmentRuleId` | id | The ID of the assignment rule. This property is only shown if an assignment rule is specified when the job is created. |
| `columnDelimiter` | ColumnDelimiterEnum | The column delimiter used for CSV job data. Values include:<br>• `BACKQUOTE`—backquote character (`)<br>• `CARET`—caret character (^)<br>• `COMMA`—comma character (,) which is the default delimiter<br>• `PIPE`—pipe character (\|)<br>• `SEMICOLON`—semicolon character (;)<br>• `TAB`—tab character |
| `concurrencyMode` | ConcurrencyModeEnum | For future use. How the request was processed. Currently only parallel mode is supported. (When other modes are added, the mode will be chosen automatically by the API and will not be user configurable.) |
| `contentType` | ContentType | The format of the data being processed. Only CSV is supported. |
| `contentUrl` | URL | The URL to use for Upload Job Data on page 23 requests for this job. Only valid if the job is in `Open` state. |
| `createdById` | string | The ID of the user who created the job. |
| `createdDate` | dateTime | The date and time in the UTC time zone when the job was created. |
| `externalIdFieldName` | string | The name of the external ID field for an upsert. |
| `id` | string | Unique ID for this job. |
| `jobType` | JobTypeEnum | The job's type. Values include:<br>• `BigObjectIngest`—BigObjects job<br>• `Classic`—Bulk API 1.0 job<br>• `V2Ingest`—Bulk API 2.0 job |

| Property | Type | Description |
|---|---|---|
| `lineEnding` | LineEndingEnum | The line ending used for CSV job data. Values include:<br>• `LF`—linefeed character<br>• `CRLF`—carriage return character followed by a linefeed character |
| `object` | string | The object type for the data being processed. |
| `operation` | | The processing operation for the job. Values include:<br>• `insert`<br>• `delete`<br>• `hardDelete`<br>• `update`<br>• `upsert` |
| `state` | JobStateEnum | The current state of processing for the job. Values include:<br>• `Open`—The job has been created, and data can be added to the job.<br>• `UploadComplete`—No new data can be added to this job. You can't edit or save a closed job.<br>• `Aborted`—The job has been aborted. You can abort a job if you created it or if you have the "Manage Data Integrations" permission.<br>• `JobComplete`—The job was processed by Salesforce.<br>• `Failed`—Some records in the job failed. Job data that was successfully processed isn't rolled back. |
| `systemModstamp` | dateTime | Date and time in the UTC time zone when the job finished. |

# Delete a Job

Deletes a job. To be deleted, a job must have a state of `UploadComplete`, `JobComplete`, `Aborted`, or `Failed`.

**URI**

/services/data/v*`XX.X`*/jobs/ingest/*`jobID`*

**Availability**

This resource is available in API version 41.0 and later.

**Formats**

JSON

**HTTP Method**

DELETE

**Authentication**

`Authorization: Bearer` ***`token`***

**Parameters**

None.

**Request Body**
None required.

**Response Body**
None. Returns a status code of 204 (No Content), which indicates that the job was successfully deleted.

**Usage Notes**
When a job is deleted, job data stored by Salesforce is also deleted and job metadata information is removed. The job will no longer display in the Bulk Data Load Jobs page in Salesforce.

# Get All Jobs

Retrieves all jobs in the org.

**URI**
/services/data/v*XX.X*/jobs/ingest

**Availability**
This resource is available in API version 41.0 and later.

**Formats**
JSON

**HTTP Method**
GET

**Authentication**
`Authorization: Bearer` ***token***

**Parameters**

| Parameter | Description |
|---|---|
| `isPkChunkingEnabled` | If set to `true`, filters jobs with PK chunking enabled. |
| `jobType` | Filters jobs based on job type. Valid values include:<br>• `BigObjectIngest`—BigObjects job<br>• `Classic`—Bulk API 1.0 job<br>• `V2Ingest`—Bulk API 2.0 job |
| `queryLocator` | Use `queryLocator` with a locator value to get a specific set of job results. Get All Jobs returns up to 1000 result rows per request, along with a `nextRecordsUrl` value that contains the locator value used to get the next set of results. |

**Request Body**
None required.

**Response Body**

| Property | Type | Description |
|---|---|---|
| `done` | boolean | Indicates whether there are more jobs to get. If `false`, use the `nextRecordsUrl` value to retrieve the next group of jobs. |

| Property | Type | Description |
|---|---|---|
| `records` | JobInfo[] | Contains information for each retrieved job. |
| `nextRecordsUrl` | URL | A URL that contains a query locator used to get the next set of results in a subsequent request if `done` isn't `true`. |

**JobInfo**

| Property | Type | Description |
|---|---|---|
| `apiVersion` | string | The API version that the job was created in. |
| `columnDelimiter` | ColumnDelimiterEnum | The column delimiter used for CSV job data. Values include:<br>• `BACKQUOTE`—backquote character (`)<br>• `CARET`—caret character (^)<br>• `COMMA`—comma character (,) which is the default delimiter<br>• `PIPE`—pipe character (\|)<br>• `SEMICOLON`—semicolon character (;)<br>• `TAB`—tab character |
| `concurrencyMode` | ConcurrencyModeEnum | For future use. How the request was processed. Currently only parallel mode is supported. (When other modes are added, the mode will be chosen automatically by the API and will not be user configurable.) |
| `contentType` | ContentType | The format of the data being processed. Only CSV is supported. |
| `contentUrl` | URL | The URL to use for Upload Job Data requests for this job. Only valid if the job is in `Open` state. |
| `createdById` | string | The ID of the user who created the job. Create the batch with the same user. |
| `createdDate` | dateTime | The date and time in the UTC time zone when the job was created. |
| `id` | string | Unique ID for this job. |
| `jobType` | JobTypeEnum | The job's type. Values include:<br>• `BigObjectIngest`: BigObjects job<br>• `Classic`: Bulk API 1.0 job<br>• `V2Ingest`: Bulk API 2.0 job |
| `lineEnding` | LineEndingEnum | The line ending used for CSV job data. Values include:<br>• `LF`—linefeed character<br>• `CRLF`—carriage return character followed by a linefeed character |
| `object` | string | The object type for the data being processed. |

| Property | Type | Description |
|---|---|---|
| `operation` | | The processing operation for the job. Values include:<br>• `insert`<br>• `delete`<br>• `hardDelete`<br>• `update`<br>• `upsert` |
| `state` | JobStateEnum | The current state of processing for the job. Values include:<br>• `Open`—The job has been created, and data can be added to the job.<br>• `UploadComplete`—No new data can be added to this job. You can't edit or save a closed job.<br>• `Aborted`—The job has been aborted. You can abort a job if you created it or if you have the "Manage Data Integrations" permission.<br>• `JobComplete`—The job was processed by Salesforce.<br>• `Failed`—Some records in the job failed. Job data that was successfully processed isn't rolled back. |
| `systemModstamp` | dateTime | Date and time in the UTC time zone when the job finished. |

# Get Job Info

Retrieves detailed information about a job.

**URI**

/services/data/v*XX.X*/jobs/ingest/*jobID*

**Availability**

This resource is available in API version 41.0 and later.

**Formats**

JSON

**HTTP Method**

GET

**Authentication**

`Authorization: Bearer` ***token***

**Request parameters**

| Parameter | Description | Required or Optional |
|---|---|---|
| `jobId` | The ID of the job. | Required |

**Request Body**

None required.

**Response Body**

| Property | Type | Description |
|---|---|---|
| `apexProcessingTime` | long | The number of milliseconds taken to process triggers and other processes related to the job data. This doesn't include the time used for processing asynchronous and batch Apex operations. If there are no triggers, the value is 0. |
| `apiActiveProcessingTime` | long | The number of milliseconds taken to actively process the job and includes `apexProcessingTime`, but doesn't include the time the job waited in the queue to be processed or the time required for serialization and deserialization. |
| `apiVersion` | string | The API version that the job was created in. |
| `assignmentRuleId` | string | The ID of an assignment rule to run for a Case or a Lead. . |
| `columnDelimiter` | ColumnDelimiterEnum | The column delimiter used for CSV job data. Values include:<br>• `BACKQUOTE`—backquote character (`)<br>• `CARET`—caret character (^)<br>• `COMMA`—comma character (,) which is the default delimiter<br>• `PIPE`—pipe character (\|)<br>• `SEMICOLON`—semicolon character (;)<br>• `TAB`—tab character |
| `concurrencyMode` | ConcurrencyModeEnum | For future use. How the request was processed. Currently only parallel mode is supported. (When other modes are added, the mode will be chosen automatically by the API and will not be user configurable.) |
| `contentType` | ContentType | The format of the data being processed. Only CSV is supported. |
| `contentUrl` | URL | The URL to use for Upload Job Data requests for this job. Only valid if the job is in `Open` state. |
| `createdById` | string | The ID of the user who created the job. |
| `createdDate` | dateTime | The date and time in the UTC time zone when the job was created. |
| `externalIdFieldName` | string | The name of the external ID field for an upsert. |
| `id` | string | Unique ID for this job. |
| `jobType` | JobTypeEnum | The job's type. Values include:<br>• `BigObjectIngest`: BigObjects job<br>• `Classic`: Bulk API 1.0 job<br>• `V2Ingest`: Bulk API 2.0 job |

| Property | Type | Description |
|---|---|---|
| `lineEnding` | LineEndingEnum | The line ending used for CSV job data. Values include:<br>• `LF`—linefeed character<br>• `CRLF`—carriage return character followed by a linefeed character |
| `numberRecordsFailed` | long | The number of records that were not processed successfully in this job.<br>This property is of type int in API version 46.0 and earlier. |
| `numberRecordsProcessed` | long | The number of records already processed.<br>This property is of type int in API version 46.0 and earlier. |
| `object` | string | The object type for the data being processed. |
| `operation` | OperationEnum | The processing operation for the job. Values include:<br>• `insert`<br>• `delete`<br>• `hardDelete`<br>• `update`<br>• `upsert` |
| `retries` | int | The number of times that Salesforce attempted to save the results of an operation. The repeated attempts are due to a problem, such as a lock contention. |
| `state` | JobStateEnum | The current state of processing for the job. Values include:<br>• `Open`: The job has been created, and job data can be uploaded to the job.<br>• `UploadComplete`: All data for a job has been uploaded, and the job is ready to be queued and processed. No new data can be added to this job. You can't edit or save a closed job.<br>• `InProgress`: The job is being processed by Salesforce. This includes automatic optimized chunking of job data and processing of job operations.<br>• `Aborted`: The job has been aborted. You can abort a job if you created it or if you have the "Manage Data Integrations" permission.<br>• `JobComplete`: The job was processed by Salesforce.<br>• `Failed`: Some records in the job failed. Job data that was successfully processed isn't rolled back. |
| `systemModstamp` | dateTime | Date and time in the UTC time zone when the job finished. |
| `totalProcessingTime` | long | The number of milliseconds taken to process the job. |

**Response Body - For an Unsuccessful Request**

If the request fails, the server returns a non-200 status, and the request body shows details of the error. For example, if the job has been deleted the status is 404 (Not Found) and the response body is:

```
[{
 "errorCode": "NOT_FOUND",
 "message": "The requested resource does not exist"
}]
```

## Example

This example gets information about the job with ID `7506g00000DhRA2AAN`:

```
curl --include --request GET \
--header "Authorization: Bearer token" \
"https://instance.salesforce.com/services/data/vXX.X/jobs/query/7506g00000DhRA2AAN
```

The response is:

```
{
  "id" : "7506g00000DhRA2AAN",
  "operation" : "insert",
  "object" : "Account",
  "createdById" : "0056g000005HQPyAAO",
  "createdDate" : "2018-12-18T22:51:36.000+0000",
  "systemModstamp" : "2018-12-18T22:51:58.000+0000",
  "state" : "Open",
  "concurrencyMode" : "Parallel",
  "contentType" : "CSV",
  "apiVersion" : 51.0,
  "jobType" : "V2Ingest",
  "contentUrl" : "services/data/v51.0/jobs/ingest/7506g00000DhRA2AAN/batches",
  "lineEnding" : "LF",
  "columnDelimiter" : "COMMA",
  "retries" : 0,
  "totalProcessingTime" : 0,
  "apiActiveProcessingTime" : 0,
  "apexProcessingTime" : 0
}
```

# Get Job Successful Record Results

Retrieves a list of successfully processed records for a completed job.

**URI**

/services/data/v*XX.X*/jobs/ingest/*jobID*/successfulResults/

**Availability**

This resource is available in API version 41.0 and later.

**Formats**

CSV

**HTTP Method**

GET

**Authentication**

```
Authorization: Bearer token
```

**Parameters**

None.

**Request Body**

None required.

**Response Body**

The response body is a CSV file that all the records that the job successfully processed. Each row corresponds to a successfully processed record and contains the following information.

| Property | Type | Description |
|---|---|---|
| `sf__Created` | boolean | Indicates if the record was created. |
| `sf__Id` | string | ID of the record that was successfully processed. |
| *`Fields from the original CSV request data`* | *various* | Field data for the row that was provided in the original job data upload request. |

**Usage Notes**

- The order of records in the response is not guaranteed to match the ordering of records in the original job data.
- Results are not recorded for batches that exceed the daily batch allocation.

# Get Job Failed Record Results

Retrieves a list of failed records for a completed job.

**URI**

/services/data/v*XX.X*/jobs/ingest/*jobID*/failedResults/

**Availability**

This resource is available in API version 41.0 and later.

**Formats**

CSV

**HTTP Method**

GET

**Authentication**

```
Authorization: Bearer token
```

**Parameters**

None.

**Request Body**

None required.

**Response Body**

The response body is a CSV file that all the records that encountered an error while being processed by the job. Each row corresponds to a failed record and contains the following information.

| Property | Type | Description |
|---|---|---|
| `sf__Error` | Error | Error code and message, if applicable. |
| `sf__Id` | string | ID of the record that had an error during processing, if applicable. |
| *`Fields from the original CSV request data`* | *various* | Field data for the row that was provided in the original job data upload request. |

**Usage Notes**

- The order of records in the response is not guaranteed to match the ordering of records in the original job data.
- Results are not recorded for batches that exceed the daily batch allocation.

# Get Job Unprocessed Record Results

Retrieves a list of unprocessed records for a completed job.

**URI**

/services/data/v*`XX.X`*/jobs/ingest/*`jobID`*/unprocessedrecords/

**Availability**

This resource is available in API version 41.0 and later.

**Formats**

CSV

**HTTP Method**

GET

**Authentication**

`Authorization: Bearer` ***`token`***

**Parameters**

None.

**Request Body**

None required.

**Response Body**

The response body is a CSV file that contains all the records that were not processed by the job.

A job that is interrupted or otherwise fails to complete can result in rows that aren't processed. Unprocessed rows are not the same as failed rows. Failed rows are processed but encounter an error during processing.

Each row corresponds to an unprocessed record and contains the following information.

| Property | Type | Description |
|---|---|---|
| *`Fields from the original CSV request data`* | *various* | Field data for the row that was provided in the original job data upload request. |

**Usage Notes**

- The order of records in the response is not guaranteed to match the ordering of records in the original job data.
- Results are not recorded for batches that exceed the daily batch allocation.

# CHAPTER 3 Bulk Queries

In this chapter ...



Bulk query jobs enable asynchronous processing of SOQL queries. They are designed to handle queries that return large amounts of data (10,000 records or more).

## Availability

Query jobs in Bulk API 2.0 are available in API version 47.0 and later.

## Comparison with Query Jobs in Bulk API

Bulk queries are also available in Bulk API. (See Bulk Query.) But the Bulk API 2.0 implementation has several advantages:

- It does not require you to handle batches. All results are returned from one endpoint.
- Limits have been simplified and are available to clients via the REST API `/limits` endpoint.
- Bulk API 2.0 Query is better integrated with other Salesforce REST APIs.
  - It does not require a special `X-SFDC-Session` header.
  - It supports all the regular OAuth workflows.
  - Its design is more consistent with the other APIs.

## Summary

The following table lists the URIs and methods supported by queries in Bulk API 2.0.

| URI | HTTP Method | Description |
|---|---|---|
| `/services/data/v`***`XX.X`***`/jobs/query` | POST | Creates a query job. |
| `/services/data/v`***`XX.X`***`/jobs/query` | GET | Gets information about all query jobs in the org. |
| `/services/data/v`***`XX.X`***`/jobs/query/`***`queryJobId`*** | GET | Gets information about one query job. |
| `/services/data/v`***`XX.X`***`/jobs/query/`***`queryJobId`***`/results` | GET | Gets the results for a query job. |

| URI | HTTP Method | Description |
|---|---|---|
| `/services/data/v`***`XX.X`***`/jobs/query/`***`queryJobId`*** | PATCH | Aborts a query job. |
| `/services/data/v`***`XX.X`***`/jobs/query/`***`queryJobId`*** | DELETE | Deletes a query job. |

# Limits

Bulk API 2.0 does not support queries with any of the following:

- GROUP BY, OFFSET, or TYPEOF clauses.
- Aggregate Functions such as `COUNT()`.
- Date functions in GROUP BY clauses. (Date functions in WHERE clauses are supported.)
- Compound address fields or compound geolocation fields.
- Parent-to-child relationship queries. (Child-to-parent relationship queries are supported.)

The following table lists the limits of query jobs in Bulk API 2.0.

| Item | Limit |
|---|---|
| The maximum number of query jobs per 24-hour rolling window).<br>The current number can be seen in the `DailyBulkV2QueryJobs` value in the response to the `/vXX.X/limits/` REST API method. | 10,000 |
| The maximum query result size that can be stored in a 24 hour rolling window.<br>The current size can be seen in the `DailyBulkV2QueryFileStorageMB` value in the response to the `/vXX.X/limits/` REST API method. | 1 TB |

# Walkthrough for Query Jobs

This walkthrough shows how to create a query job, monitor its progress, and get the job results.

This walkthrough uses Workbench to issue Bulk API 2.0 calls, but you can use any tool or development environment that can make REST requests.

1. Create the job.
   a. In Workbench, log in to your org.
   b. Go to **Utilities** > **REST Explorer**. Create a query job by issuing a POST request with the following details:

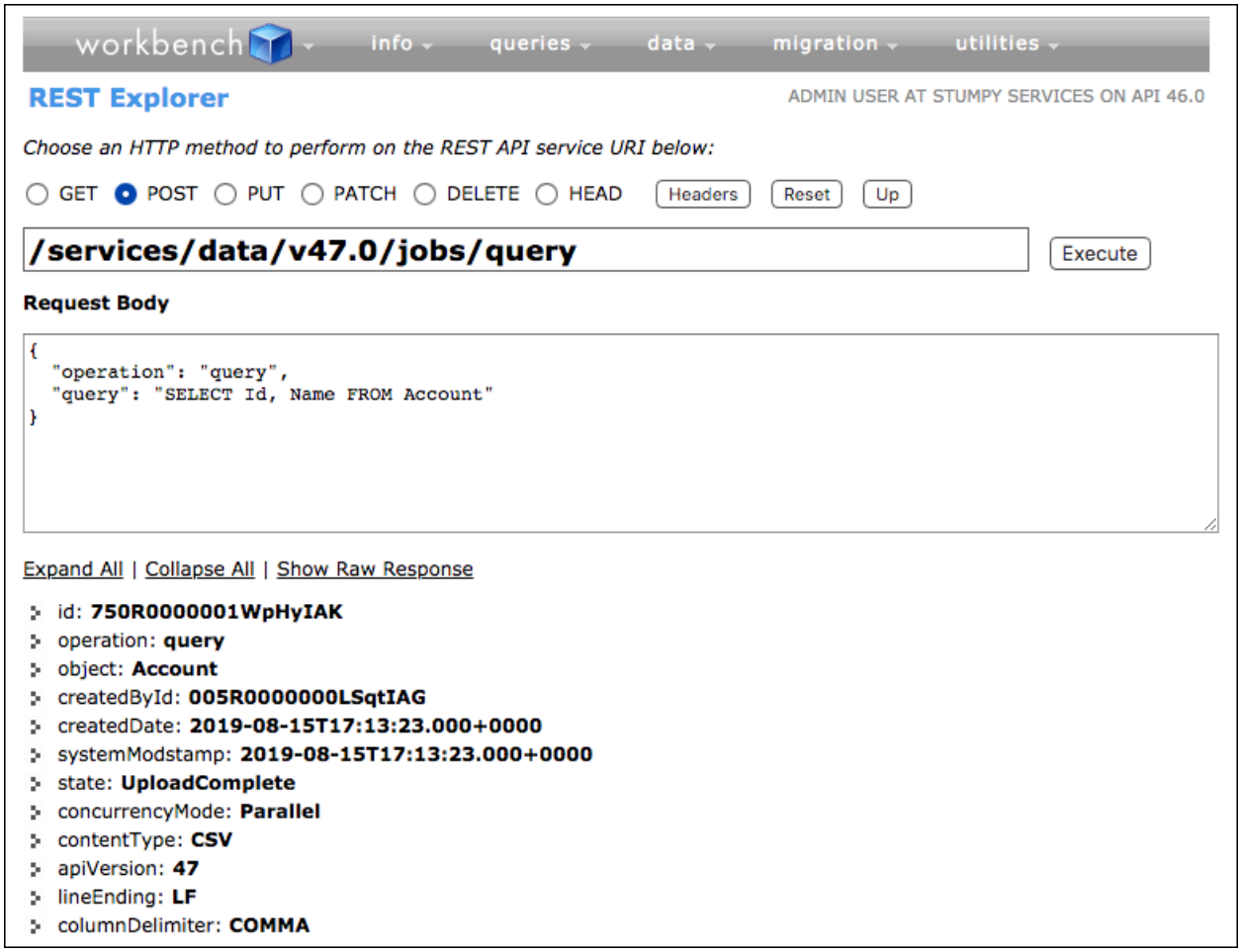
      URI:

      ```
      /services/data/vXX.X/jobs/query
      ```

      Request Body:

      ```
      {
        "operation": "query",
        "query": "SELECT Id, Name FROM Account"
      }
      ```

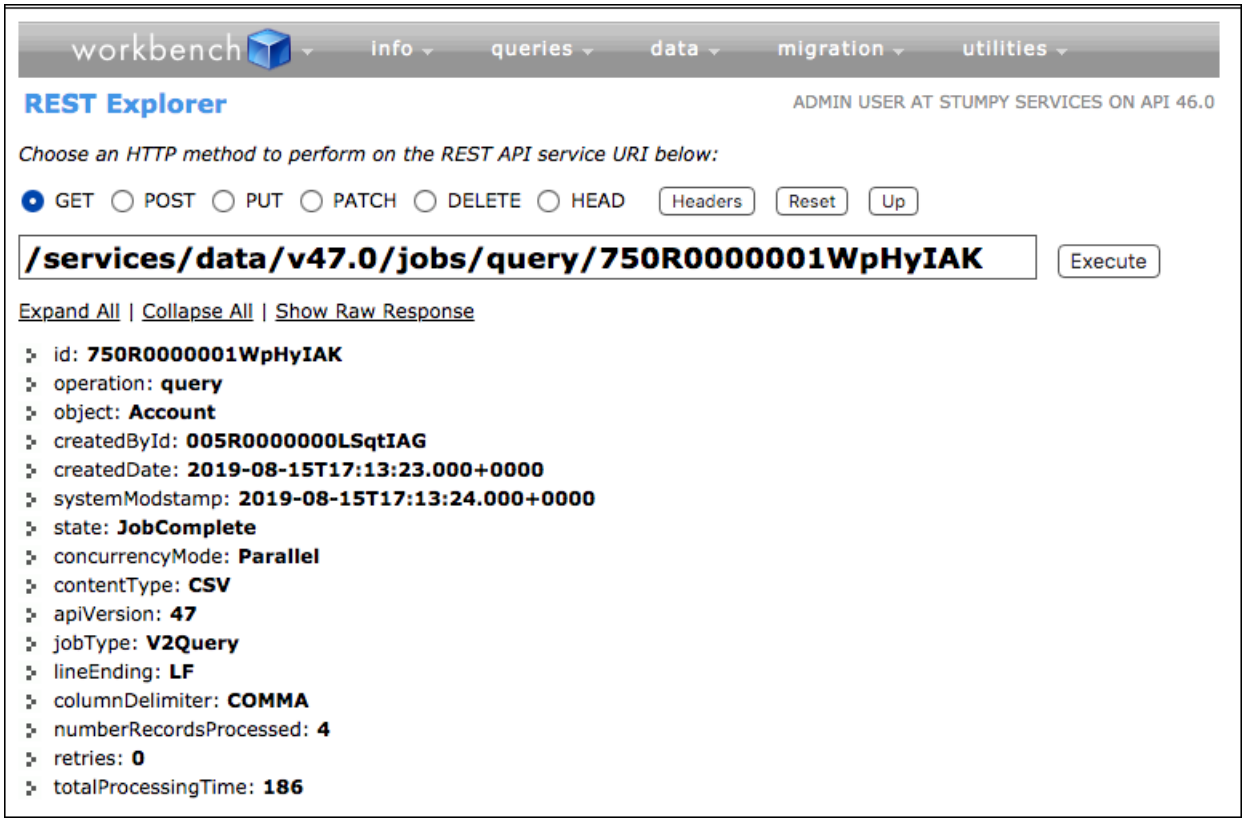
      The response includes the job ID and shows the job's state to be `UploadComplete`. (You will use the job ID to monitor the job or get its results.)



2. To monitor the state of the job, create a GET request. *jobId* is the job ID that was returned by the POST request that created the query job.

   ```
   /services/data/vXX.X/jobs/query/jobId
   ```

   The response shows the current state of the job.



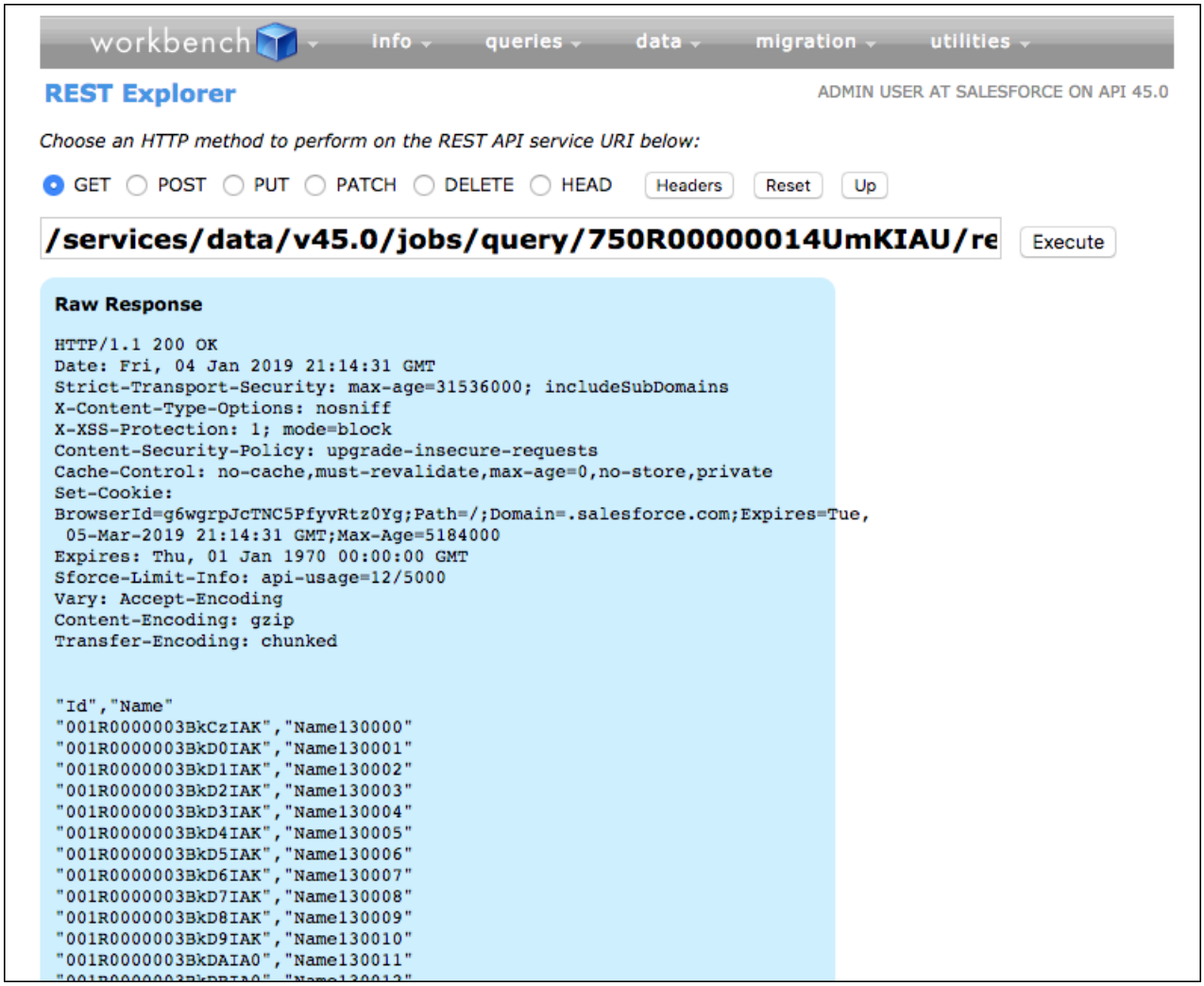

   Repeat this step until the state is `JobComplete`.

3. To get the results of the job, create a GET request with the following details:

   URI:

   ```
   /services/data/vXX.X/jobs/query/jobId/results
   ```

   The response shows the results of the SOQL query run when you created the query job.




# Create a Query Job

Creates a query job.

## Syntax

**URI**

`/services/data/v`***XX.X***`/jobs/query`

**Available since release**

This resource is available in API version 47.0 and later.

**Format**

application/json

**HTTP method**

POST

**Authentication**

Authorization:Bearer *token*

**Headers**

Optionally, use the `Sforce-Call-Options` header to specify a default namespace.

**Request body**

The request body specifies the query to be performed.

```
{
  "operation": "query",
  "query": "SELECT Id FROM Account"
}
```

Note: Bulk API 2.0 does not support queries with any of the following:

- GROUP BY, OFFSET, or TYPEOF clauses.
- Aggregate Functions such as `COUNT()`.
- Date functions in GROUP BY clauses. (Date functions in WHERE clauses are supported.)
- Compound address fields or compound geolocation fields.
- Parent-to-child relationship queries. (Child-to-parent relationship queries are supported.)

You can also specify some optional parameters. For example:

```
{
  "operation" : "query",
  "query" : "SELECT Id FROM Account",
  "contentType" : "CSV",
  "columnDelimiter" : "CARET",
  "lineEnding" : "CRLF"
}
```

**Request parameters**

| Parameter | Description | Required or Optional |
|---|---|---|
| `operation` | The type of query. Possible values are:<br>• `query`—Returns data that has not been deleted or archived. For more information, see query() in the *SOAP API Developer Guide*.<br>• `queryAll`—Returns records that have been deleted because of a merge or delete, and returns information about archived Task and Event records. For more information, see queryAll() in the *SOAP API Developer Guide*. | Required |
| `query` | The query to be performed. | Required |
| `contentType` | The format to be used for the results. Currently the only supported value is `CSV` (comma-separated variables). Defaults to `CSV`.<br>Note:<br>The actual separator may not be a comma. The `columnDelimiter` parameter specifies what character to use. | Optional |

| Parameter | Description | Required or Optional |
|---|---|---|
| `columnDelimiter` | The column delimiter used for CSV job data. The default value is `COMMA`. Possible values are:<br>• `BACKQUOTE`—backquote character (`)<br>• `CARET`—caret character (^)<br>• `COMMA`—comma character (,)<br>• `PIPE`—pipe character (\|)<br>• `SEMICOLON`—semicolon character (;)<br>• `TAB`—tab character | Optional |
| `lineEnding` | The line ending used for CSV job data, marking the end of a data row. The default is `LF`. Possible values are:<br>• `LF`—linefeed character<br>• `CRLF`—carriage return character followed by a linefeed character | Optional |

**Response Body**

```
{
   "id" : "750R0000000zlh9IAA",
   "operation" : "query",
   "object" : "Account",
   "createdById" : "005R0000000GiwjIAC",
   "createdDate" : "2018-12-10T17:50:19.000+0000",
   "systemModstamp" : "2018-12-10T17:50:19.000+0000",
   "state" : "UploadComplete",
   "concurrencyMode" : "Parallel",
   "contentType" : "CSV",
   "apiVersion" : 46.0,
   "lineEnding" : "LF",
   "columnDelimiter" : "COMMA"
}
```

**Response Parameters**

| Parameter | Description |
|---|---|
| `id` | The unique ID for this job. |
| `operation` | The type of query. |
| `object` | The object type being queried. |
| `createdById` | The ID of the user who created the job. |
| `createdDate` | The UTC date and time when the job was created. |
| `systemModstamp` | The UTC date and time when the API last updated the job information. |

| Parameter | Description |
|---|---|
| `state` | The current state of processing for the job. Possible values are:<br>• `UploadComplete`—All job data has been uploaded and the job is ready to be processed. Salesforce has put the job in the queue.<br>• `InProgress`—Salesforce is processing the job.<br>• `Aborted`—The job has been aborted. See Abort a Query Job.<br>• `JobComplete`—Salesforce has finished processing the job.<br>• `Failed`—The job failed. |
| `concurrencyMode` | Reserved for future use. How the request is processed. Currently only parallel mode is supported. (When other modes are added, the API will choose the mode automatically. The mode will not be user configurable.) |
| `contentType` | The format to be used for the results. Currently the only supported value is `CSV`. |
| `apiVersion` | The API version that the job was created in. |
| `lineEnding` | The line ending used for CSV job data, marking the end of a data row. |
| `columnDelimiter` | The column delimiter used for CSV job data. |

# Example

This example creates a job that queries Accounts.

```
curl --include --request POST \
--header "Authorization: Bearer token" \
--header "Accept: application/json " \
--header "Content-Type: application/json" \
--data '{
  "operation": "query",
  "query": "SELECT Id, Name FROM Account"
}' \
https://instance.salesforce.com/services/data/vXX.X/jobs/query
```

The response is:

```
HTTP/1.1 200 OK
{
   "id" : "750R0000000zw4yIAA",
   "operation" : "query",
   "object" : "Account",
   "createdById" : "005R0000000GiwjIAC",
   "createdDate" : "2018-12-17T21:00:17.000+0000",
```

```
   "systemModstamp" : "2018-12-17T21:00:17.000+0000",
   "state" : "UploadComplete",
   "concurrencyMode" : "Parallel",
   "contentType" : "CSV",
   "apiVersion" : 46.0,
   "lineEnding" : "LF",
   "columnDelimiter" : "COMMA"
}
```

# Get Information About a Query Job

Gets information about one query job.

## Syntax

**URI**

`/services/data/v`***XX.X***`/jobs/query/`***queryJobId***

**Available since release**

This resource is available in API version 47.0 and later.

**Formats**

JSON

**HTTP methods**

GET

**Authentication**

Authorization: Bearer *token*

**Request parameters**

| Parameter | Description | Required or Optional |
|---|---|---|
| `queryJobId` | The ID of the query job. | Required |

**Response Body**

```
{
   "id" : "750R0000000zlh9IAA",
   "operation" : "query",
   "object" : "Account",
   "createdById" : "005R0000000GiwjIAC",
   "createdDate" : "2018-12-10T17:50:19.000+0000",
   "systemModstamp" : "2018-12-10T17:51:27.000+0000",
   "state" : "JobComplete",
   "concurrencyMode" : "Parallel",
   "contentType" : "CSV",
   "apiVersion" : 46.0,
   "jobType" : "V2Query",
   "lineEnding" : "LF",
```

```
   "columnDelimiter" : "COMMA",
   "numberRecordsProcessed" : 500,
   "retries" : 0,
   "totalProcessingTime" : 334
}
```

**Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| `id` | string | The unique ID for this job. |
| `operation` | OperationEnum | The type of query. Possible values are:<br>• `query`—Returns data that has not been deleted or archived. For more information, see query() in the *SOAP API Developer Guide*.<br>• `queryAll`—Returns records that have been deleted because of a merge or delete, and returns information about archived Task and Event records. For more information, see queryAll() in the *SOAP API Developer Guide*. |
| `object` | string | The object type being queried. |
| `createdById` | string | The ID of the user who created the job. |
| `createdDate` | dateTime | The UTC date and time when the job was created. |
| `systemModstamp` | dateTime | The UTC date and time when the API last updated the job information. |
| `state` | JobStateEnum | The current state of processing for the job. Possible values are:<br>• `UploadComplete`—All job data has been uploaded and the job is ready to be processed. Salesforce has put the job in the queue.<br>• `InProgress`—Salesforce is processing the job.<br>• `Aborted`—The job has been aborted. See Abort a Query Job.<br>• `JobComplete`—Salesforce has finished processing the job.<br>• `Failed`—The job failed. |
| `concurrencyMode` | ConcurrencyModeEnum | Reserved for future use. How the request is processed. Currently only parallel mode |

| Parameter | Type | Description |
|---|---|---|
| | | is supported. (When other modes are added, the API will choose the mode automatically. The mode will not be user configurable.) |
| `contentType` | ContentType | The format that is used for the results. Currently the only supported value is `CSV`. |
| `apiVersion` | string | The API version that the job was created in. |
| `jobType` | JobTypeEnum | The job's type. For a query job, the type is always `V2Query`. |
| `lineEnding` | LineEndingEnum | The line ending used for CSV job data, marking the end of a data row. The default is `LF`. Possible values are:<br>• `LF`—linefeed character<br>• `CRLF`—carriage return character followed by a linefeed character |
| `columnDelimiter` | ColumnDelimiterEnum | The column delimiter used for CSV job data. The default value is `COMMA`. Possible values are:<br>• `BACKQUOTE`—backquote character (`)<br>• `CARET`—caret character (^)<br>• `COMMA`—comma character (,)<br>• `PIPE`—pipe character (\|)<br>• `SEMICOLON`—semicolon character (;)<br>• `TAB`—tab character |
| `numberRecordsProcessed` | long | The number of records processed in this job. |
| `retries` | int | The number of times that Salesforce attempted to save the results of an operation. Repeated attempts indicate a problem such as a lock contention. |
| `totalProcessingTime` | long | The number of milliseconds taken to process the job. |

**Response Body - For an Unsuccessful Request**

If the request fails, the server returns a non-200 status, and the request body shows details of the error. For example, if the job has been deleted the status is 404 (Not Found) and the response body is:

```
[{
 "errorCode": "NOT_FOUND",
 "message": "The requested resource does not exist"
}]
```

## Example

This example gets information about the job with ID `750R0000000zxikIAA`:

```
curl --include --request GET \
--header "Authorization: Bearer token" \
"https://instance.salesforce.com/services/data/vXX.X/jobs/query/750R0000000zxikIAA
```

The response is:

```
{
   "id" : "750R0000000zxikIAA",
   "operation" : "query",
   "object" : "Account",
   "createdById" : "005R0000000GiwjIAC",
   "createdDate" : "2018-12-18T22:51:36.000+0000",
   "systemModstamp" : "2018-12-18T22:51:58.000+0000",
   "state" : "JobComplete",
   "concurrencyMode" : "Parallel",
   "contentType" : "CSV",
   "apiVersion" : 46.0,
   "jobType" : "V2Query",
   "lineEnding" : "LF",
   "columnDelimiter" : "COMMA",
   "numberRecordsProcessed" : 740003,
   "retries" : 0,
   "totalProcessingTime" : 21046
}
```

# Get Results for a Query Job

Gets the results for a query job. The job must have the state `JobComplete`.

## Syntax

**URI**

`/services/data/vXX.X/jobs/query/queryJobId/results`

or

```
/services/data/vXX.X/jobs/query/queryJobId/results
   ?locator=locator
   &maxRecords=maxRecords
```

Note: Use the same API version to get query results that you used to create the query. Otherwise, the call returns a 409 error.

**Available since release**

This resource is available in API version 47.0 and later.

**Formats**

CSV

**HTTP methods**

GET

**Authentication**

Authorization: Bearer *token*

**Request parameters**

| Parameter | Description | Required or Optional |
|---|---|---|
| `queryJobId` | The ID of the query job. | Required |
| `locator` | A string that identifies a specific set of query results. Providing a value for this parameter returns only that set of results.<br>Omitting this parameter returns the first set of results.<br>You can find the locator string for the next set of results in the response of each request. See Example and Rules and Guidelines.<br>As long as the associated job exists, the locator string for a set of results does not change. You can use the locator to retrieve a set of results multiple times. | Optional |
| `maxRecords` | The maximum number of records to retrieve per set of results for the query. The request is still subject to the size limits.<br>If you are working with a very large number of query results, you may experience a timeout before receiving all the data from Salesforce. To prevent a timeout, specify the maximum number of records your client is expecting to receive in the `maxRecords` parameter. This splits the results into smaller sets with this value as the maximum size.<br>If you don't provide a value for this parameter, the server uses a default value based on the service. | Optional |

**Response Body**

If the request is successful, the status code is 200 (OK) and the request body contains the results of the job's query. For example:

```
"Id","Name"
"005R0000000UyrWIAS","Jane Dunn"
"005R0000000GiwjIAC","George Wright"
"005R0000000GiwoIAC","Pat Wilson"
...
```

Note: In API version 50.0 and later, the order of the columns returned by the query is the same as the order you requested them. In version 49.0 and earlier, the order of the columns is returned alphabetically.

**Response Headers**

| Header | Description |
|---|---|
| `Sforce-NumberOfRecords` | The number of records in this set. |
| `Sforce-Locator` | The locator for the next set of results (if there are any). Use this value in other `GET` request to retrieve the next set of query results.<br>This value is a pseudo random string (for example, `MTAwMDA`). The length of this string varies depending on how many sets of results there are.<br>If there are no more sets of query results, this value is the string 'null'.<br>See Example and Rules and Guidelines. |

# Example

This example retrieves the results for the job with ID `750R0000000zxr8IAA`. It also shows how to use the `locator` and `maxRecords` query parameters. (In this example, we use them both, but they are independent. You don't have to use them together.)

We start by sending an initial request to retrieve the first set of query results. We don't use the `locator` parameter because we want to get the first set of results.

```
curl --include --request GET \
--header "Authorization: Bearer token" \
--header "Accept: text/csv" \
https://instance.salesforce.com/services/data/vXX.X/jobs/query/750R0000000zxr8IAA/results
?maxRecords=50000
```

Note: The `Accept` header must match what was specified when the job was created. Currently, only `text/csv` is supported.

The response body is:

```
HTTP/1.1 200 OK
...
Sforce-Locator: MTAwMDA
Sforce-NumberOfRecords: 50000
...

"Id","Name"
"005R0000000UyrWIAS","Jane Dunn"
"005R0000000GiwjIAC","George Wright"
"005R0000000GiwoIAC","Pat Wilson"
...
```

The response includes `MTAwMDA` as a value for the `Sforce-Locator` header. This value is not 'null', which means that there are more query results that we can retrieve.

To retrieve the next set of query results, we send another request, using the `locator` parameter and the locator string, `MTAwMDA`.

```
curl --include --request GET \
--header "Accept: text/csv" \
https://instance.salesforce.com/services/data/vXX.X/jobs/query/750R0000000zxr8IAA/results
?locator=MTAwMDA&maxRecords=50000
```

The response body is:

```
HTTP/1.1 200 OK
...
Sforce-Locator: MjAwMDAw
Sforce-NumberOfRecords: 50000
...

"Id","Name"
"005R0000000UyrWIAv","James Wu"
"005R0000000GiwjIxx","Samantha Jones"
"005R0000000GiwoIAB","Doug West"
...
```

Notice that the locator value has changed. This means that there is another set of query results that we can retrieve. We use the new locator string, `MjAwMDAw`, in another request.

```
curl --include --request GET \
--header "Accept: text/csv" \
https://instance.salesforce.com/services/data/vXX.X/jobs/query/750R0000000zxr8IAA/results
?locator=MjAwMDAw&maxRecords=50000
```

We repeat this process until the value of the `Sforce-Locator` header is 'null', which indicates that there are no more results to retrieve.

```
HTTP/1.1 200 OK

Sforce-Locator: null
Sforce-NumberOfRecords: 6155
...

"Id","Name"
...
```

# Rules and Guidelines

To retrieve your full set of query results, follow these rules and guidelines.

1. Use `/services/data/v`***`XX.X`***`/jobs/query/`***`queryJobId`***`/results` to get the first set of results for the job.
2. If there are no more results, the `Sforce-Locator` header's value is the string 'null'. Otherwise, set the `locator` query parameter to that value to get the next set of results.

   Note: For `locator`, use only the value from the `Sforce-Locator` header. Don't try to guess what it is. How this parameter is evaluated is subject to change.

3. Repeat this process until the `Sforce-Locator` header's value is the string 'null'. That set is the last set of results.

# Abort a Query Job

Aborts a query job.

Note:

- To abort a job, you must be the job's creator or have the Manage Data Integrations permission.
- You can only abort jobs that are in the following states:
  - `UploadComplete`
  - `InProgress`

## Syntax

**URI**

`/services/data/v`***`XX.X/`***`jobs/query/`***`queryJobId`***

**Available since release**

This resource is available in API version 47.0 and later.

**Formats**

JSON

**HTTP methods**

PATCH

**Authentication**

Authorization: Bearer *`token`*

**Request body**

The request body must be the following:

```
{
  "state": "Aborted"
}
```

**Request parameters**

| Parameter | Description | Required or Optional |
|---|---|---|
| `queryJobId` | The ID of the query job to be deleted. | Required |

**Response Body**

If the request is successful, the response is similar to that for Get Results for a Query Job but the `state` is `Aborted`. For example:

```
{
   "id" : "750R000000146UvIAI",
   "operation" : "query",
   "object" : "Account",
   "createdById" : "005R0000000GiwjIAC",
   "createdDate" : "2018-12-18T16:15:31.000+0000",
   "systemModstamp" : "2018-12-18T16:15:32.000+0000",
```

```
   "state" : "Aborted",
   "concurrencyMode" : "Parallel",
   "contentType" : "CSV",
   "apiVersion" : 46.0
}
```

**Response Parameters**

| Parameter | Description |
|---|---|
| `id` | The unique ID for this job. |
| `operation` | The type of query. Possible values are:<br>• `query`—Returns data that has not been deleted or archived. For more information, see query() in the *SOAP API Developer Guide*.<br>• `queryAll`—Returns records that have been deleted because of a merge or delete, and returns information about archived Task and Event records. For more information, see queryAll() in the *SOAP API Developer Guide*. |
| `object` | The object type being queried. |
| `createdById` | The ID of the user who created the job. |
| `createdDate` | The UTC date and time when the job was created. |
| `systemModstamp` | The UTC date and time when the API last updated the job information. |
| `state` | The current state of processing for the job. Possible values are:<br>• `UploadComplete`—All job data has been uploaded and the job is ready to be processed. Salesforce has put the job in the queue.<br>• `InProgress`—Salesforce is processing the job.<br>• `Aborted`—The job has been aborted. See Abort a Query Job.<br>• `JobComplete`—Salesforce has finished processing the job.<br>• `Failed`—The job failed. |
| `concurrencyMode` | Reserved for future use. How the request is processed. Currently only parallel mode is supported. (When other modes are added, the API will choose the mode automatically. The mode will not be user configurable.) |
| `contentType` | The format that is used for the results. Currently the only supported value is `CSV`. |
| `apiVersion` | The API version that the job was created in. |

**Response Body - For an Unsuccessful Request**

If the request fails, the server returns a non-200 status, and the request body shows details of the error. For example:

```
HTTP/1.1 400 Bad Request
[{
 "errorCode": "INVALIDJOBSTATE",
 "message": "Aborting already Completed Job not allowed"
}]
```

## Example

This example aborts the job with ID `750R000000146UvIAI`:

```
curl  --request PATCH \
--header "Authorization: Bearer token" \
--header "Content-Type: application/json" \
--data '{
  "state": "Aborted"
}' \
https://instance.salesforce.com/services/data/vXX.X/jobs/query/750R000000146UvIAI
```

The response is:

```
{
  "id": "750R000000146UvIAI",
  "operation": "query",
  "object": "Account",
  "createdById": "005R0000000GiwjIAC",
  "createdDate": "2018-12-18T20:51:39.000+0000",
  "systemModstamp": "2018-12-18T20:51:41.000+0000",
  "state": "Aborted",
  "concurrencyMode": "Parallel",
  "contentType": "CSV",
  "apiVersion": 46.0
}
```

# Delete a Query Job

Deletes a query job. When a job is deleted, job data stored by Salesforce is deleted and job metadata information is removed. The job will no longer display in the Bulk Data Load Jobs page in Salesforce.

Note: You can only delete a job if its state is `JobComplete`, `Aborted`, or `Failed`.

## Syntax

**URI**

`/services/data/vXX.X/jobs/query/queryJobId`

**Available since release**

This resource is available in API version 47.0 and later.

**Formats**
None

**HTTP methods**
DELETE

**Authentication**
Authorization: Bearer *token*

**Request parameters**

| Parameter | Description | Required or Optional |
|---|---|---|
| queryJobId | The ID of the query job to be deleted. | Required |

**Response Body**
If the method is successful, the status code is 204 (No Content) and there is no response body.

**Response Body - For an Unsuccessful Request**
If the request fails, the server returns a 400 (Bad Request) status, and the request body shows details of the error. For example:

```
HTTP/1.1 400 Bad Request
[{
 "errorCode": "API_ERROR",
 "message": "Error encountered when deleting the job because the job is not terminated"
}]
```

## Example

This example deletes the job with ID 750R0000000zxnaIAA.

```
curl --include --request DELETE \
--header "Authorization: Bearer token \
--header "Content-Type: " \
https://instance.salesforce.com/services/data/vXX.X/jobs/query/750R0000000zxnaIAA
```

The response status is

```
204 No Content
```

# Get Information About All Query Jobs

Gets information about all query jobs in the org. The information includes Bulk API 2.0 query jobs and all Bulk API jobs.

## Syntax

**URI**
/services/data/v*XX.X*/jobs/query

```
/services/data/vXX.X/jobs/query/
   ?isPkChunkingEnabled=isPkChunkingEnabled
```

```
&jobType=jobType
&concurrencyMode=concurrencyMode
&queryLocator=queryLocator
```

**Available since release**

This resource is available in API version 47.0 and later.

**Formats**

JSON

**HTTP methods**

GET

**Authentication**

Authorization: Bearer *token*

**Request parameters**

| Parameter | Description | Required or Optional |
|---|---|---|
| `isPkChunkingEnabled` | If set to true, the request only returns information about jobs where PK Chunking is enabled. This only applies to Bulk API (not Bulk API 2.0) jobs.<br>For more information on PK Chunking, see Use PK Chunking to Extract Large Data Sets from Salesforce. | Optional |
| `jobType` | Gets information only about jobs matching the specified job type. Possible values are:<br>• `Classic`—Bulk API jobs. This includes both query jobs and ingest jobs.<br>• `V2Query`—Bulk API 2.0 query jobs.<br>• `V2Ingest`—Bulk API 2.0 ingest (upload and upsert) jobs. | Optional |
| `concurrencyMode` | For future use. Gets information only about jobs matching the specified concurrency mode. Possible values are `serial` and `parallel`.<br>Note: Currently only parallel mode is supported. | Optional |
| `queryLocator` | Gets information about jobs starting with that locator value.<br>Note: Do not enter your own value here. Always use the value from the `nextRecordsUrl` from the previous set.<br>See Example and Rules and Guidelines. | Optional |

**Response Body**

The response contains a completion flag, an array of records, and a locator value to be used to obtain more records. For example:

```
{
   "done" : false,
   "records" : [
```

```
      {
         "id" : "750R0000000zhfdIAA",
         "operation" : "query",
         "object" : "Account",
         "createdById" : "005R0000000GiwjIAC",
         "createdDate" : "2018-12-07T19:58:09.000+0000",
         "systemModstamp" : "2018-12-07T19:59:14.000+0000",
         "state" : "JobComplete",
         "concurrencyMode" : "Parallel",
         "contentType" : "CSV",
         "apiVersion" : 51.0,
         "jobType" : "V2Query",
         "lineEnding" : "LF",
         "columnDelimiter" : "COMMA"
      },
      {
         "id" : "750R0000000zhjzIAA",
         "operation" : "query",
         "object" : "Account",
         "createdById" : "005R0000000GiwjIAC",
         "createdDate" : "2018-12-07T20:52:28.000+0000",
         "systemModstamp" : "2018-12-07T20:53:15.000+0000",
         "state" : "JobComplete",
         "concurrencyMode" : "Parallel",
         "contentType" : "CSV",
         "apiVersion" : 51.0,
         "jobType" : "V2Query",
         "lineEnding" : "LF",
         "columnDelimiter" : "COMMA"
      },
     ...
   ],
   "nextRecordsUrl" :
"/services/data/v51.0/jobs/ingest?queryLocator=01gR0000000opRTIAY-2000"
}
```

**Response Parameters**

| Parameter | Description |
|---|---|
| `done` | This is `true` if this is the last (or only) set of results. It is `false` if there are more records to fetch. See Example and Rules and Guidelines. |
| `records` | An array of record objects. |
| `nextRecordsUrl` | The URI to get the next set of records (if there are any).<br>This method returns up to 1,000 result rows per request. If there are more than 1,000 records, use the `nextRecordsUrl` to get the next set of records. See Example and Rules and Guidelines.<br>This parameter is `null` if there are no more records to fetch. |

**Record Objects**

| Parameter | Description |
|---|---|
| `id` | The unique ID for this job. |
| `operation` | The type of query. Possible values are:<br>• `query`—Returns data that has not been deleted or archived. For more information, see query() in the *SOAP API Developer Guide*.<br>• `queryAll`—Returns records that have been deleted because of a merge or delete, and returns information about archived Task and Event records. For more information, see queryAll() in the *SOAP API Developer Guide*. |
| `object` | The object type being queried. |
| `createdById` | The ID of the user who created the job. |
| `createdDate` | The UTC date and time when the job was created. |
| `systemModstamp` | The UTC date and time when the API last updated the job information. |
| `state` | The current state of processing for the job. Possible values are:<br>• `UploadComplete`—All job data has been uploaded and the job is ready to be processed. Salesforce has put the job in the queue.<br>• `InProgress`—Salesforce is processing the job.<br>• `Aborted`—The job has been aborted. See Abort a Query Job.<br>• `JobComplete`—Salesforce has finished processing the job.<br>• `Failed`—The job failed. |
| `concurrencyMode` | Reserved for future use. How the request is processed. Currently only parallel mode is supported. (When other modes are added, the API will choose the mode automatically. The mode will not be user configurable.) |
| `contentType` | The format to be used for the results. Currently the only supported value is `CSV` (comma-separated variables). Defaults to `CSV`.<br>Note:<br>The actual separator may not be a comma. The `columnDelimiter` parameter specifies what character to use. |

| Parameter | Description |
| --- | --- |
| `apiVersion` | The API version that the job was created in. |
| `lineEnding` | The line ending used for CSV job data, marking the end of a data row. The default is `LF`. Possible values are:<br>• `LF`—linefeed character<br>• `CRLF`—carriage return character followed by a linefeed character |
| `columnDelimiter` | The column delimiter used for CSV job data. The default value is `COMMA`. Possible values are:<br>• `BACKQUOTE`—backquote character (`)<br>• `CARET`—caret character (^)<br>• `COMMA`—comma character (,)<br>• `PIPE`—pipe character (\|)<br>• `SEMICOLON`—semicolon character (;)<br>• `TAB`—tab character |

## Example

This example shows how to use the `nextRecordsUrl` query parameter.

The first request doesn't use `nextRecordsUrl`, because we don't know what value to use yet.

```
curl --request GET \
--header "Authorization: Bearer token"
https://instance.salesforce.com/services/data/vXX.X/jobs/query
```

The response body is:

```
{
    "done": false,
    "nextRecordsUrl":
"/services/data/vXX.X/jobs/ingest?queryLocator=01gRM000000NS1vYAG-1000",
    "records": [
        {
           ...
        }
   ]
}
```

Because there are more records than can be returned in a single response, the value of `done` in the response isn't `true`. We use the value of `nextRecordsUrl`, `/services/data/vXX.X/jobs/ingest?queryLocator=01gRM000000NS1vYAG-1000`, as the URI to get the next set of records:

```
curl --request GET \
--header "Authorization: Bearer token"
https://instance.salesforce.com/services/data/vXX.X/jobs/query?queryLocator=01gRM000000NS1vYAG-1000
```

Repeat this process until `done` is `true`, indicating that there are no more results to fetch.

## Rules and Guidelines

To use this URI, follow these rules and guidelines.

1. Use `/services/data/v`***`XX.X`***`/jobs/query` to get the first set of records.
2. If there are more records than can be listed, the response body has `done` set to `false`. Use the value of `nextRecordsUrl` to get the next set of records.
3. Repeat this process until `done` is `true`. That set is the last set of records.