Skip to main content Documentation Technology areas close AI and ML Application development Application hosting Compute Data analytics and pipelines Databases Distributed, hybrid, and multicloud Generative AI Industry solutions Networking Observability and monitoring Security Storage Cross-product tools close Access and resources management Costs and usage management Google Cloud SDK, languages, frameworks, and tools Infrastructure as code Migration Related sites close Google Cloud Home Free Trial and Free Tier Architecture Center Blog Contact Sales Google Cloud Developer Center Google Developer Center Google Cloud Marketplace Google Cloud Marketplace Documentation Google Cloud Skills Boost Google Cloud Solution Center Google Cloud Support Google Cloud Tech Youtube Channel / English Deutsch Español – América Latina Français Indonesia Italiano Português – Brasil – Console Sign in BigQuery Guides Reference Samples Resources Contact Us Start free Documentation Guides Reference Samples Resources Technology areas More Cross-product tools More Related sites More Console Contact Us Start free Discover Product overview How does BigQuery work? Storage Analytics Administration Get started Use the BigQuery sandbox Quickstarts Try the Cloud console Query public data Load and query data Enable asset management Try the command-line tool Query public data Load and query data Try the client libraries Explore BigQuery tools Explore the console Explore the command-line tool Migrate Overview Migrate a data warehouse Introduction to BigQuery Migration Service Migration assessment Migrate schema and data Migrate data pipelines Migrate SQL Translate SQL queries interactively Translate SQL queries using the API Translate SQL queries in batch Generate metadata for translation and assessment Transform SQL translations with YAML Map SQL object names for batch translation Migration guides Amazon Redshift Migration overview Migrate Amazon Redshift schema and data Migrate Amazon Redshift schema and data when using a VPC SQL translation reference Apache Hive Migration overview Migrate Apache Hive schema and data SQL translation reference IBM Netezza Migrate from IBM Netezza SQL translation reference Oracle Migration guide SQL translation reference Snowflake Migration guide SQL translation reference Teradata Migration overview Migrate Teradata schema and data Migration tutorial SQL translation reference Design Organize resources API dependencies Understand editions Datasets Introduction Create datasets List datasets Cross-region replication Managed disaster recovery Migrate to managed disaster recovery Dataset data retention Tables BigQuery tables Introduction Create and use tables BigQuery tables for Apache Iceberg Specify table schemas Specify a schema Specify nested and repeated columns Specify default column values Segment with partitioned tables Introduction Create partitioned tables Manage partitioned tables Query partitioned tables Optimize with clustered tables Introduction Create and use clustered tables Query clustered tables External tables Introduction Types of external tables BigLake external tables BigQuery Omni Object tables External tables External table definition file Externally partitioned data Use metadata caching Amazon S3 BigLake external tables BigLake external tables for Apache Iceberg Azure Blob Storage BigLake tables Bigtable external table BigLake external tables for Cloud Storage Cloud Storage object tables Cloud Storage external tables Delta Lake BigLake tables Google Drive external tables Views Logical views Introduction Create logical views Materialized views Introduction Create materialized views Create materialized view replicas Manage all view types Get information about views Manage views Routines Introduction Manage routines User-defined

functions User-defined functions in Python User-defined aggregate functions Table functions Remote functions SQL stored procedures Stored procedures for Apache Spark Analyze object tables by using remote functions Remote functions and Translation API tutorial Connections Introduction Amazon S3 connection Apache Spark connection Azure Blob Storage connection Cloud resource connection Spanner connection Cloud SQL connection AlloyDB connection SAP Datasphere connection Manage connections Configure connections with network attachments Default connections Indexes Search indexes Introduction Manage search indexes Vector indexes Introduction Manage vector indexes Load, transform, and export Introduction Load data Introduction BigQuery Data Transfer Service Introduction Data location and transfers Authorize transfers Enable transfers Set up network connections Cloud SQL instance access AWS VPN and network attachment Azure VPN and network attachment Manage transfers Transfer run notifications Troubleshoot transfer configurations Use service accounts Use third-party transfers Use custom organization policies Data source change log Transfer guides Amazon S3 Introduction Schedule transfers Transfer runtime parameters Azure Blob Storage Introduction Schedule transfers Transfer runtime parameters Campaign Manager Schedule transfers Report transformation Cloud Storage Introduction Schedule transfers Transfer runtime parameters Comparison Shopping Service Center Introduction Schedule transfers Transfer report schema Display & Video 360 Schedule transfers Report transformation Facebook Ads Schedule transfers Report transformation Google Ad Manager Schedule transfers Report transformation Google Ads Schedule transfers Report transformation Google Analytics 4 Schedule transfers Report transformation Google Merchant Center Introduction Schedule transfers Query your data Migration guides Best sellers Price competitiveness Transfer report schema Best Sellers table Local Inventories table Performance table Price Benchmarks table Price Competitiveness table Price Insights table Product Inventory table Product Targeting table Products table Regional Inventories table Top Brands table Top Products table Google Play Schedule transfers Transfer report transformation MySQL Schedule transfers Oracle Schedule transfers PostgreSQL Schedule transfers Salesforce Schedule transfers Salesforce Marketing Cloud Schedule transfers Search Ads 360 Schedule transfers Transfer report transformation Migration guide ServiceNow Schedule transfers YouTube channel Schedule transfers Transfer report transformation YouTube content owner Schedule transfers Transfer report transformation Batch load data Introduction Auto-detect schemas Load Avro data Load Parquet data Load ORC data Load CSV data Load JSON data Load externally partitioned data Load data from a Datastore export Load data from a Firestore export Load data using the Storage Write API Load data into partitioned tables Write and read data with the Storage API Read data with the Storage Read API Write data with the Storage Write API Introduction Stream data with the Storage Write API Batch load data with the Storage Write API Best practices Supported protocol buffer and Arrow data types Stream updates with change data capture Use the legacy streaming API Load data from other Google services Discover and catalog Cloud Storage data Load data using third-party apps Load data using cross-cloud operations Transform data Introduction Prepare data Introduction Prepare data with Gemini Transform with DML Transform data in partitioned tables Work with change history Transform data with pipelines Introduction Create pipelines Export data Introduction Export query results Export to Cloud Storage Export to Bigtable Export to Spanner Export to Pub/Sub Export as Protobuf

columns Analyze Introduction Explore your data Create queries with table explorer Generate profile insights Generate data insights Analyze with a data canvas Analyze data with Gemini Query BigQuery data Run a query Write queries with Gemini Write query results Query data with SQL Introduction Arrays JSON data Multi-statement queries Parameterized queries Pipe syntax Recursive CTEs Sketches Table sampling Time series Transactions Wildcard tables Use geospatial analytics Introduction Work with geospatial analytics Work with raster data Best practices for spatial analysis Visualize geospatial data Grid systems for spatial analysis Geospatial analytics syntax reference Geospatial analytics tutorials Get started with geospatial analytics Use geospatial analytics to plot a hurricane's path Visualize geospatial analytics data in a Colab notebook Use raster data to analyze temperature Search data Search indexed data Work with text analyzers Access historical data Work with queries Save queries Introduction Create saved queries Continuous queries Introduction Create continuous queries Use cached results Work with sessions Introduction Create sessions Write queries in sessions Run queries in sessions Terminate sessions View query history in sessions Find sessions Troubleshoot queries Optimize queries Introduction Use the query plan explanation Get query performance insights Optimize query computation Use history-based optimizations Optimize storage for query performance Use materialized views Use BI Engine Use nested and repeated data Optimize functions Query external data sources Manage open source metadata BigQuery metastore Introduction Use with Apache Spark and standard tables, BigQuery tables for Apache Iceberg, and external tables Use with Apache Spark in BigQuery Studio Use with Apache Spark or Flink in Dataproc Use with Apache Spark in Dataproc Serverless Use with stored procedures Manage Iceberg resources Create tables with Apache Spark and query in BigQuery Additional features Migrate from Dataproc Metastore BigLake Metastore Use external tables and datasets Amazon S3 data Query Amazon S3 data Export query results to Amazon S3 Query Apache Iceberg data Query open table formats with manifests Azure Blob Storage data Query Azure Blob Storage data Export query results to Azure Blob Storage Query Cloud Bigtable data Cloud Storage data Query Cloud Storage data in BigLake tables Query Cloud Storage data in external tables Work with Salesforce Data Cloud data Query Google Drive data Create AWS Glue federated datasets Create Spanner external datasets Run federated queries Federated queries Query SAP Datasphere data Query AlloyDB data Query Spanner data Query Cloud SQL data Use notebooks Introduction Use Colab notebooks Introduction Create notebooks Explore query results Use DataFrames Introduction Try BigQuery DataFrames Use BigQuery DataFrames Use Jupyter notebooks Use the BigQuery JupyterLab plugin Use managed Jupyter notebooks Use analysis and BI tools Introduction Use Connected Sheets Use Tableau Use Looker Use Looker Studio Use third-party tools Google Cloud Ready - BigQuery Overview Partners AI and machine learning Introduction Generative AI and pretrained models Choose generative AI and task-specific functions Choose a natural language processing function Choose a document processing function Choose a transcription function Generative AI Overview Built-in models The TimesFM time series forecasting model Tutorials Generate text Generate text using public data and Gemini Generate text using public data and Gemma Generate text using your data Generate structured data Handle quota errors by calling ML.GENERATE_TEXT iteratively Analyze images with a Gemini vision model Tune text generation models Tune a model using your data Use tuning and evaluation to improve model

performance Generate embeddings Generate text embeddings using an LLM Generate image embeddings using an LLM Generate video embeddings using an LLM Handle quota errors by calling ML.GENERATE_EMBEDDING iteratively Generate and search multimodal embeddings Generate text embeddings using pretrained TensorFlow models Vector search Search embeddings with vector search Perform semantic search and retrieval-augmented generation Task-specific solutions Overview Tutorials Natural language processing Understand text Translate text Document processing Process documents Parse PDFs in a retrieval-augmented generation pipeline Speech recognition Transcribe audio files Computer vision Annotate images Run inference on image data Analyze images with an imported classification model Analyze images with an imported feature vector model Machine learning ML models and MLOps End-to-end journey per model Model creation Hyperparameter tuning overview Model evaluation overview Model inference overview Explainable AI overview Model weights overview ML pipelines overview Model monitoring overview Manage BigQueryML models in Vertex AI Use cases Forecasting Anomaly detection Recommendation Classification Regression Dimensionality reduction Clustering Tutorials Get started with BigQuery ML Regression and classification Create a linear regression model Create a logistic regression classification model Create a boosted tree classification model Clustering Cluster data with a k-means model Recommendation Create recommendations based on explicit feedback with a matrix factorization model Create recommendations based on implicit feedback with a matrix factorization model Time series forecasting Forecast a single time series with an ARIMA_PLUS univariate model Forecast multiple time series with an ARIMA_PLUS univariate model Forecast time series with a TimesFM univariate model Scale an ARIMA_PLUS univariate model to millions of time series Forecast a single time series with a multivariate model Forecast multiple time series with a multivariate model Use custom holidays with an ARIMA_PLUS univariate model Limit forecasted values for an ARIMA_PLUS univariate model Forecast hierarchical time series with an ARIMA_PLUS univariate model Anomaly detection Anomaly detection with a multivariate time series Imported and remote models Make predictions with imported TensorFlow models Make predictions with scikit-learn models in ONNX format Make predictions with PyTorch models in ONNX format Make predictions with remote models on Vertex AI Hyperparameter tuning Improve model performance with hyperparameter tuning Export models Export a BigQuery ML model for online prediction Augmented analytics Contribution analysis Tutorials Get data insights from contribution analysis using a summable metric Get data insights from contribution analysis using a summable ratio metric Create and manage features Feature preprocessing overview Supported input feature types Automatic preprocessing Manual preprocessing Feature serving Perform feature engineering with the TRANSFORM clause Work with models List models Manage models Get model metadata Update model metadata Export models Delete models Reference patterns Administer Introduction Manage resources Organize resources Understand reliability Manage code assets Manage data preparations Manage notebooks Manage saved queries Manage pipelines Manage tables Manage tables Manage table data Modify table schemas Manage table clones Introduction Create table clones Manage table snapshots Introduction Create table snapshots Restore table snapshots List table snapshots View table snapshot metadata Update table snapshot metadata Delete table snapshots Create periodic table snapshots Manage

configuration settings Manage datasets Manage datasets Update dataset properties Manage materialized views Manage materialized view replicas Schedule resources Introduction Schedule code assets Schedule data preparations Schedule notebooks Schedule pipelines Schedule DAGs Schedule jobs and queries Run jobs programmatically Schedule queries Workload management Introduction Slots Slot reservations Slots autoscaling Use reservations Get started Estimate slot capacity requirements View slot recommendations and insights Purchase and manage slot commitments Work with slot reservations Work with reservation assignments Manage jobs Use query queues Legacy reservations Introduction to legacy reservations Legacy slot commitments Purchase and manage legacy slot commitments Work with legacy slot reservations Manage BI Engine Introduction Reserve BI Engine capacity Monitor workloads Introduction Monitor resource utilization Monitor jobs Monitor sharing listings Monitor BI Engine Monitor data quality Monitor Data Transfer Service Monitor materialized views Monitor reservations Monitor continuous queries Dashboards, charts and alerts Optimize resources Control costs Estimate and control costs Create custom query quotas Optimize with recommendations Introduction Manage cluster and partition recommendations Manage materialized view recommendations Organize with labels Introduction Add labels View labels Update labels Filter using labels Delete labels Manage data quality Monitor data quality with scans Data Catalog overview Work with Data Catalog Govern Introduction Control access to resources Introduction Control access with IAM Manage resource access policies Control access with tags Control access with conditions Control access with authorization Authorized datasets Authorized routines Authorized views Tutorials Create an authorized view Restrict network access Control access with VPC service controls Regional endpoints Control column and row access Control access to table columns Introduction to column-level access control Restrict access with column-level access control Impact on writes Manage policy tags Manage policy tags across locations Best practices for using policy tags Control access to table rows Introduction to row-level security Work with row-level security Use row-level security with other BigQuery features Best practices for row-level security Protect sensitive data Mask data in table columns Introduction to data masking Mask column data Anonymize data with differential privacy Use differential privacy Extend differential privacy Restrict data access using analysis rules Use Sensitive Data Protection Manage encryption Encryption at rest Customer-managed encryption keys Column-level encryption with Cloud KMS AEAD encryption Audit workloads Introduction Audit policy tags View Data Policy audit logs Data Transfer Service audit logs Sharing audit logs BigQuery audit logs reference Migrate audit logs BigLake API audit logs BigQuery Migration API audit logs Share data and AI assets Introduction Manage data exchanges Manage listings Manage subscriptions Configure user roles View and subscribe to listings Share sensitive data with data clean rooms Entity resolution Introduction Use entity resolution VPC Service Controls for Sharing Stream sharing with Pub/Sub Commercialize listings on Cloud Marketplace Develop Introduction BigQuery code samples BigQuery API basics BigQuery APIs and libraries overview Authentication Introduction Get started Authenticate as an end user Authenticate with JSON Web Tokens Run jobs programmatically Paginate with BigQuery API API performance tips Batch requests Repositories Introduction Create repositories Workspaces Introduction Create and use workspaces Use the VS Code extension Choose a Python library Use ODBC and JDBC drivers AI and ML Application development Application hosting Compute Data

analytics and pipelines Databases Distributed, hybrid, and multicloud Generative AI Industry solutions Networking Observability and monitoring Security Storage Access and resources management Costs and usage management Google Cloud SDK, languages, frameworks, and tools Infrastructure as code Migration Google Cloud Home Free Trial and Free Tier Architecture Center Blog Contact Sales Google Cloud Developer Center Google Developer Center Google Cloud Marketplace Google Cloud Marketplace Documentation Google Cloud Skills Boost Google Cloud Solution Center Google Cloud Support Google Cloud Tech Youtube Channel Home BigQuery Documentation Guides Send feedback Stay organized with collections Save and categorize content based on your preferences. API performance tips This document covers some techniques you can use to improve the performance of your application. In some cases, examples from other APIs or generic APIs are used to illustrate the ideas presented. However, the same concepts are applicable to the BigQuery API. Compression using gzip An easy and convenient way to reduce the bandwidth needed for each request is to enable gzip compression. Although this requires additional CPU time to uncompress the results, the trade-off with network costs usually makes it very worthwhile. In order to receive a gzip-encoded response you must do two things: Set an Accept-Encoding header, and modify your user agent to contain the string gzip. Here is an example of properly formed HTTP headers for enabling gzip compression: Accept-Encoding: gzip User-Agent: my program (gzip) Working with partial resources Another way to improve the performance of your API calls is by sending and receiving only the portion of the data that you're interested in. This lets your application avoid transferring, parsing, and storing unneeded fields, so it can use resources including network, CPU, and memory more efficiently. There are two types of partial requests: Partial response: A request where you specify which fields to include in the response (use the fields request parameter). Patch: An update request where you send only the fields you want to change (use the PATCH HTTP verb). More details on making partial requests are provided in the following sections. Partial response By default, the server sends back the full representation of a resource after processing requests. For better performance, you can ask the server to send only the fields you really need and get a partial response instead. To request a partial response, use the fields request parameter to specify the fields you want returned. You can use this parameter with any request that returns response data. Note that the fields parameter only affects the response data; it does not affect the data that you need to send, if any. To reduce the amount of data you send when modifying resources, use a patch request. Example The following example shows the use of the fields parameter with a generic (fictional) "Demo" API. Simple request: This HTTP GET request omits the fields parameter and returns the full resource. https://www.googleapis.com/demo/v1 Full resource response: The full resource data includes the following fields, along with many others that have been omitted for brevity. { "kind": "demo", ... "items": [{ "title": "First title", "comment": "First comment.", "characteristics": { "length": "short", "accuracy": "high", "followers": ["Jo", "Will"], }, "status": "active", ... }, { "title": "Second title", "comment": "Second comment.", "characteristics": { "length": "long", "accuracy": "medium" "followers": [ ], }, "status": "pending", ... }, ...] } Request for a partial response: The following request for this same resource uses the fields parameter to significantly reduce the amount of data returned.
https://www.googleapis.com/demo/v1?fields=kind,items(title,characteristics/length) Partial

response: In response to the request above, the server sends back a response that contains only the kind information along with a pared-down items array that includes only HTML title and length characteristic information in each item. 200 OK { "kind": "demo", "items": [{ "title": "First title", "characteristics": { "length": "short" } }, { "title": "Second title", "characteristics": { "length": "long" } }, ...] } Note that the response is a JSON object that includes only the selected fields and their enclosing parent objects. Details on how to format the fields parameter is covered next, followed by more details about what exactly gets returned in the response. Fields parameter syntax summary The format of the fields request parameter value is loosely based on XPath syntax. The supported syntax is summarized below, and additional examples are provided in the following section. Use a comma-separated list to select multiple fields. Use a/b to select a field b that is nested within field a; use a/b/c to select a field c nested within b. Exception: For API responses that use "data" wrappers, where the response is nested within a data object that looks like data: { ... }, do not include "data" in the fields specification. Including the data object with a fields specification like data/a/b causes an error. Instead, just use a fields specification like a/b. Use a sub-selector to request a set of specific sub-fields of arrays or objects by placing expressions in parentheses "( )". For example: fields=items(id,author/email) returns only the item ID and author's email for each element in the items array. You can also specify a single sub-field, where fields=items(id) is equivalent to fields=items/id. Use wildcards in field selections, if needed. For example: fields=items/pagemap/* selects all objects in a pagemap. More examples of using the fields parameter The examples below include descriptions of how the fields parameter value affects the response. Note: As with all query parameter values, the fields parameter value must be URL encoded. For better readability, the examples in this document omit the encoding. Identify the fields you want returned, or make field selections. The fields request parameter value is a comma-separated list of fields, and each field is specified relative to the root of the response. Thus, if you are performing a list operation, the response is a collection, and it generally includes an array of resources. If you are performing an operation that returns a single resource, fields are specified relative to that resource. If the field you select is (or is part of) an array, the server returns the selected portion of all elements in the array. Here are some collection-level examples: Examples Effect items Returns all elements in the items array, including all fields in each element, but no other fields. etag,items Returns both the etag field and all elements in the items array. items/title Returns only the title field for all elements in the items array. Whenever a nested field is returned, the response includes the enclosing parent objects. The parent fields do not include any other child fields unless they are also selected explicitly. context/facets/label Returns only the label field for all members of the facets array, which is itself nested under the context object. items/pagemap/*/title For each element in the items array, returns only the title field (if present) of all objects that are children of pagemap. Here are some resource-level examples: Examples Effect title Returns the title field of the requested resource. author/uri Returns the uri sub-field of the author object in the requested resource. links/*/href Returns the href field of all objects that are children of links. Request only parts of specific fields using sub-selections. By default, if your request specifies particular fields, the server returns the objects or array elements in their entirety. You can specify a response that includes only certain sub-fields. You do this using "( )" sub-selection syntax, as in the example below. Example Effect items(title,author/uri) Returns only the

values of the title and author's uri for each element in the items array. Handling partial responses After a server processes a valid request that includes the fields query parameter, it sends back an HTTP 200 OK status code, along with the requested data. If the fields query parameter has an error or is otherwise invalid, the server returns an HTTP 400 Bad Request status code, along with an error message telling the user what was wrong with their fields selection (for example, "Invalid field selection a/b"). Here is the partial response example shown in the introductory section above. The request uses the fields parameter to specify which fields to return. https://www.googleapis.com/demo/v1?fields=kind,items(title,characteristics/length) The partial response looks like this: 200 OK { "kind": "demo", "items": [{ "title": "First title", "characteristics": { "length": "short" } }, { "title": "Second title", "characteristics": { "length": "long" } }, ...] } Note: For APIs that support query parameters for data pagination (maxResults and nextPageToken, for example), use those parameters to reduce the results of each query to a manageable size. Otherwise, the performance gains possible with partial response might not be realized. Patch (partial update) You can also avoid sending unnecessary data when modifying resources. To send updated data only for the specific fields that you're changing, use the HTTP PATCH verb. The patch semantics described in this document are different (and simpler) than they were for the older, GData implementation of partial update. The short example below shows how using patch minimizes the data you need to send to make a small update. Example This example shows a simple patch request to update only the title of a generic (fictional) "Demo" API resource. The resource also has a comment, a set of characteristics, status, and many other fields, but this request only sends the title field, since that's the only field being modified: PATCH https://www.googleapis.com/demo/v1/324 Authorization: Bearer your_auth_token Content-Type: application/json { "title": "New title" } Response: 200 OK { "title": "New title", "comment": "First comment.", "characteristics": { "length": "short", "accuracy": "high", "followers": ["Jo", "Will"], }, "status": "active", ... } The server returns a 200 OK status code, along with the full representation of the updated resource. Since only the title field was included in the patch request, that's the only value that is different from before. Note: If you use the partial response fields parameter in combination with patch, you can increase the efficiency of your update requests even further. A patch request only reduces the size of the request. A partial response reduces the size of the response. So to reduce the amount of data sent in both directions, use a patch request with the fields parameter. Semantics of a patch request The body of the patch request includes only the resource fields you want to modify. When you specify a field, you must include any enclosing parent objects, just as the enclosing parents are returned with a partial response. The modified data you send is merged into the data for the parent object, if there is one. Add: To add a field that doesn't already exist, specify the new field and its value. Modify: To change the value of an existing field, specify the field and set it to the new value. Delete: To delete a field, specify the field and set it to null. For example, "comment": null. You can also delete an entire object (if it is mutable) by setting it to null. If you are using the Java API Client Library, use Data.NULL_STRING instead; for details, see JSON null. Note about arrays: Patch requests that contain arrays replace the existing array with the one you provide. You cannot modify, add, or delete items in an array in a piecemeal fashion. Using patch in a read-modify-write cycle It can be a useful practice to start by retrieving a partial response with the data you want to modify. This is especially important for

resources that use ETags, since you must provide the current ETag value in the If-Match HTTP header in order to update the resource successfully. After you get the data, you can then modify the values you want to change and send the modified partial representation back with a patch request. Here is an example that assumes the Demo resource uses ETags: GET https://www.googleapis.com/demo/v1/324?fields=etag,title,comment,characteristics Authorization: Bearer your_auth_token This is the partial response: 200 OK { "etag": "ETagString" "title": "New title" "comment": "First comment.", "characteristics": { "length": "short", "level": "5", "followers": ["Jo", "Will"], } } The following patch request is based on that response. As shown below, it also uses the fields parameter to limit the data returned in the patch response: PATCH https://www.googleapis.com/demo/v1/324?fields=etag,title,comment,characteristics Authorization: Bearer your_auth_token Content-Type: application/json If-Match: "ETagString" { "etag": "ETagString" "title": "", /* Clear the value of the title by setting it to the empty string. */ "comment": null, /* Delete the comment by replacing its value with null. */ "characteristics": { "length": "short", "level": "10", /* Modify the level value. */ "followers": ["Jo", "Liz"], /* Replace the followers array to delete Will and add Liz. */ "accuracy": "high" /* Add a new characteristic. */ }, } The server responds with a 200 OK HTTP status code, and the partial representation of the updated resource: 200 OK { "etag": "newETagString" "title": "", /* Title is cleared; deleted comment field is missing. */ "characteristics": { "length": "short", "level": "10", /* Value is updated.*/ "followers": ["Jo" "Liz"], /* New follower Liz is present; deleted Will is missing. */ "accuracy": "high" /* New characteristic is present. */ } } Constructing a patch request directly For some patch requests, you need to base them on the data you previously retrieved. For example, if you want to add an item to an array and don't want to lose any of the existing array elements, you must get the existing data first. Similarly, if an API uses ETags, you need to send the previous ETag value with your request in order to update the resource successfully. Note: You can use an "If-Match: *" HTTP header to force a patch to go through when ETags are in use. If you do this, you don't need to do the read before the write. For other situations, however, you can construct the patch request directly, without first retrieving the existing data. For example, you can easily set up a patch request that updates a field to a new value or adds a new field. Here is an example: PATCH https://www.googleapis.com/demo/v1/324?fields=comment,characteristics Authorization: Bearer your_auth_token Content-Type: application/json { "comment": "A new comment", "characteristics": { "volume": "loud", "accuracy": null } } With this request, if the comment field has an existing value, the new value overwrites it; otherwise it is set to the new value. Similarly, if there was a volume characteristic, its value is overwritten; if not, it is created. The accuracy field, if set, is removed. Handling the response to a patch After processing a valid patch request, the API returns a 200 OK HTTP response code along with the complete representation of the modified resource. If ETags are used by the API, the server updates ETag values when it successfully processes a patch request, just as it does with PUT. The patch request returns the entire resource representation unless you use the fields parameter to reduce the amount of data it returns. If a patch request results in a new resource state that is syntactically or semantically invalid, the server returns a 400 Bad Request or 422 Unprocessable Entity HTTP status code, and the resource state remains unchanged. For example, if you attempt to delete the value for a required field, the server returns an error. Alternate notation when PATCH HTTP verb is not supported If your firewall does not

allow HTTP PATCH requests, then do an HTTP POST request and set the override header to PATCH, as shown below: POST https://www.googleapis.com/... X-HTTP-Method-Override: PATCH ... Difference between patch and update In practice, when you send data for an update request that uses the HTTP PUT verb, you only need to send those fields which are either required or optional; if you send values for fields that are set by the server, they are ignored. Although this might seem like another way to do a partial update, this approach has some limitations. With updates that use the HTTP PUT verb, the request fails if you don't supply required parameters, and it clears previously set data if you don't supply optional parameters. It's much safer to use patch for this reason. You only supply data for the fields you want to change; fields that you omit are not cleared. The only exception to this rule occurs with repeating elements or arrays: If you omit all of them, they stay just as they are; if you provide any of them, the whole set is replaced with the set that you provide. Send feedback Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 4.0 License, and code samples are licensed under the Apache 2.0 License. For details, see the Google Developers Site Policies. Java is a registered trademark of Oracle and/or its affiliates. Last updated 2025-05-05 UTC. Need to tell us more? [[["Easy to understand","easyToUnderstand","thumb-up"],["Solved my problem","solvedMyProblem","thumb-up"],["Other","otherUp","thumb-up"]],[["Hard to understand","hardToUnderstand","thumb-down"],["Incorrect information or sample code","incorrectInformationOrSampleCode","thumb-down"],["Missing the information/samples I need","missingTheInformationSamplesINeed","thumb-down"],["Other","otherDown","thumb-down"]],["Last updated 2025-05-05 UTC."],[[["Enable gzip compression by setting the `Accept-Encoding` header to `gzip` and including `gzip` in your user agent to significantly reduce bandwidth usage."],["Use partial responses by including the `fields` request parameter to specify which fields are needed, reducing the amount of data transferred, parsed, and stored."],["Employ patch requests with the `PATCH` HTTP verb to update only the necessary fields of a resource, minimizing the data sent in update requests."],["Utilize sub-selections with parentheses `( )` within the `fields` parameter to request only specific sub-fields of arrays or objects, allowing for even more granular control over the response data."],["Combine patch requests with the partial response `fields` parameter to reduce data transfer in both directions, improving the overall efficiency of update operations."]]],[]] Why Google Choosing Google Cloud Trust and security Modern Infrastructure Cloud Multicloud Global infrastructure Customers and case studies Analyst reports Whitepapers Products and pricing See all products See all solutions Google Cloud for Startups Google Cloud Marketplace Google Cloud pricing Contact sales Support Google Cloud Community Support Release Notes System status Resources GitHub Getting Started with Google Cloud Google Cloud documentation Code samples Cloud Architecture Center Training and Certification Developer Center Engage Blog Events X (Twitter) Google Cloud on YouTube Google Cloud Tech on YouTube Become a Partner Google Cloud Affiliate Program Press Corner About Google Privacy Site terms Google Cloud terms Manage cookies Our third decade of climate action: join us Sign up for the Google Cloud newsletter Subscribe English Deutsch Español – América Latina Français Indonesia Italiano Português – Brasil –