Google Code

Developer's Guide

Picasa Web Albums allows client applications to view and update albums, photos, and comments in the form of Google Data API feeds. Your client application can use the Picasa Web Albums Data API to create new albums, upload photos, add comments, edit or delete existing albums, photos, and comments, and query for items that match particular criteria.

Contents

Audience

This document is intended for programmers who want to write client applications that can interact with Picasa Web Albums.

This document assumes that you understand the general ideas behind the Google Data APIs protocol. It also assumes that you know how to program in Java.

The code examples in this document use the Java client library; see the client library documentation for more information and more (non-Picasa Web Albums-specific) examples. If you're writing a client in another language, see the client library documentation for that language, if available; you can translate the examples in this document into other languages as needed.

Interacting with Picasa Web Albums: examples

Assume you're creating a client application that needs to communicate with Picasa Web Albums. You want your client to be able to get the feed from an album and add a photo to that album. These examples show you how to do that.

To communicate with Picasa Web Albums, your application needs to be able to do the following, among other things:

1. Find a feed URL.
2. Request the feed. If you want to request a feed of albums, for example, you can take either of two approaches:
   - Request a list of albums manually, using the raw Google Data protocol.
   - Request a list of albums using the client library.
3. Request other feeds as needed. For example, Request a list of photos.
4. Create a new album. There are four options here, depending on the kind of client you're writing:
   - If you're writing an installed (desktop) client:

        - Add an album manually
        - Add an album using the client library
    - If you're writing a web application client:
        - Add an album manually
        - Add an album using the client library
5. Post a new photo.
6. Send a query.
7. Update a photo.
8. Delete a photo.
9. Insert and delete tags and comments.

This guide focuses on operations involving albums and photos, but you can interact with Picasa Web Albums in a variety of other ways a
example, you can create, query, and delete albums, tags, and comments, and you can update albums and query users, all using the Go
protocol or the client libraries.

For general information about Google Data APIs and more example code that you can adapt for use with Picasa Web Albums, see Usin
Client Library.

Note: If you don't already have an account on Picasa Web Albums, you may want to create one for testing purposes.

Choose a feed type

You can request any of several different kinds of feeds from Picasa Web Albums; for more information, see Picasa Web Albums feed typ
reference document.

For each feed, you can specify what kinds of entries should be returned, using the `kind` parameter in the URL.

Choose one of the following feed types:

Desired items	**Feed type**	**URL**
Albums or tags associated with a given user	User-based	http://picasaweb.google.com/data/feed/api/user/*userID*?kind=*kinds*
Photos or tags from a given album	Album-based (by name or ID)	By album name: http://picasaweb.google.com/data/feed/api/user/*userID*/album/*albumName*?kind=*kinds*
		By album ID: http://picasaweb.google.com/data/feed/api/user/*userID*/albumid/*albumID*?kind=*kinds*
Comments or tags associated with a given photo	Photo-based	http://picasaweb.google.com/data/feed/api/user/*userID*/albumid/*albumID*/photoid/*photoID*?kind=*kinds*

To specify which kind or kinds of items you want, you list one or more kinds in the `kind` query parameter. The kind listed in boldface in t
column is the default kind for that feed type; if you want the default kind, you can leave out the `kind` query parameter.

Request a list of albums manually

The general idea of how to request a Picasa Web Albums feed is that you determine the feed URL, and then you send an HTTP `GET` req
URL. Picasa Web Albums then returns a Google Data feed containing entries of whatever sort you specified in the `kind` URL parameter
include a `kind` parameter, the default kind depends on the feed you requested; for more information, see Kind values in the reference d

Google provides client libraries for interacting with Google Data API services in a variety of programming languages. You can use the cli
send the HTTP request and handle the returned feed; we'll show you an example of how to do that below. But whether or not you use th
the following is what's going on at the protocol level. If you're using a UNIX system and you want to try this out without writing any code,
UNIX command-line utilities `curl` or `wget` useful; for more information, see the manual pages for those utilities.

To get a feed listing all of the albums belonging to user "liz", you send an HTTP request like the following to Picasa Web Albums:

```
GET http://picasaweb.google.com/data/feed/api/user/liz?kind=album
```

After you send the `GET` request, Picasa Web Albums may return a redirect, depending on various factors. If so, send another `GET` reque

Picasa Web Albums then returns an HTTP `200 OK` status code and a feed containing all the albums for the user. If there's only one ent Albums returns something similar to the following feed.

We've slightly edited the following example to make it a little more readable by humans; in particular, a real Picasa Web Albums feed co URLs.

```
<feed xmlns='http://www.w3.org/2005/Atom'
    xmlns:media='http://search.yahoo.com/mrss/'
    xmlns:gphoto='http://schemas.google.com/photos/2007'>
  <id>http://picasaweb.google.com/data/feed/api/user/liz</id>
  <updated>2007-03-06T00:08:51.553Z</updated>
  <category scheme='http://schemas.google.com/g/2005#kind'
    term='http://schemas.google.com/photos/2007#user'></category>
  <title type='text'>liz</title>
  <subtitle type='text'></subtitle>
  <icon>http://picasaweb.google.com/image/liz/imageKey/liz.jpg?imgmax=64&amp;crop=1</icon>
  <link rel='http://schemas.google.com/g/2005#feed'
    type='application/atom+xml'
    href='http://picasaweb.google.com/data/feed/api/user/liz'>
  </link>
  <link rel='alternate' type='text/html'
    href='http://picasaweb.google.com/liz'></link>
  <link rel='self' type='application/atom+xml'

href='http://picasaweb.google.com/data/feed/api/user/liz?start-index=1&amp;max-results=10000&amp;k
  </link>
  <author>
    <name>Liz</name>
    <uri>http://picasaweb.google.com/liz</uri>
  </author>
  <generator version='1.00' uri='http://picasaweb.google.com/'>Picasaweb</generator>
  <gphoto:user>liz</gphoto:user>
  <gphoto:nickname>Liz</gphoto:nickname>

<gphoto:thumbnail>http://picasaweb.google.com/image/liz/imageKey/liz.jpg?imgmax=64&amp;crop=1</gph
  <entry>
    <id>http://picasaweb.google.com/data/entry/api/user/liz/albumid/albumID</id>
    <published>2007-03-01T08:00:00.000Z</published>
    <updated>2007-03-02T02:09:52.000Z</updated>
    <category scheme='http://schemas.google.com/g/2005#kind'
      term='http://schemas.google.com/photos/2007#album'></category>
    <title type='text'>Netherfield Park</title>
    <summary type='text'>Photos of my vacation.</summary>
    <rights type='text'>public</rights>
    <link rel='self' type='application/atom+xml'
      href='http://picasaweb.google.com/data/entry/api/user/liz/albumid/albumID'>
    </link>
    <link rel='http://schemas.google.com/g/2005#feed'
      type='application/atom+xml'
      href='http://picasaweb.google.com/data/feed/api/user/liz/albumid/albumID'>
    </link>
    <link rel='alternate' type='text/html'
      href='http://picasaweb.google.com/liz/NetherfieldPark'></link>
    <author>
      <name>Liz</name>
      <uri>http://picasaweb.google.com/liz</uri>
    </author>
    <gphoto:id>albumID</gphoto:id>
    <gphoto:name>NetherfieldPark</gphoto:name>
    <gphoto:location></gphoto:location>
    <gphoto:access>public</gphoto:access>
    <gphoto:timestamp>1172736000000</gphoto:timestamp>
    <gphoto:numphotos>184</gphoto:numphotos>
    <gphoto:numphotosremaining>316</gphoto:numphotosremaining>
    <gphoto:bytesUsed>11060624</gphoto:bytesUsed>
    <gphoto:user>liz</gphoto:user>
    <gphoto:nickname>Liz</gphoto:nickname>
    <gphoto:commentingEnabled>true</gphoto:commentingEnabled>
    <gphoto:commentCount>0</gphoto:commentCount>
    <media:group>
      <media:title type='plain'>Netherfield Park</media:title>
      <media:description type='plain'>Photographs taken on vacation.</media:description>
      <media:keywords></media:keywords>
      <media:content
        url='http://picasaweb.google.com/image/liz/imageKey/Netherfield.jpg'
        type='image/jpeg' medium='image'>
      </media:content>
      <media:thumbnail

url='http://picasaweb.google.com/image/liz/imageKey/Netherfield.jpg?imgmax=160&amp;crop=1'
        height='160' width='160'>
      </media:thumbnail>
      <media:credit>Liz</media:credit>
    </media:group>
  </entry>
</feed>
```

For information about what each of those elements means, see the Google Data APIs Protocol Reference document, or the Atom 1.0 specification. For Picasa Web Albums specific information, see the Elements section of the Picasa Web Albums Data API reference document.

If your request fails for some reason, Picasa Web Albums may return a different status code; for information about the status codes, see the protocol reference document.

Note: The entries in a feed are ordered by the values of their `updated` elements, with the most recently updated entry appearing first in the feed.

Request a list of albums using the client library

The above description shows how to send a request and receive a response by hand; if you already have code that handles feeds, then you can simply use it to send the relevant `GET` request, as shown above.

However, if you don't already have your own feed-handling code, then we encourage you to use Google's client library where possible.

To use the client library, the general steps are:

1. Obtain or construct the appropriate URL.
2. Create an object of the `PicasawebService` class, which handles communication and authentication with Picasa Web Albums.
3. Use client library methods to send the request and receive any results the service sends back.

For example, to request the feed of albums belonging to user "liz", you could use the following Java code.

```
URL albumsUrl = new
  URL("http://picasaweb.google.com/data/feed/api/user/liz?kind=album");
PicasawebService myService = new PicasawebService("exampleCo-exampleApp-1");

// Send the request for the user's albums.
UserFeed myUserFeed = myService.getFeed(albumsUrl, UserFeed.class);

// Print the title of the returned feed:
System.out.println(myUserFeed.getTitle().getPlainText());
```

In the above example, first you specify the URL, then you create a `PicasawebService` object.

Then you call the `getFeed` method. That method handles all of the communication with Picasa Web Albums—it sends the HTTP `GET` request, it receives the redirect (if any), it sends the second GET request, it waits for the XML response to arrive, and it parses the XML and assigns the element contents and attribute values to appropriate parts of the given feed object. (In this case, the UserFeed class is a subclass of the generic Feed class.)

If an error occurs during the getFeed call, then the client library generates an appropriate exception, so your code should be prepared to handle exceptions.

The final line of the above code calls the feed's getTitle method, which returns a TextConstruct object. To transform the text construct into a String, we call the title's getPlainText method.

In most contexts, you'll probably want to do something more with the feed than just print out the title. For example, the client library lets you generate Atom or RSS representations of the feed and perform various other transformations. For more information, see the Javadoc documentation.

Request a list of photos

You can request any of the other available feeds in much the same way.

For example, to get a feed listing all of the photos in an album named "NetherfieldPark", belonging to user "liz", you send an HTTP request like the following to Picasa Web Albums:

```
GET http://picasaweb.google.com/data/feed/api/user/liz/album/NetherfieldPark?kind=photo
```

In this example, we're using the album-by-name URI instead of the album-by-ID URI. In some cases your client may already have the album name; in others, to request an album by name you may have to construct the album-by-name URI using the value of the <gphoto:name> element in the list-of-albums feed from the previous example. If you prefer to request a feed by album ID, you can use the URI given in the <link rel='http://schemas.google.com/g/2005#feed'> element in the list-of-albums feed.

Note: The photos in a feed are ordered by the values of their updated elements, with the most recently updated photo appearing first in the feed.You can't request a photo feed sorted in a different order.

To request the list-of-photos feed using the Java client library, you could use the following Java code.

```
URL albumAndPhotosUrl = new
  URL("http://picasaweb.google.com/data/feed/api/user/liz/album/NetherfieldPark?kind=photo");
PicasawebService myService = new PicasawebService("exampleCo-exampleApp-1");

// Send the request for the album's photos.
AlbumFeed myAlbumFeed = myService.getFeed(albumAndPhotosUrl, AlbumFeed.class);

// Print the title of the returned feed:
System.out.println(myAlbumFeed.getTitle().getPlainText());
```

In addition to requesting a list of photos for a particular album, you can also request a list of photos recently uploaded by a user. This type of feed is retrieved using a different URL, but the format of the feed is the same. The following URL will retrieve a feed containing the most recent 100 photos uploaded by liz. Photos contained in 'unlisted' albums will not appear in this feed unless you are authenticated as the appropriate user.

```
GET http://picasaweb.google.com/data/feed/api/user/liz?kind=photo&max-results=100
```

Add an album manually (desktop client version)

Most Google Data operations require authentication. Google Data uses two different authentication systems, depending on what kind of client you're writing. If your client is a standalone single-user "installed" client (such as a desktop application), then you should use the "ClientLogin" system; if your client is a multi-user web application client, then you should use the "AuthSub" system. (For more information about authentication, see the Google Account Authentication documentation.)

Because the two systems are so different, this document covers them in separate sections. Therefore, if your client is an installed application, read this section. If your client is a web app, skip this section and read Add an album manually (web client version) instead.

Adding an album is a little bit more complicated than getting a feed, because you need to create the XML code or client-library object representing the new album, and because authentication is required.

Here's what you do at the protocol level:

First, create an album. For example, you might create the following XML code:

```
<entry xmlns='http://www.w3.org/2005/Atom'
    xmlns:media='http://search.yahoo.com/mrss/'
    xmlns:gphoto='http://schemas.google.com/photos/2007'>
  <title type='text'>Trip To Italy</title>
  <summary type='text'>This was the recent trip I took to Italy.</summary>
  <gphoto:location>Italy</gphoto:location>
  <gphoto:access>public</gphoto:access>
  <gphoto:commentingEnabled>true</gphoto:commentingEnabled>
  <gphoto:timestamp>1152255600000</gphoto:timestamp>
  <media:group>
    <media:keywords>italy, vacation</media:keywords>
  </media:group>
  <category scheme='http://schemas.google.com/g/2005#kind'
    term='http://schemas.google.com/photos/2007#album'></category>
</entry>
```

Then authenticate the user. To do that, send a `POST` request to the following URL:

```
https://www.google.com/accounts/ClientLogin
```

Include the relevant parameters in the body of the `POST` request, as described in the ClientLogin documentation. Use `lh2` as the service name.

If the request succeeds, then the response contains an alphanumeric string labeled `Auth`.

After a successful authentication request, use the `Auth` value to create an Authorization header for a new `POST` request:

```
Authorization: GoogleLogin auth=yourAuthValue
```

In the body of the new `POST` request, place the Atom `<entry>` element you created above, using the `application/atom+xml` content type.

Now send the `POST` request to the appropriate Picasa Web Albums URL:

```
http://picasaweb.google.com/data/feed/api/user/userID
```

Picasa Web Albums creates a new album using the data you sent, then returns an HTTP `201 CREATED` status code, along with a copy of the new album in the form of an `<entry>` element. The returned entry is similar to the one you sent, but the returned one contains various elements added by Picasa Web Albums, such as an `<id>` element.

If your request fails for some reason, then Picasa Web Albums may return a different status code; for information about the status codes, see the HTTP status codes section of the protocol reference document.

Add an album using the client library (desktop client version)

As an alternative to working with the raw HTTP and XML, you can create a new album using the client library. To do that, you follow the same general steps as in the get-a-feed example, with a couple of extra steps mixed in:

1. Obtain or construct the appropriate URL.
2. Construct an `AlbumEntry` object, using client library methods to create objects corresponding to Atom elements.
3. Create a `PicasawebService` object.
4. Add the user's credentials to the `PicasawebService`.
5. Use client library methods to send the request and receive any results the service sends back.

For example, to add an album to a gallery, use the following Java code. You'll have to replace the credentials in this example with the user's email address and password.

```
URL postUrl = new URL("http://picasaweb.google.com/data/feed/api/user/ userID");
AlbumEntry myEntry = new AlbumEntry();

myEntry.setTitle(new PlainTextConstruct("Trip to France"));
myEntry.setDescription(new
  PlainTextConstruct("My recent trip to France was delightful!"));

Person author = new Person("Elizabeth Bennet", null, "liz@gmail.com");
myEntry.getAuthors().add(author);

PicasawebService myService =
  new PicasawebService("exampleCo-exampleApp-1");
myService.setUserCredentials("liz@gmail.com", "mypassword");

// Send the request and receive the response:
AlbumEntry insertedEntry = myService.insert(postUrl, myEntry);
```

The above code creates a new Atom entry, then sets various element values. Then it creates a `PicasawebService`, uses the `setUserCredentials` method to set the user ID and password, and calls the `insert` method to add the album and receive the response. If an error occurs during the insertion, then the client library generates an appropriate exception, so your code should be prepared to handle exceptions.

Add an album manually (web client version)

This section is for use only with a multi-user web application client. If your client is an installed application, then skip this section and read Add an album manually (desktop client version) instead.

Adding an album is a little bit more complicated than getting a feed, because you need to create the XML code or client-library object representing the album, and authentication is required.

Here's what you do at the protocol level:

First, create an album. For example, you might create the following XML code:

```
<entry xmlns='http://www.w3.org/2005/Atom'
    xmlns:media='http://search.yahoo.com/mrss/'
    xmlns:gphoto='http://schemas.google.com/photos/2007'>
  <title type='text'>Trip To Italy</title>
  <summary type='text'>This was the recent trip I took to Italy.</summary>
  <gphoto:location>Italy</gphoto:location>
  <gphoto:access>public</gphoto:access>
  <gphoto:commentingEnabled>true</gphoto:commentingEnabled>
  <gphoto:timestamp>1152255600000</gphoto:timestamp>
  <media:group>
    <media:keywords>italy, vacation</media:keywords>
  </media:group>
  <category scheme='http://schemas.google.com/g/2005#kind'
    term='http://schemas.google.com/photos/2007#album'></category>
</entry>
```

Then authenticate the user, using the AuthSub authentication system.

To acquire an AuthSub token for a given Picasa Web Albums user, your application must redirect the user to the AuthSubRequest URL, which prompts them to log into their Google account.

After the user logs in, the AuthSub system redirects them to the URL you specified in the `next` query parameter of the AuthSubRequest URL. The AuthSub system appends an authentication token to that URL, as the value of the `token` query parameter. Your application then uses that authentication token in subsequent interactions with Picasa Web Albums.

For details, including information on registering your application with Google and on exchanging a one-time token for a session token, see the AuthSub documentation.

Use the authentication token to create an Authorization header for a new `POST` request:

```
Authorization: AuthSub token="yourAuthToken"
```

Note that the values in the Authorization header for AuthSub should be surrounded by quotation marks.

In the body of the new `POST` request, place the Atom `<entry>` element you created above, using the `application/atom+xml` content type.

Now send the `POST` request to the appropriate Picasa Web Albums URL:

```
http://picasaweb.google.com/data/feed/api/user/userID
```

Picasa Web Albums creates a new album using the entry you sent, then returns an HTTP `201 CREATED` status code, along with a copy of the new album in the form of an `<entry>` element. The returned entry is similar to the one you sent, but the returned one contains various elements added by Picasa Web Albums, such as an `<id>` element.

If your request fails for some reason, then Picasa Web Albums may return a different status code; for information about the status codes, see the HTTP status codes section of the protocol reference document.

Add an album using the client library (web client version)

As an alternative to working with the raw HTTP and XML, you can create a new album using the client library. To do that, you follow the same general steps as in Request the feed using the client library, with a couple of extra steps mixed in:

1. Construct the AuthSub URL.
2. Send the user to a Google page to enter their AuthSub credentials.
3. When the user returns, acquire a session token.
4. Create a `PicasawebService` object.
5. Tell the client library to use the given session token for interaction with the service, using the `setAuthSubToken` method.
6. Construct an `AlbumEntry` object, using client library methods to set properties corresponding to Atom elements.
7. Use client library methods to send the request and receive any results the service sends back.

To add an album using AuthSub authentication, first follow the instructions in the AuthSub and GData client library documentation to acquire a session token, using the `getRequestUrl`, `getTokenFromReply`, `exchangeForSessionToken`, and `setAuthSubToken` methods. The scope string to use is `http://picasaweb.google.com/data/`.

After you've called the `PicasawebService` object's `setAuthSubToken` method, the client library sends the token with every request, so you don't need to do anything further with the token.

The following Java code shows how to add the album after you've acquired a session token:

```
PicasawebService myService =
  new PicasawebService("exampleCo-exampleApp-1");

myService.setAuthSubToken(sessionToken, null);

URL postUrl =
  new URL("http://picasaweb.google.com/data/feed/api/user/userID");
AlbumEntry myEntry = new AlbumEntry();

myEntry.setTitle(new PlainTextConstruct("Trip to Mongolia"));
myEntry.setDescription(new
  PlainTextConstruct("My trip to Mongolia was most enjoyable, but cold."));

Person author = new Person("Elizabeth Bennet", null, "liz@gmail.com");
myEntry.getAuthors().add(author);

// Send the request and receive the response:
AlbumEntry insertedEntry = myService.insert(postUrl, myEntry);
```

The above code sets the AuthSub session token, then creates a new Atom entry, then sets various element values. Then it calls the `insert` method to insert the post and receive the response. If an error occurs during the insertion, then the client library generates an appropriate exception, so your code should be prepared to handle exceptions.

Post a new photo

There are two ways to add a photo to an album using the data API:

- Upload the binary image data along with its metadata. To do this, use MIME content type"multipart/related"; send photo metadata in one part of the `POST` body, and binary-encoded image data in another part. This is the preferred approach.
- Upload the binary image data without the metadata.

While all photos appearing on the Picasa Web Albums site are in the JPEG format, photos of any of the following types can be uploaded using the API:

- image/bmp
- image/gif
- image/jpeg
- image/png

Post a photo with metadata

To send metadata along with the photo, post to the following URL:

```
POST http://picasaweb.google.com/data/feed/api/user/liz/album/NetherfieldPark
```

And use the following format for the body of the `POST`:

```
Content-Type: multipart/related; boundary="END_OF_PART"
Content-Length: 24680246
MIME-version: 1.0

Media multipart posting
--END_OF_PART
Content-Type: application/atom+xml

<entry xmlns='http://www.w3.org/2005/Atom'>
  <title>darcy-beach.jpg</title>
  <summary>Darcy on the beach</summary>
  <category scheme="http://schemas.google.com/g/2005#kind"
    term="http://schemas.google.com/photos/2007#photo"/>
</entry>
--END_OF_PART

Content-Type: image/jpeg

...binary image data goes here...
--END_OF_PART--
```

Note that the `<title>` element contains the filename you want to use for the image.

To upload a photo and metadata using the Java client library, use code similar to the following:

```
PhotoEntry myPhoto = new PhotoEntry();
myPhoto.setTitle(new PlainTextConstruct("darcy-beach.jpg"));
myPhoto.setDescription(new PlainTextConstruct("Darcy on the beach"));
myPhoto.setClient("myClientName");
myPhoto.setTimestamp (new Date());

MediaFileSource myMedia = new MediaFileSource(new File("/usr/home/liz/photos/photo23a.jpg"),
"image/jpeg");
myPhoto.setMediaSource(myMedia);

PhotoEntry returnedPhoto = myService.insert(feedUrl, myPhoto);
```

The MediaFileSource object specifies a source file on the local disk for the image; you can instead use MediaSource to use an image from another kind of data stream. The insert method pulls the image data from the specified file or stream and sends it to Picasa Web Albums.

The setClient method sets the name of the client that's uploading the photo. You can use any name for your client that you want.

Post a photo without metadata

To send a photo without its associated metadata, post to the following URL:

```
POST http://picasaweb.google.com/data/feed/api/user/liz/album/NetherfieldPark
```

And use the following format for the body of the POST:

```
Content-Type: image/jpeg
Content-Length: 47899
Slug: darcy-beach.jpg

...binary image data goes here...
```

The optional Slug: HTTP header specifies a filename for Picasa Web Albums to use for the photo.

To upload a photo without metadata using the Java client library, use code similar to the following:

```
MediaFileSource myMedia = new MediaFileSource(new File("/usr/home/liz/photos/photo23a.jpg"),
"image/jpeg");
PhotoEntry returnedPhoto = myService.insert(feedUrl, PhotoEntry.class, myMedia);
```

The Java client library internally uses the value returned by the MediaSource.getName method as the value of the Slug: header.

Send a query

Google Data lets you specify various query parameters, such as a limit on the number of entries returned. For more information about the query parameters for Google Data queries, see the Google Data APIs Protocol Reference document. Picasa Web Albums supports the standard Google Data start-index and max-results parameters. It does not currently support the other standard parameters.

Queries can be sent either authenticated or unauthenticated. Unauthenticated queries return only public information; authenticated queries return any data the authenticated user has access to.

To request a feed with a limit on number of entries returned, send an HTTP request like the following to Picasa Web Albums, using the feed URL:

```
GET http://picasaweb.google.com/data/feed/api/user/liz?kind=album&max-results=10
```

When you send that GET request, Picasa Web Albums returns an HTTP 200 OK status code and a feed containing up to ten albums.

You can, of course, use the client library to send a query instead of sending it by hand. In this case, you'll need to add a new step to the process: constructing a Query object.

For example, to view entries created or modified in a particular date range, use the following Java code.

```
URL feedUrl = new URL("http://picasaweb.google.com/data/feed/api/user/liz?kind=album");

Query myQuery = new Query(feedUrl);
myQuery.setMaxResults(10);

// Send the request and receive the response:
AlbumFeed resultFeed = myService.query(myQuery, AlbumFeed.class);
```

The above code creates a new Query and sets the maximum number of results; then it calls the service's `query` method to send the query and receive the response.

Update a photo

You can replace the metadata and/or the binary image data for a photo.

Update a photo and its metadata

To replace both the binary data and metadata for the photo that you inserted earlier, use an HTTP request like the following:

```
PUT
http://picasaweb.google.com/data/entry/api/user/liz/albumid/albumID/photoid/photoID/versionNumber
```

The URI is the value of the `<link rel="edit">` tag that was returned after you did the earlier `POST`. Note that if you try to update using a version number other than the latest one, you may receive an error; for more information, see Optimistic concurrency (versioning) in the Google Data reference document.

In the body of the `PUT`, include the updated metadata and image data, in the same multipart format that you used to do the `POST` with metadata.

Note: As usual with Google Data, you can't do a partial update of an entry; you have to send the full entry data to replace the existing data. The exception is that you can choose to send only the metadata or only the image, as described in the following sections.

You can update any or all of the following metadata properties:

- `<title>`
- `<description>`
- `<gphoto:checksum>`
- `<gphoto:client>`
- `<gphoto:rotation>`
- `<gphoto:timestamp>`
- `<gphoto:commentingEnabled>`

To send an update using the Java client library, use code similar to the following:

```
PhotoEntry myPhoto = ...get photo entry...
myPhoto.setMediaSource(updatedMediaSource);
myPhoto.setTitle(new PlainTextConstruct("New Title"));
myPhoto.updateMedia(true);
```

Update only the photo

To replace only the photo's binary data, use the following HTTP request:

```
PUT
http://picasaweb.google.com/data/media/api/user/liz/albumid/albumID/photoid/photoID/versionNumber
```

The URI is the value of the `<link rel="edit-media">` tag that was returned after you did the earlier `POST`. Note that if you try to update using a version number other than the latest one, you may receive an error; for more information, see Optimistic concurrency (versioning) in the Google Data reference document.

In the body of the `PUT`, include the replacement image data, in the same format that you used to do the `POST` without metadata.

To do the same thing using the Java client library, use code similar to the following:

```
PhotoEntry myPhoto = ...get photo entry...
myPhoto.setMediaSource(updatedMediaSource);
myPhoto.updateMedia(false);
```

Update only the metadata

To replace only the photo's metadata (and not the image itself), you follow the steps that you would follow to send an update to a non-media feed.

In particular, send the following HTTP request:

```
PUT
http://picasaweb.google.com/data/entry/api/user/liz/albumid/albumID/photoid/photoID/versionNumber
```

The URI is the value of the `<link rel="edit">` tag that was returned after you did the earlier `POST`. Note that if you try to update using a version number other than the latest one, you may receive an error; for more information, see Optimistic concurrency (versioning) in the Google Data reference document.

In the body of the `PUT`, provide the updated metadata, in the form of an `<atom:entry>` element containing image metadata.

Note: As usual with GData, you can't do a partial update of an entry; you have to send the full entry metadata to replace the existing metadata.

The photo itself is not re-sent to the server.

To update the metadata (and not the photo) using the Java client library, use code similar to the following:

```
PhotoEntry myPhoto = ...get photo entry...
myPhoto.setTitle(new PlainTextConstruct("New Title"));
myPhoto.update();
```

Delete a photo

To delete a photo and its metadata, send an HTTP `DELETE` request to either the edit link or the edit-media link. In both cases, the photo and metadata are deleted.

For example, to delete the photo from the previous examples:

```
DELETE
http://picasaweb.google.com/data/entry/api/user/liz/albumid/albumID/photoid/photoID/versionNumber
```

As with updating, if you try to delete using a version number other than the latest one, you may receive an error; for more information, see Optimistic concurrency (versioning) in the Google Data reference document.

Or you can use the Java client library:

```
myPhoto.delete();
```

Insert and delete tags and comments

In addition to albums and photos, the Picasa Web Albums Data API lets you interact with two other kinds of data: tags and comments. You can't update tags or comments, but you can insert and delete them.

The process is similar to the process for inserting and deleting albums.

For example, to manually add a tag to a photo, first you create the XML code for the tag, including a `<category>` element indicating that it's a tag:

```
<entry xmlns='http://www.w3.org/2005/Atom'>
  <title>AtTheBeach</title>
  <category scheme="http://schemas.google.com/g/2005#kind"
    term="http://schemas.google.com/photos/2007#tag"/>
</entry>
```

Then you send the tag as the body of an authenticated `POST` request to the photo's `POST` URI:

```
POST http://picasaweb.google.com/data/feed/api/user/userID/album/albumName/photo/photoID
```

To manually add a comment, you follow the same process (using the same `POST` URI), but use a `<content>` element for the comment's text, and use a slightly different `<category>` element to indicate that it's a comment:

```
<entry xmlns='http://www.w3.org/2005/Atom'>
  <content>This is what all the gentlemen are wearing this season.</content>
  <category scheme="http://schemas.google.com/g/2005#kind"
    term="http://schemas.google.com/photos/2007#comment"/>
</entry>
```

To insert a tag or comment using the client library, first create a `TagEntry` or `CommentEntry` object, then use the `PicasawebService.insert` method (with the same `POST` URI as above) to insert the entry.

To delete a tag or comment, send an HTTP `DELETE` request to the tag's or comment's edit URI, or call the `TagEntry.delete` or `CommentEntry.delete` methods.