

Google Data APIs for Cocoa Applications

Greg Robbins
David Oster
Google Mac Client Team

It's your data.

Your calendar... photos... movies... documents... spreadsheets...
address book...

If you upload it to Google, you should be able to download it
completely from Google.

What's in it for my users?

Google data APIs let users of your Mac application

- store
- access
- share

their words, numbers, schedules, images, videos, and more.

What is a Google data API?

GData is:

- a subclass of Atom Publishing Protocol (APP is used for uploading to blogs)

APP is:

- a subclass of RSS (for browsing web publications)

RSS is:

- a subclass of XML (`<for><communicating what="anything" how="verbosely" /> </for>`)

So GData is: XML and protocol for talking with *any* Google service. (And with some non-Google services, too.)

Is GData a complex protocol?

No. GData is a REST protocol.

What's REST?

REST means:

1. You ask for something.
2. The server gives it to you.

So cookies or persistent state should **not** affect what you get back.

...kinda like: http

More than browsing

Like a web browser, GData uses http commands.

In http, GET doesn't change the data on the server.

POST (like uploading a form) *does* change the server's data.

GData also relies on two other http commands, PUT and DELETE.

All together, this is CRUD:

- Create (POST)
- Retrieve (GET)
- Update (PUT)
- Delete (DELETE)

Which Google services have APIs?

All of them, eventually.

Already...

- Google Calendar
- Picasa Web Albums
- Google Docs & Spreadsheets
- Google Base
- Blogger
- YouTube
- and the rest

Other companies use GData and Atom Publishing Protocol, too, and the Objective-C library should work with their services as well.

Ask... get...

Ask a web browser for... <http://www.google.com/calendar/feeds/default>

Get this feed back...

```
<feed>
  <id>http://www.google.com/calendar/feeds/default</id>
  <updated>2007-10-03T23:00:45.684Z</updated>
  <title type="text">Pablo Picasso's Calendar List</title>
  <entry>
    <id>http://www.google.com/calendar/feeds/default/picasso%40gmail.com</id>
    <title type="text">Pablo's Main Calendar</title>
    <gCal:timezone value="America/Los_Angeles"/>
    <gCal:color value="#2952A3"/>
    <gCal:accesslevel value="owner"/>
  </entry>
</feed>
```

XML? HTTP transactions? User authentication?

Ick.

Mac developers want Cocoa APIs.

Mac developers at Google want Cocoa APIs.

KVC... Cocoa bindings... object introspection... mmmm.

Ask... get... Cocoa-style.

```
[calendar fetchThisFeed:url  
        callWhenDone:myMethod];
```

Because we are lazy enough to want it to be this easy,
we wrote the Google data APIs Objective-C Client Library.

Yes, it's **open source**. Genuinely.

<http://gdata-objectivec-client.googlecode.com/>

But first, terminology.

Element: any item of data
a start time, a photo's name, a formula...

Entry: a record made of elements
*an appointment on your calendar, a photo,
a cell in a spreadsheet...*

Feed: a list of entries
a list of appointments, a photo album, a worksheet...

Service: an object that talks to a server
Calendar, Photos, Spreadsheets...

Ticket: one transaction with a service
scheduling an appointment, uploading a photo...

Ask... for real

Using Objective-C, get a feed listing all of my calendars .

```
service = [[GDataServiceGoogleCalendar alloc] init];

[service setUserAgent:@"MyCalendarApp-1.0"];
[service setUserCredentialsWithUsername:username
 password:password];

GDataServiceTicket *ticket;
ticket = [service fetchCalendarFeedForUsername:username
 delegate:self
 didFinishSelector:@selector(finishedWithFeed:)
 didFailSelector:@selector(failedWithError:)];
```

Get... for real

```
- (void)ticket:(GDataServiceTicket *)ticket
finishedWithFeed:(GDataFeedCalendar *)feed {
    if ([[feed entries] count] > 0) {
        GDataEntryCalendar *firstCalendar = [[feed entries] objectAtIndex:0];
        GDataTextConstruct *titleTextConstruct = [firstCalendar title];
        NSString *title = [titleTextConstruct stringValue];
        NSLog(@"first calendar's title:%@", title);
    }
}
```

Huh? A *text construct*?

There's XML underlying the elements, and it has other attributes besides the text.

```
<title type="text" language="english">Pablo Picasso's Calendar</title>
```

You won't have to touch XML, but you'll occasionally have to look at it. Sorry.

Want to upload a photo?

```
// get the URL for the album
NSURL *albumURL = [GDataServiceGooglePicasaWeb
    picasaWebFeedURLForUserID:@"my.account@gmail.com" albumID:nil
    albumName:@"MyBestPhotos" photoID:nil kind:nil access:nil];

// make a new entry for this photo
GDataEntryPhoto *newPhoto = [GDataEntryPhoto photoEntry];
[newPhoto setTitleWithString:@"Sunset Photo"];
[newPhoto setPhotoDescriptionWithString:@"A nice day"];

// attach the photo data
NSData *data = [NSData dataWithContentsOfFile:@"SunsetPhoto.jpg"];
[newPhoto setPhotoData:data];
[newPhoto setPhotoMIMEType:@"image/jpeg"];
```

Want to upload a photo? *continued*

```
// now upload it
GDataServiceTicket *ticket;
ticket = [service fetchPicasaWebEntryByInsertingEntry:newPhoto
          forFeedURL:albumURL
          delegate:self
          didFinishSelector:@selector(addPhotoTicket:finishedWithEntry:)
          didFailSelector:@selector(addPhotoTicket:failedWithError:)];
```

Why are we calling “fetch” to upload?

GData servers always send back a feed or entry, like the one you just uploaded.

Where’s the networking?

It’s all inside that “fetch” call. (NSURLConnection does the heavy lifting.)

Should I spawn a new thread? No.

A Cocoa idiom:

The service creates the thread to do the work asynchronously, so your code doesn't need to make threads.

Your code will be called back on the same thread that started the fetch.

Is this going to be painful to use?

Nope.

- Builds as a framework (toss it into your app's bundle)
- Works with KVC & bindings
- Built-in http logging (helps answer, "Why doesn't that work?")
- Built-in data cache (when the server says "Same as before")
- Plenty of sample code
- 100% open source, Apache 2.0 license

Where do I start?

Code and documentation

<http://gdata-objectivec-client.googlecode.com/>

Discussion

<http://groups.google.com/group/gdata-objectivec-client>

End

GData Objective-C Client Library

What does GData add to APP?

Atom Publishing Protocol provides:

- Simple Format — Atom Syndication Format
- Simple Protocol — HTTP

Google Data APIs add:

- Data model
- Query standards
- Concurrency
- Authentication

Why REST rather than SOAP?

- SOAP:
 - Arbitrary actions (verbs)
 - Arbitrarily data payloads
 - Complex service description and behavior (WSDL)
- REST:
 - Common actions (HTTP Get, Post, Put, Delete)
 - Operate on whole resources (XML documents)
 - Uniform resource names and links (URIs)
- Why REST over SOAP?
 - It's how the web works (stateless, cachable, scalable)
 - It's easy on the programmer

Optimistic Concurrency

```
→ GET /feeds/entry1
← 200 OK
...
<link rel="edit"
href="http://www.example.com/feeds/entry1/version1/" />

→ PUT /feeds/entry1/version1/
← 200 OK
...
<link rel="edit"
href="http://www.example.com/feeds/entry1/version2/" />

→ DELETE /feeds/entry1/version1/
← 409 CONFLICT
```

Query examples

- Full Text Search

http://www.example.com/feeds/friends?q=John

- Categories (Tags)

http://www.example.com/feeds/friends/-/work|play

- Update Time

http://www.example.com/feeds/friends?

updated-min=2005-08-09T10:57:00-08:00

- Custom ... –e.g. spreadsheets supports min-row, max-row, ...

- Output Format

http://www.example.com/feeds/friends?alt=json

GData/APP Areas for Improvement

- Partial-entry updates
- Standard authentication mechanisms
- eTags for resource versions
- Standard errors from services
- Incorporation of other XML formats

Authentication: ClientLogin

To obtain an auth token, POST this to <https://www.google.com/accounts/ClientLogin>

```
Email=johndoe@gmail.com&Passwd=north23AZ  
&service=cl&source=Gulp-CalGulp-1.05
```

The server responds:

```
HTTP/1.0 200 OK
```

```
SID=DQAAAGgA...7Zg8CTN  
LSID=DQAAAGsA...lk8BBbG  
Auth=DQAAAGgA...dk3fA5N
```

Then add the auth token to http headers for the fetches from service:

```
Authorization: GoogleLogin auth=DQAAAGgA...dk3fA5N
```