Google Calendar: Downloading Future Events

By Robert Molina, Technical Support Engineer, 4D Inc.

Technical Note 07-30

Abstract

This Technical Note provides information on how to interface with the Google Calendar API using 4D Internet Commands. Specifically, it shows how to download future events from your Google Calendars and display the data in a 4D Database. In addition, a sample database is provided.

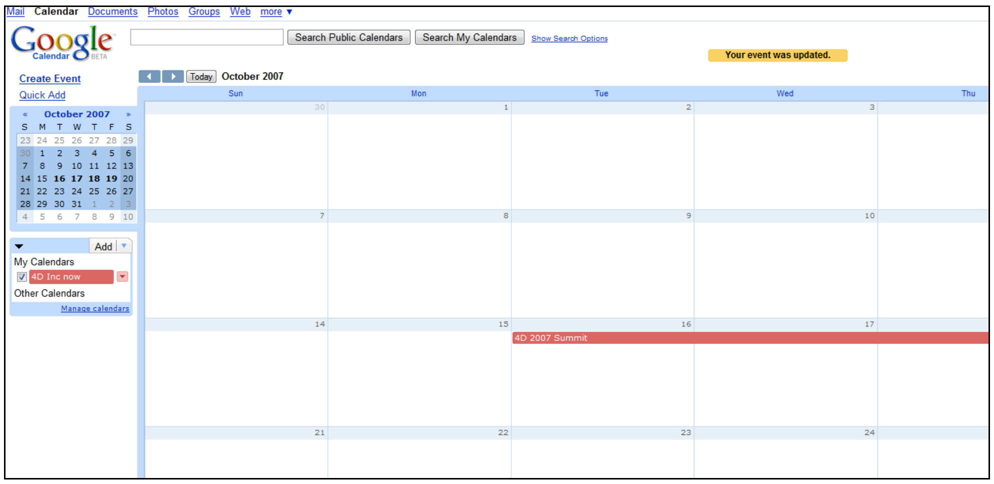
Introduction

Google is known industry wide for its popular search engine. It was founded in September 27, 1998. Their mission statement is "to organize the world's information and make it universally accessible and useful”. Since their existence in 1998, Google has evolved to provide other services such as web-based email, online mapping, video sharing and a web based calendar. In addition to providing these services to the public, Google has also provided Developers with an Application Program Interface (API) that allows interaction with their web based services.

The Technical Note focuses on a specific service, Google Calendar. It provides examples of authenticating, requesting a calendar list and requesting future events from the Google Calendar Service.

What is Google Calendar?

Google Calendar is a web-based application that assists users with time management. It is similar to the desktop calendar applications such as iCal and Microsoft Outlook. Unlike desktop applications, Google Calendar is not platform dependent. Here is an example of a Google Calendar:



Features of Google Calendar

- **Calendar Sharing**: Set up a calendar for your company softball team, and share it with the whole roster. (Your shortstop will never forget about practice again.) Or share with friends and family so you can view each other's schedules side by side.
- **Gmail Integration**: Add your friend's Super Bowl party to your calendar without ever leaving your Gmail inbox. Gmail now recognizes events mentioned in emails.
- **Search**: Find the date of the Baxter family BBQ (you knew it was sometime this summer). Or, search public calendars to discover new events you're interested in and add them to your own calendar.
- **Mobile Access**: Receive event reminders and notifications on your mobile phone.
- **Event Publishing**: Share your organization's events with the world. Learn more with our Event Publisher Guide
- **Languages**: The interface is currently available in English, French, Italian, German, Spanish, Danish, Dutch, Norwegian, Finnish, Swedish, Russian, Chinese-Simplified, Chinese-Traditional, Korean, Japanese, Portuguese and Polish. You can enter calendar information in many other languages, too.
- **Cost**: Free

For more information you can visit the Google Calendar site:

http://www.google.com/googlecalendar/overview.html

Technologies used with the Google Calendar API

The following are the technologies used in the interaction with Google Calendar API.

XML

The exchange of data will be done using XML data format. The data will be placed in the HTTP body. This body will be parsed with the built-in DOM XML commands to obtain the wanted data.

HTTP

Google provides client libraries to interact with the API such as Java, .Net, and PHP. There is no client library for 4th Dimension. Therefore, we will be sending raw HTTP requests and also be receiving raw HTTP responses. The plug-in that will assist in making those requests is 4D Internet Commands. We will be generating `POST` and `GET` requests.

REST (Representational State Transfer)

REST is an architectural style for distributed computing. Put simply, it allows the exchanging of data using XML over HTTP with the use of URL’s. For more information regarding this technology in relation to 4th Dimension, please view “Building a REST Client” Technical Note:
http://www.4d.com/knowledgebase?CaseID=43778

GDATA (Google Data API)

Gdata is a simple standard protocol for reading and writing data used in Google Services such as Google Calendar. It consists of a combination of technologies. It uses Atom 1.0, RSS 2.0 (***R**eally **S**imple **S**yndication*) and Atom Publishing Protocol. These technologies are used for syndicating or distributing web feeds such as blogs and podcasts. It notifies a subscribed client that a new blog has been written or a new podcast is available for download. When publishing Gdata feeds, it conforms to the Atom publishing protocol and requires sending an HTTP `Put` Request. When receiving Gdata, it conforms to the RSS or Atom formats which require sending a HTTP `GET` Request.

Note *The Atom format is the default format. This format will be used in this Technical Note. If you are interested in receiving the data in RSS format, please view http://code.google.com/apis/gdata/reference.html#query-requests*

In addition, the GData protocol does not provide ways to create or delete a feed. That is up to the service.

Note *Google Calendar service currently does not delete feeds. Although it does allow querying of the service to check for deleted feeds in order to update or sync with a client application.*

For more information regarding RSS and Atom feel free to view the suggested links:

RSS Feeds:
http://feedvalidator.org/docs/rss2.html

Atom:
http://en.wikipedia.org/wiki/Atom_%28standard%29

The Example Implementation

The implementation presented in this Technical Note uses the Google Calendar API to fetch all future events from a Google Calendar (or Calendars). There are three main steps to complete this process:

- Authentication
- Request the Calendar List
- Request Future Events

The steps are described in-depth in the following sections.

Authentication

Gdata is able to provide public data feeds or private data feeds. Private data feeds, in the context of the Google Calendar Service is simply a Calendar that has not been made public. Therefore, events from this Calendar cannot be accessed by anyone other than the user who created it and whom he or she has shared the calendar with.

In order to request private data feeds, the obvious detail one needs is a Google Account. A google account can be created here:

https://www.google.com/accounts/NewAccount

In addition, if you already have a Gmail account this also qualifies as a Google account.

Building Request to Authenticate

The following is code that builds the request. Please refer to the project method *G_Authenticate* to see the code in its entirety.

```
$body:="Email="+<>account+"&Passwd="+$pass+"&service="+<>service+"&source="+<>sourc
$body := $body +\r\n"
$body_length:=Length($body)

$request_string:="POST /accounts/ClientLogin "+"HTTP/1.1"+"\r\n"
$request_string:=$request_string+"Content-Length: "+String($body_length)+"\r\n"
$request_string:=$request_string+"Content-Type: application/x-www-form-
urlencoded"+"\r\n"+"\r\n"
$request_string:=$request_string+$body
```

When the code is executed, the variable `$request_string` will contain the following:

```
POST /accounts/ClientLogin HTTP/1.0
Content-Length: 73
Content-Type: application/x-www-form-urlencoded

Email=rmolina@4d.com&Passwd=password&service=cl&source=4D-4Dcal-ver1.0
```

Some key things to make note of in the request is the "\r\n" characters. These are control characters that are needed to separate each line.

Note *To separate the HTTP header and HTTP body, there are two consecutive "\r\n" or CRLF. This follows the HTTP 1.0 standard as explained in RFC 1945.*

The first line in the header specifies the type of request as well as the specific URL (**`/accounts/ClientLogin`**) that tells the web service that the request is attempting to authenticate. Because there is a body in this request, the Content-Length and Content-Type Headers must be included.

In the body, there are four parameters that are passed. Each are separated with an “&”. Here is a description of each parameter.

- **Email**
  This parameter is the email registered with the Google account.

Note *If using a Gmail account, the domain “@gmail.com” is optional when sending account information.*

- **Passwd**
  This parameter is the user’s password.

- **Service**
  This parameter is used to pass the Google service which the user is trying to authenticate. The name used for Google Calendar service is ‘**cl**’. This parameter is required when accessing GData services.

- **Source**
  This parameter is a short string that identifies your application. This normally has the form “companyName-applicationName-versionId”.

These four parameters are the only ones used in the example database. There are three other parameters **accountType**, **logintoken**, and **logincaptcha** that can be used as well that do not apply to the example. For more information regarding these parameters please visit the “Google Account Authentication” page:

http://code.google.com/apis/accounts/AuthForInstalledApps.html

Note *The parameters are case sensitive and should be entered accordingly when inserted in the body of the request.*

Sending the Request

As mentioned previously, sending the request requires the use of 4D Internet Commands. A TCP connection is opened with the following line of code.

```
$error:=TCP_Open (<>hostname;<>port;$tcp_id;3)
```

This request is being sent using Asynchronous SSL protocol. This is highly suggested since the username/email and password is being sent over the web.

Receiving the Response

Once connection is established and a request to the service has been made, then the code simply waits for incoming data. The incoming data will be formatted as an HTTP response. Here is a sample response where the authentication was approved and the service returns a HTTP 200.

```
HTTP/1.0 200 OK
Content-Type: text/plain
Cache-control: no-cache
Pragma: no-cache
Content-Length: 497
Date: Tue, 31 Jul 2007 01:45:46 GMT
Server: GFE/1.3
Connection: Close

SID=DQAAAGcAAADoZbAolHTfJ9ISSH4LizLIbcuzQeKfiOZ9ig_Gia_Rjj3q2l-PdCo61fc139iR-
aklBNqbWd57dPCxKpO62fTdByqgufo7CShY9D3JpPDZvEUcOZFDVL5Z_hYsM-
sbiFlfmjIbXE4VpqtskKJaZEBV
LSID=DQAAAGkAAAD6k9mb5_Kl3-
u5I8A9La4xD67LFEXYFy4G9qWL7YKeOwJkV1_pF3bkgpoZUsKjJ06kgqFS7YX2SGs-
1xjQANwijQlVhkDR6MKJfVAnS4ps1qaweQfXVEUNS5rWDlq-qOipf08T6lkUnHTSP11drTTX
Auth=DQAAAGoAAAD6k9mb5_Kl3-
u5I8A9La4xD67LFEnYFy4G9qWL7YKeOwJkV1_pF3bkgpoZUsljJ05HZIuEfr6qspyAhYVPgLZdh5beUYPURlDo
SICa_dRiGQnDYWs2m8QEs_AP8sWTDbXfJD4yFms35H9tQg4AMN0N
```

The response contains the Auth token. The Auth token provides private access to the specified user's Google Calendar service. These tokens should be kept private as much as possible because if another unauthorized user gets a hold of these tokens, they can gain access to your account. These tokens are specific to the user and service from which they have been generated from. Having these tokens prevents having to authenticate each time a request is being made to the service. In addition to the Auth Token, there are two other tokens, SID and LSID. These tokens at the moment currently do not serve a purpose and are intended for future implementations.

Note *The Auth token generated by the ClientLogin is session type token that remains valid for a set time limit. This time limit is specified by the Google Calendar Service. In my testing, I was not able to determine how long an Auth Token remains valid.*

Along with the HTTP 200 response, you may also receive a HTTP 403 response which normally contains an Error code.

```
HTTP/1.0 403 Forbidden
Content-Type: text/plain
Cache-control: no-cache
Pragma: no-cache
Content-Length: 24
Date: Tue, 31 Jul 2007 02:33:11 GMT
Server: GFE/1.3
Connection: Close

Error=BadAuthentication
```

The error returned is BadAuthentication which means that the username or password is incorrect. Here is list of other error codes that may be returned when trying to authenticate.

Error Code	**Description**
BadAuthentication	The login request used a username or password that is not recognized.
NotVerified	The account email address has not been verified. The user will need to access their Google account directly to resolve the issue before logging in using a non-Google application.
TermsNotAgreed	The user has not agreed to terms. The user will need to access their Google account directly to resolve the issue before logging in using a non-Google application.
CaptchaRequired	A CAPTCHA is required. (A response with this error code will also contain an image URL and a CAPTCHA token.)
Unknown	The error is unknown or unspecified; the request contained invalid input or was malformed.
AccountDeleted	The user account has been deleted.
AccountDisabled	The user account has been disabled.
ServiceDisabled	The user's access to the specified service has been disabled. (The user account may still be valid.)
ServiceUnavailable	The service is not available; try again later.

The next step is to request the calendar list.

Request the Calendar List

The code for requesting a calendar list is not much different compared to a request to receive an authentication token. The main difference here is that there is no body and all information is stored within the header portion of the HTTP request. Here is an example:

```
$request_string:="GET /calendar/feeds/"+$account+" HTTP/1.0"+"\r\n"
$request_string:=$request_string+"Authorization:GoogleLogin"+"auth="+$authkey+"\r\n”
```

Two variables are included in the request string `$account` and `$authkey`.   The `$account` is simply the email account and the `$authkey` is the authentication token.

Here is the request:

```
GET /calendar/feeds/rmolina@4d.com HTTP/1.0
Authorization: GoogleLogin auth=SQAAAGoAAAD6k9mb5_Kl3-
u5I8T9La4xD67LFEXYFy4G9qWL7YKeOwKkV1_pF3bkgpoZUsKjJ05HZIuEfr6qsTyAhYVPgLZdh5TeUYPURlDo
SICa_dRiTMnDkWs2m8QEs_AP8sWTDbXfWD4yFms35H9tQg4AMN0N
```

A Different Response

The response received resulting from the above request may not be what one is expecting. For example:

```
HTTP/1.0 302 Moved Temporarily
Set-Cookie: S=calendar=yymxep90KJs
Location: https://www.google.com/calendar/feeds/rmolina@4d.com?gsessionid=yPmxWp90KJT
Content-Type: text/html; charset=UTF-8
Cache-control: private
Content-Length: 257
Date: Tue, 31 Jul 2007 03:26:37 GMT
Server: GFE/1.3
Connection: Close

<HTML>
<HEAD>
<TITLE>Moved Temporarily</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<H1>Moved Temporarily</H1>
The document has moved <A
HREF="https://www.google.com/calendar/feeds/rmolina@4d.com?gsessionid=yymxep90KJs">her
e</A>.
</BODY>
</HTML>
```

This time the response received is a "302 Moved Temporarily". When receiving this response, it means that the service is redirecting to a new location. The `Location:` header field is what is needed in order to build the next request. The following code returns the redirect URL with the included gsessionID. The code is from the project method *G_GetSessionIdURL.*

```
C_BLOB($1)
C_BLOB($parseblob)
C_INTEGER($positionstart)
C_INTEGER($positionend)
C_TEXT($0)
$parseblob:=$1
$positionstart:=PositionInBlob ($parseblob;"Location")
If ($positionstart>=0)
       DELETE FROM BLOB($parseblob;0;$positionstart)
       $positionstart:=PositionInBlob ($parseblob;"com")
       $positionend:=PositionInBlob ($parseblob;Char(10))
       `remove the end
       DELETE FROM BLOB($parseblob;$positionend;(BLOB size($parseblob)-1))
       DELETE FROM BLOB($parseblob;0;($positionstart+3))
       $0:=BLOB to text($parseblob;Text without length )
Else
       $0:=""
End if
```

A walkthrough of the code above:

- First check to see if you find the string "Location" is in the blob.
- Once finding the position of where the string "Location" starts, delete all bytes before the string.

- Now find where the position of “com” starts
- Find the position of **`char`**`(10)`
- Starting at the position of **`char`**`(10)`,delete all bytes after it.
- Starting at the beginning of the blob, delete all bytes up to and including “com”

The resulting string should look something like this:

```
/calendar/feeds/rmolina@4d.com?gsessionid=yPmxWp90KJT
```

This URL can then be passed on to send another request. In the example database, this URL is passed on to the project method *G_RedirectRequest.* Here is the request that is built and eventually gets sent within the *G_RedirectRequest* method .

```
GET /calendar/feeds/rmolina@4d.com?gsessionid=GocpHEWJ0CM HTTP/1.0
Authorization: GoogleLogin auth=DQSAAHkAAABs-
Kw4XGwQZJEgs6LYu4yzRPIQNvGpkPNe2M18XLOyhVdJo1_MFh-YCY-
smWXUrqk7clC2iJDvJSULhvIQhA_BDYFQuBC6e_dghmR7mq-UN6fq-0i44aEkl89s89lJNQKNX-
sB4pmOb72HCnUeMYLSAsWggSwJIOiXXsUo0fp_2A
```

The service then responds back with the following data feed.

```
HTTP/1.0 200 OK
Content-Type: application/atom+xml; charset=UTF-8
Cache-Control: max-age=0, must-revalidate, private
Last-Modified: Tue, 31 Jul 2007 07:57:45 GMT
Date: Tue, 31 Jul 2007 07:57:45 GMT
Server: GFE/1.3
Connection: Close

<?xml version='1.0' encoding='UTF-8'?><feed xmlns='http://www.w3.org/2005/Atom'
xmlns:openSearch='http://a9.com/-/spec/opensearchrss/1.0/'
xmlns:gCal='http://schemas.google.com/gCal/2005'
xmlns:gd='http://schemas.google.com/g/2005'><id>http://www.google.com/calendar/feeds/r
molina%404d.com</id><updated>2007-07-31T07:57:45.661Z</updated><title type='text'>4D
Inc's Calendar List</title><link rel='http://schemas.google.com/g/2005#feed'
type='application/atom+xml'
href='http://www.google.com/calendar/feeds/rmolina%404d.com'/><link
rel='http://schemas.google.com/g/2005#post' type='application/atom+xml'
href='http://www.google.com/calendar/feeds/rmolina%404d.com'/><link rel='self'
type='application/atom+xml'
href='http://www.google.com/calendar/feeds/rmolina%404d.com'/><author><name>4D
Inc</name><email>rmolina@4d.com</email></author><generator version='1.0'
uri='http://www.google.com/calendar'>Google
Calendar</generator><openSearch:startIndex>1</openSearch:startIndex><entry><id>http://
www.google.com/calendar/feeds/rmolina%404d.com/rmolina%404d.com</id><published>2007-
07-31T07:57:45.665Z</published><updated>2007-07-30T15:43:40.000Z</updated><title
type='text'>4D Inc</title><link rel='alternate' type='application/atom+xml'
href='http://www.google.com/calendar/feeds/rmolina%404d.com/private/full'/><link
rel='http://schemas.google.com/acl/2007#accessControlList' type='application/atom+xml'
href='http://www.google.com/calendar/feeds/rmolina%404d.com/acl/full'/><link
rel='self' type='application/atom+xml'
href='http://www.google.com/calendar/feeds/rmolina%404d.com/rmolina%404d.com'/><author
><name>4D Inc</name><email>rmolina@4d.com</email></author><gCal:timezone
value='America/Los_Angeles'/><gCal:hidden value='false'/><gCal:color
```

```
value='#A32929'/><gCal:selected value='true'/><gCal:accesslevel
value='owner'/></entry></feed>
```

Now it is time to parse this xml. The first step is to remove the HTTP body. The project method that will parse the XML is *`G_ParseCalList`*. Here is a piece of the source code:

```
$positionstart:=PositionInBlob ($responseblob;Char(13)+Char(10)+Char(13)+Char(10))
DELETE FROM BLOB($responseblob;0;($positionstart+4))
```

As stated previously, the HTTP header and HTTP body is separated by two sets of **`Char(13)+Char(10).`** Thus, when calling the **`DELETE FROM BLOB`** command, it deletes everything in the HTTP header up to the '<?xml'.
At this point we can parse the xml.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:gCal="http://schemas.google.com/gCal/2005"
xmlns:gd="http://schemas.google.com/g/2005" xmlns:openSearch="http://a9.com/-
/spec/opensearchrss/1.0/">

  <id>http://www.google.com/calendar/feeds/rmolina%404d.com</id>

  <updated>2007-07-31T08:20:00.874Z</updated>

  <title type="text">4D Inc's Calendar List</title>

  <link href="http://www.google.com/calendar/feeds/rmolina%404d.com"
rel="http://schemas.google.com/g/2005#feed" type="application/atom+xml"/>

  <link href="http://www.google.com/calendar/feeds/rmolina%404d.com"
rel="http://schemas.google.com/g/2005#post" type="application/atom+xml"/>

  <link href="http://www.google.com/calendar/feeds/rmolina%404d.com" rel="self"
type="application/atom+xml"/>

  <author>
    <name>4D Inc</name>
    <email>rmolina@4d.com</email>
  </author>

  <generator uri="http://www.google.com/calendar" version="1.0">Google
Calendar</generator>

  <openSearch:startIndex>1</openSearch:startIndex>

  <entry>
    <id>http://www.google.com/calendar/feeds/rmolina%404d.com/rmolina%404d.com</id>
    <published>2007-07-31T08:20:00.877Z</published>
    <updated>2007-07-30T15:43:40.000Z</updated>
    <title type="text">4D Inc</title>
    <link href="http://www.google.com/calendar/feeds/rmolina%404d.com/private/full"
rel="alternate" type="application/atom+xml"/>
    <link href="http://www.google.com/calendar/feeds/rmolina%404d.com/acl/full"
rel="http://schemas.google.com/acl/2007#accessControlList"
type="application/atom+xml"/>
    <link
href="http://www.google.com/calendar/feeds/rmolina%404d.com/rmolina%404d.com"
rel="self" type="application/atom+xml"/>
```

```
    <author>
      <name>4D Inc</name>
      <email>rmolina@4d.com</email>
    </author>
    <gCal:timezone value="America/Los_Angeles"/>
    <gCal:hidden value="false"/>
    <gCal:color value="#A32929"/>
    <gCal:selected value="true"/>
    <gCal:accesslevel value="owner"/>
  </entry>

</feed>
```

Which Data to Get?

The above Gdata feed contains one calendar entry. Here are the elements that the sample database parses with the project method *`G_ParseCalList`* .

Standard Atom Elements

`/feed/entry/id`
This is a typical element in an Atom type feed. This is a unique identifier that allows the feed entry to be distinguished from other entries.

`/feed/entry/title`
This is simply the title of the entry. In the example feed above, the entry title is '4D Inc'

`/feed/entry/updated`
This element contains information about when the entry was last updated. This is element will be used when trying to synchronize your desktop application with the service.

`/feed/entry/link`
This element provides a link to current feed as well as an HTML version of the feed.

Gcal Namespace Elements

(Elements specifically for Google Calendar)

`gCal:timezone`
This indicates the timezone of the calendar.

`gCal:hidden`
This element indicates whether the calendar is hidden or not.

`gCal:color`
This element contains the RGB color hex value used to highlight a calendar in the user's browser.

`gCal:accesslevel`
This indicates the level of access of the current user that requested the feed.

Google Data Namespace Elements
(Elements for all Google Services)

`gd:where`
This indicates location of the event.

Note *When parsing the Google Data Namespace Elements or Gcal Namespace elements, treat them as you would any other element. For example, to obtain the data from*

```
<gCal:timezone value="America/Los_Angeles"/>
```

Use the following code:

```
$currentref2:=DOM Get XML element($currentref;"gCal:timezone";1;$elementValue)
DOM GET XML ATTRIBUTE BY NAME($currentref2;"value";$attribValue)
INSERT ELEMENT($caltimezone;$i)
$caltimezone{$i}:=$attribValue
```

For more information regarding Google Data Namespace Elements or Gcal Namespace elements, please visit:

http://code.google.com/apis/calendar/reference.html#gcal_reference
http://code.google.com/apis/gdata/elements.html#gdReference

Requesting Future Events

With an auth token and a list of calendars, future events can be requested. This process is quite similar to the example provided regarding requesting a calendar list. The main differences will simply be the actual request sent and the xml data received.

```
$request_string:="GET "+$feed+"?futureevents=true"+" HTTP/1.0"+"\r\n"
$request_string:=$request_string+"Authorization: GoogleLogin"+" auth="+$authkey+"\r\n"
```

The `$feed` variable contains data from alternative link which came from the attribute 'alternate' from the element **`/feed/entry/link`** in the previous example.

```
GET /calendar/feeds/rmolina%404d.com/private/full?futureevents=true HTTP/1.0
Authorization: GoogleLogin auth=DEAAAGoARADBe8kbX4PmUv_spkGHZS-
K8E8DGbxVh4fzCVDCYkrAW3bOsSqTnN1OwLZKtcq-SxipqxWbqRC81HKik7TydVdkpSBEtqMm87-yXcYXx-
B45_mZ6iX_yPYgrJPdeKbFXl2f4c99-aB11C01-7OUEfO-
```

The parameter `futureevents=true` is a calendar query parameter. It is a shortcut that allows requesting events scheduled at future times. It can accept either the values True or False. True will request for future events, while false will simply notify the Google Calendar Service to ignore the parameter.

```
HTTP/1.0 302 Moved Temporarily
Set-Cookie: S=calendar=xKtYlMByjDE
Location:
https://www.google.com/calendar/feeds/rmolina%404d.com/private/full?futureevents=true&
gsessionid=xKtYlMByjDE
Content-Type: text/html; charset=UTF-8
Cache-control: private
Content-Length: 294
Date: Tue, 31 Jul 2007 12:19:58 GMT
Server: GFE/1.3
Connection: Close
```

As shown previously, the request may be redirected. We are then able to receive the feed. Below is the xml that gets parsed (this xml does not display the feed tag in order to put emphasis on the single entry).

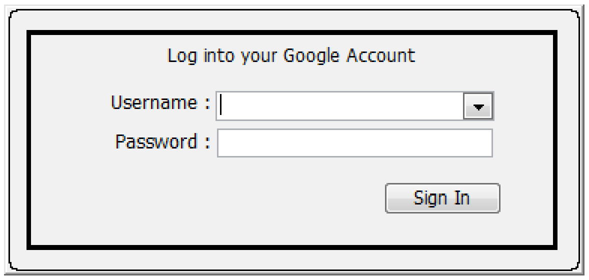
```
 <entry>

<id>http://www.google.com/calendar/feeds/rmolina%404d.com/private/full/m772g7
1facff1mf1jsh9ru5c4c</id>
    <published>2007-07-30T01:15:17.000Z</published>
    <updated>2007-07-30T01:17:35.000Z</updated>
    <category scheme="http://schemas.google.com/g/2005#kind"
term="http://schemas.google.com/g/2005#event"/>
    <title type="text">4D 2007 Summit</title>
    <content type="text">This 3 day event will be fun-filled as well
informative. </content>
    <link
href="http://www.google.com/calendar/event?eid=bTc3Mmc3MWZhY2ZmMW1mMWpzaDlydT
VjNGMgcm1vbGluYUA0ZC5jb20" rel="alternate" title="alternate"
type="text/html"/>
    <link
href="http://www.google.com/calendar/feeds/rmolina%404d.com/private/full/m772
g71facff1mf1jsh9ru5c4c" rel="self" type="application/atom+xml"/>
    <link
href="http://www.google.com/calendar/feeds/rmolina%404d.com/private/full/m772
g71facff1mf1jsh9ru5c4c/63321441455" rel="edit" type="application/atom+xml"/>
    <author>
      <name>4D Inc</name>
      <email>rmolina@4d.com</email>
    </author>
    <gd:comments>
      <gd:feedLink
href="http://www.google.com/calendar/feeds/rmolina%404d.com/private/full/m772
g71facff1mf1jsh9ru5c4c/comments"/>
    </gd:comments>
    <gd:eventStatus
value="http://schemas.google.com/g/2005#event.confirmed"/>
    <gd:visibility value="http://schemas.google.com/g/2005#event.default"/>
    <gd:transparency
value="http://schemas.google.com/g/2005#event.transparent"/>
    <gd:when endTime="2007-10-20" startTime="2007-10-16">
      <gd:reminder method="alert" minutes="10"/>
    </gd:when>
    <gd:where valueString="Memphis, Tennessee"/>
  </entry>
```

Again, there is not much difference from the previous example. It has the normal feed/entry/id element as well as the others discussed in the calendar list request example. In addition, this event feed has the element `feed/entry/content` allowing to read in a description of the event. Another distinguishable difference is the `gd:when` element. It has an endTime attribute as well as a startTime attribute to specify how long the event will last.

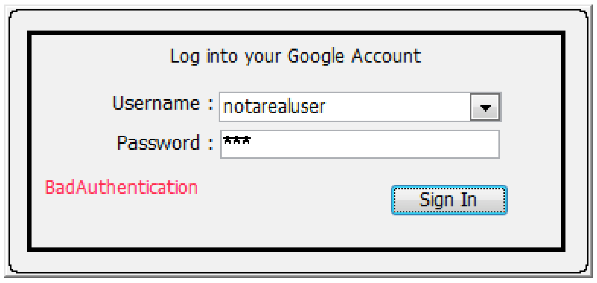
Using the Sample Database.

This Technical Note comes with a sample database. This sample database provides a simple demonstration of how to authenticate with the Google Calendar Service as well as download a calendar list and calendar events.

When launching the application you should be greeted with the following log-in screen.



If you do not enter a correct password, you will get the following display.

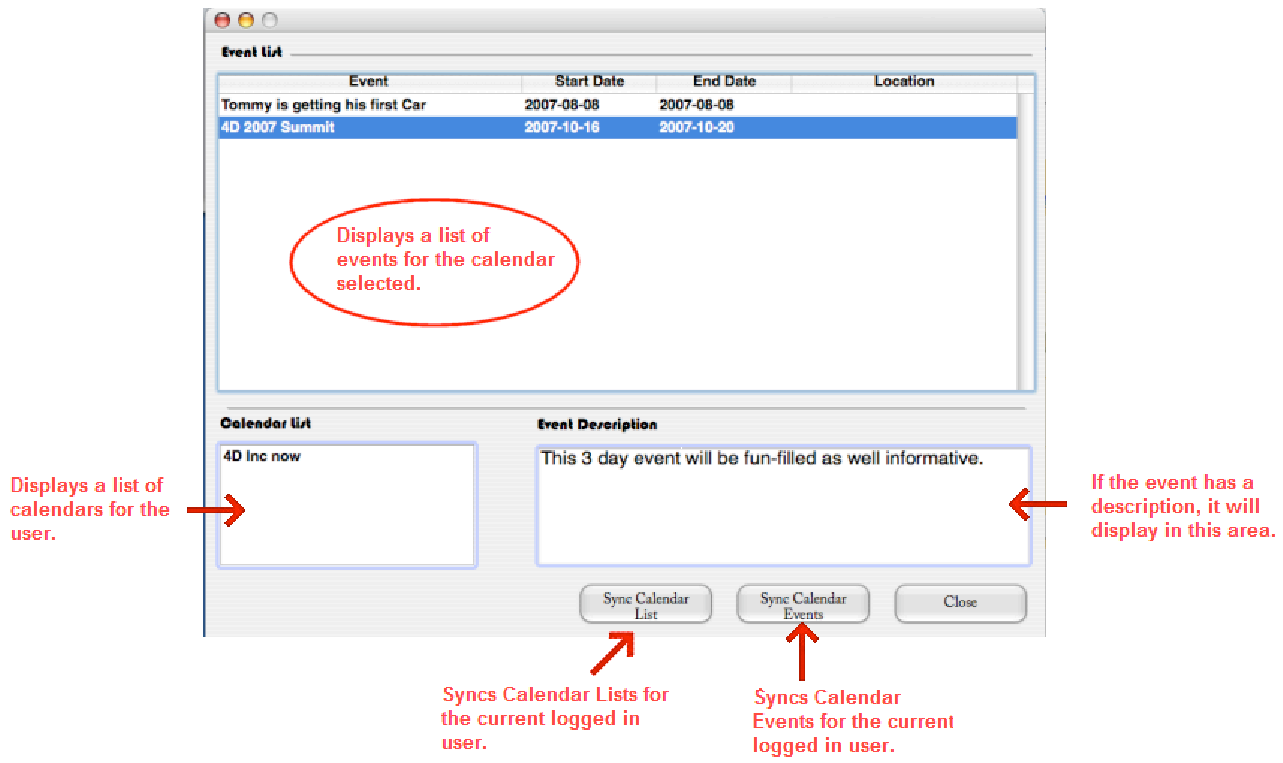

Log into your Google Account

Username : notarealuser

Password : ***

BadAuthentication

After successfully logging in, the following window will appear:


Event List
Displays a list of events for the calendar selected.
Calendar List
Event Description
Displays a list of calendars for the user.
If the event has a description, it will display in this area.
Sync Calendar List
Sync Calendar Events
Close
Syncs Calendar Lists for the current logged in user.
Syncs Calendar Events for the current logged in user.

Event	Start Date	End Date	Location
Tommy is getting his first Car	2007-08-08	2007-08-08	
4D 2007 Summit	2007-10-16	2007-10-20	

4D Inc now

This 3 day event will be fun-filled as well informative.

The first thing you would want to do is click on the 'Sync Calendar List' button. This will download a list of your calendars. Once you have downloaded your calendars, then you can click on 'Sync Calendar Event' button which will then download all future events for your list of calendars.

Conclusion

This Technical Note showed how 4th Dimension may interact with a REST Web service such as Google Calendar using 4D Internet Commands.

Related Resources

Google Calendar Data API Overview
http://code.google.com/apis/calendar/overview.html

Google Calendar Data API Reference Guide
http://code.google.com/apis/calendar/reference.html

Receiving RSS Data feeds
http://code.google.com/apis/gdata/reference.html#query-requests

Google Calendar Overview
http://www.google.com/googlecalendar/overview.html

RSS Feeds
http://feedvalidator.org/docs/rss2.html

Atom
http://en.wikipedia.org/wiki/Atom_%28standard%29

Building a REST Client (Technical Note)
http://www.4d.com/knowledgebase?CaseID=43778