

[Go to content](#) [Go to menu](#)

Richard J. Prinz

The devil is in the details

Using Google service discovery API's with Pharo Smalltalk

Wednesday, March 18, 2015



While working on a project which uses some Google API's I came across Googles API registry. Under <https://www.googleapis.com/discovery/v1/apis/> all API's for Google services like Google Drive or Gmail are described. More infos can be found at the [Google APIs Discovery Service](#) page. From the API registry a client could discover services and automatically build proxy classes and methods based on the provided metadata. Thats what I have done for Pharo Smalltalk. Other companies like Microsoft also provide some form of

service discovery for their service API's (see <https://msdn.microsoft.com/en-us/office/office365/api/discovery-service-rest-operations>)

I wrote a very simple Pharo solution which allows it to query the Google service discovery for an API (e.g. Google Drive) and to generate proxy classes for the selected API. This makes using the Google services from Pharo very easy.

Installation

First start installing:

[View Raw Code](#) [?](#)

```

1. Gofer new
2.   url: 'http://www.min.at/prinz/repo/gapi';
3.   package: 'ConfigurationOfGoogleApi';
4.   disablePackageCache;
5.   load.
6.
7. (Smalltalk at: #ConfigurationOfGoogleApi) loadBleedingEdge.
  
```

License: [MIT](#)

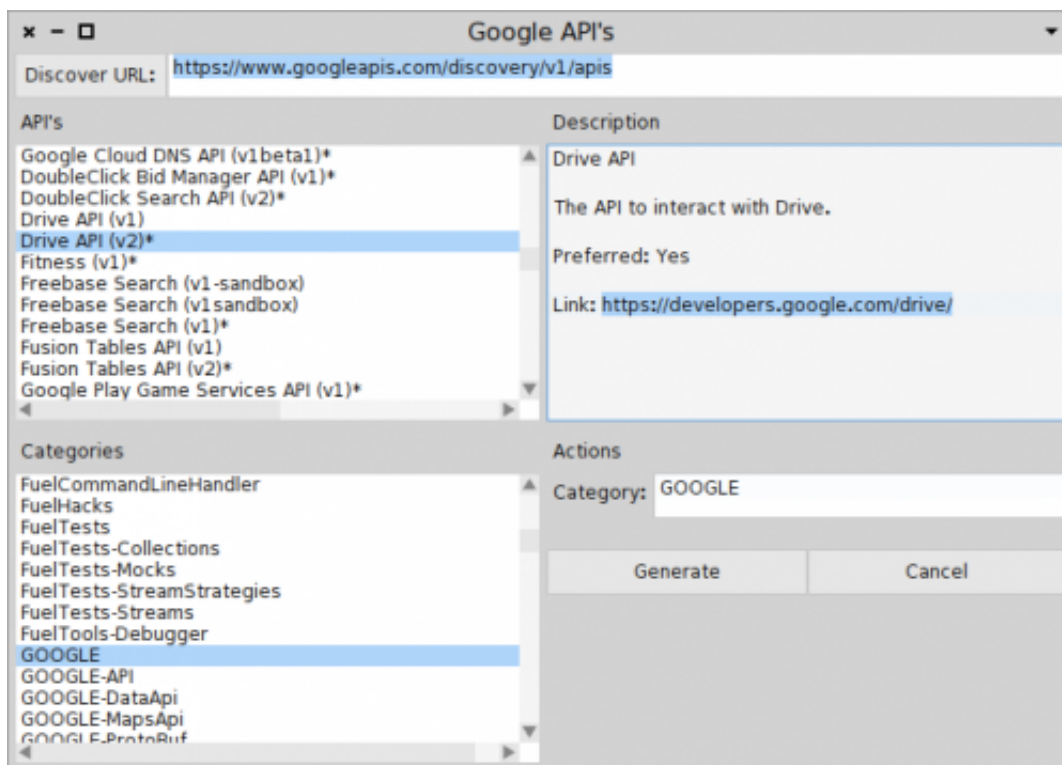
Generating code

To query for an API open the API browser with:

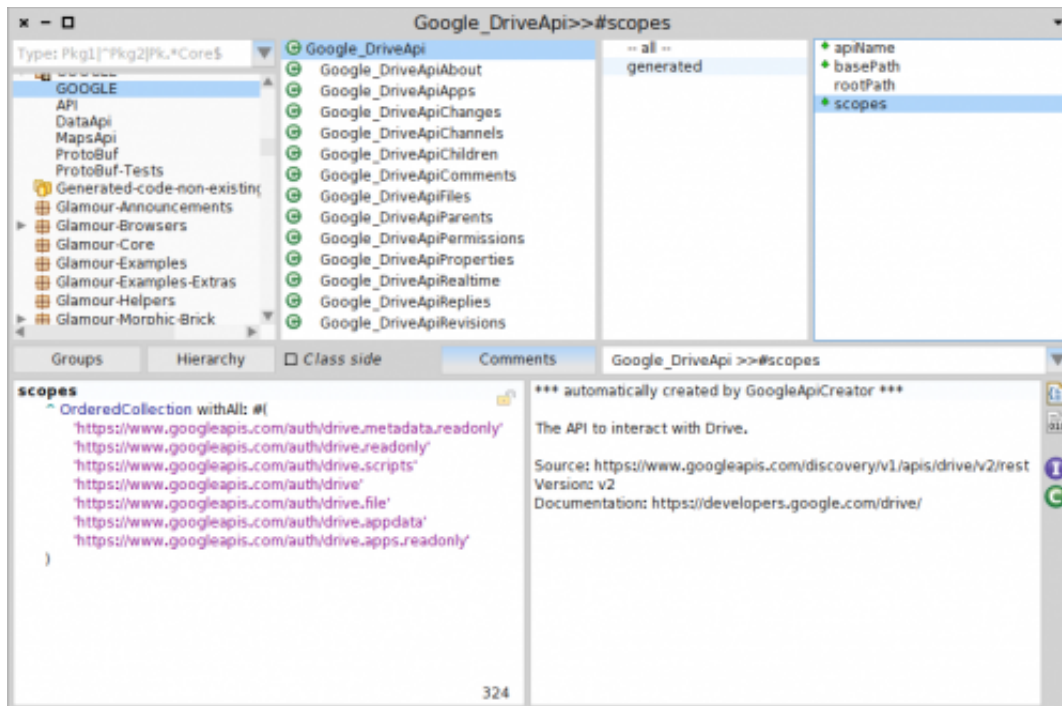
[View Raw Code](#) [?](#)

1. **GoogleApiBrowser** open.

The API browser lists all available service API's including their documentation. Actual API versions (preferred) are marked with an asterisk. These are actual versions of Google API's. Older versions of API's are also listed and you can use them too but they may be removed sometime. Select an API and an existing or new class category under which to create the proxy classes.



After clicking the **Generate** button and selecting **Yes** that you really want to generate the classes in the given category you can view them with the system browser. The example shows the clases for the Google Drive API.



Classes are named **Google<apiName><serviceName>**. Class and method comments are automatically generated from API metadata. Mandatory arguments are method parameters. Optional arguments are passed using an options dictionary.

Using an API

The following steps describe the process of developing a simple program which creates a text file in the Google Drive of a user.

To actually use a Google API some more steps are involved. All Google API's use OAUTH for authentication and every API call must belong to a registered application. To do this you first need a Google account. Next you must create a project in the [Google Developer Console](https://console.developers.google.com/).

I have described the process of doing this in [another post](#). For this solution place the downloaded JSON file in a directory named **google_api_data** and rename the JSON file to **<apiName>**.

<appName>.config.json. Where **<apiName>** is the name of the Google API and **<appName>** can be used to have more than one Google API project of the same type in a single Pharo image. The name of an API can be found with the `#apiName` method. In our example:

[View Raw Code](#) [?](#)

```
1. GoogleDriveApi new apiName. 'drive'
```

The default application name is **default**. So the JSON file name should be **drive.default.config.json**.

There are some predefined examples available in the *GoogleApiExamples* class. The `#addTestFileToGoogleDrive` method contains our example and looks as follows:

[View Raw Code](#) [?](#)

```
1. addTestFileToGoogleDrive
2.
   | fileApi fileName localFilePath remoteFile now result |
3.
4.     " Add a simple text file to Google Drive. See "
5.
6.     " https://developers.google.com/drive/v2/reference/files
7.     /insert "
8.
9.     " create and authenticate API "
10.    fileApi := GoogleDriveApiFiles new.
11.    fileApi authenticate.
12.
13.    " create a local file "
14.    now := DateAndTime now.
15.    fileName := ( '{1}{2}{3}-{4}{5}{6}' format: {
16.        now year . now month asTwoCharacterString .
17.        now dayOfMonth asTwoCharacterString .
18.        now hours asTwoCharacterString .
19.        now minutes asTwoCharacterString .
20.        now seconds asTwoCharacterString } ), '.txt'.
21.
22.    localFilePath := (FileSystem workingDirectory) /
23.        'google_api_data_tests' / fileName.
24.
25.    localFilePath asFileReference writeStreamDo: [ :stream |
26.
27.        stream nextPutAll:
28.
29.        'I am a test file created by Pharo smalltalk named',
30.        ' (' , fileName, ')' ].
31.
32.    " define remote file parameters "
33.    remoteFile := Dictionary new.
34.    remoteFile
35.        add: 'title'          -> fileName;
36.        add: 'description'    -
```

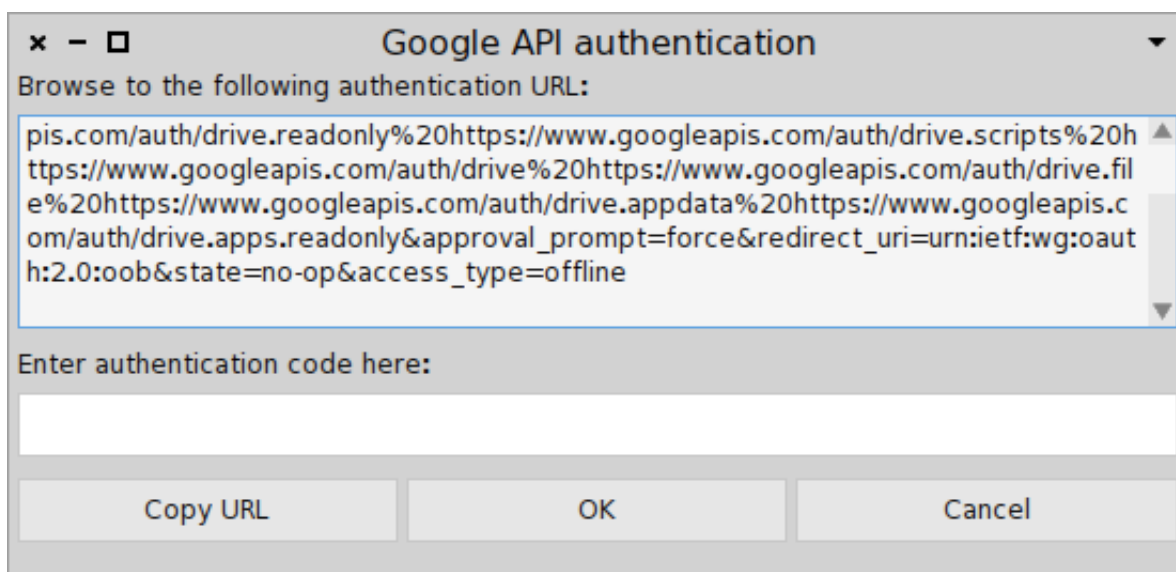
```

32. > 'A file created with Pharo SmallTalk';
33.     add: 'mimeType'      -> 'text/plain';
34.     add: 'CONTENT'      ->
35. ( ( FileStream readOnlyFileNamed: localFilePath )
36.     contents ).
37. " send request "
38.
39. [ result := fileApi insert: remoteFile ] on: Error do: [
40.     :ex |
41.         ex inspect.
42.         self halt ].
43. " after transfer delete local file "
44. localFilePath asFileReference delete.
45. ^ result

```

Each API call returns a two elements array with a *ZnResponse* holding the http status of the call and a dictionary containing the response of the called service.

As every Google API uses OAUTH you must call *#authenticate* before using any other method of an API. The first time this is done (meaning the API was never authenticated) the authentication dialog is shown.



Copy and paste the shown URL to your web browser and open it. You first must login to your Google account (if not already done so) and will then be prompted with a security dialog showing all requested rights by your application. If you trust your app click accept and you will come to a page with an

authentication code. Copy and paste this code back to the Pharo authentication dialog and click OK. The authentication information is stored in the same directory as the JSON file before, is named **<apiName>.<appName>.auth.json** and contains OAUTH tokens. This process is only done if there is no local authentication file available. If a local auth file is available the *#authenticate* method uses it.

Now when you do

[View Raw Code](#) [?](#)

```
1. GoogleApiExamples addTestFileToGoogleDrive.
```

you should end up with a textfile named from current date and time in your Google Drive. To verify this you can use another example which lists all available files in Google Drive in a Transcript.

[View Raw Code](#) [?](#)

```
1. GoogleApiExamples listGoogleDriveFiles.
```

The code for listing the files looks like:

[View Raw Code](#) [?](#)

```
1. listGoogleDriveFiles
2.     | files fileApi result items |
3.
4.     " list files in Google Drive in the Transcript see "
5.
6.     " https://developers.google.com/drive/v2/reference/files
7.     /list "
8.
9.     fileApi := GoogleDriveApiFiles new.
10.    fileApi authenticate.
11.
12.    " send request "
13.    [ result := fileApi list ] on: Error do: [ :ex |
14.        ex inspect.
15.        self halt ].
16.
17.    " display listing in transcript window "
18.    files := result at: 2.
19.    items := files at: 'items' ifAbsent: [ Array new ].
20.    items do: [ :item || fileName labels isDeleted |
```

```

19.   fileName := item at: 'title' ifAbsent: [ 'unknown' ].
20.   labels := item at: 'labels' ifAbsent: [ Dictionary new ]
21.   .
22.   isDeleted := labels at: 'trashed' ifAbsent: [ false ].
23.       Transcript show: fileName, ' ',
24.           [ isDeleted
25.               ifTrue: [ '(deleted)' ]
26.               ifFalse: [ '' ] ] value; cr ].
27.
28.   ^ result

```

Status



This is the first version of this project. The Google API's I have used so far work like expected but I have not tested every single API method so it might be possible that one or another function do not work.

Posted by prinz (1318 views) Filed under [Programming](#), [Smalltalk](#)
[FitBit data export using node.js and a Raspberry PI](#) »
[« Raspberry PI power plug case](#)

• Admin area

- [Login](#)

• Menu

- [Home](#)
- [Blog](#)
- [Flipboard](#)
- [Publications](#)
- [Software](#)
- [Electronics](#)
- [HAM Radio](#)
- [Contact](#)
- [Internal](#)
-  [rss](#)  [atom](#)

• Categories

- [General](#)
- [Hacking](#)
- [Hardware](#)
 - [Embedded](#)
 - [Arduino](#)
 - [Raspberry PI](#)
- [Programming](#)
 - [Smalltalk](#)
 - [node.js](#)
 - [Windows](#)
 - [Unix/Linux](#)
 - [Mobiles](#)
- [Google Glass](#)
- [HAM Radio](#)
- [R/C](#)

• Archives

- [2015](#)
 - [June](#)
 - [April](#)
 - [March](#)
- [2014](#)
 - [December](#)
 - [October](#)
 - [May](#)
 - [March](#)
 - [January](#)
- [2013](#)
 - [September](#)
 - [July](#)
 - [April](#)
 - [March](#)
 - [February](#)

• Last 10 entries

- [Walkera UP02 software clone: UP42](#)
- [RX2635H using the ITG-3205 mems gyro](#)
- [RX2635H using external oscillator and serial port](#)
- [Walkera RX2635H hello world firmware](#)
- [Walkera quadcopter receiver components \(RX2635H\)](#)
- [Walkera quadcopter firmware hacking](#)
- [Raspberry PI power plug case](#)
- [Using Google service discovery API's with Pharo Smalltalk](#)
- [FitBit data export using node.js and a Raspberry PI](#)
- [Walkera X-Z-18 internals](#)

• Search

o

Search

Based in parts on [Simpla theme](#). Ported to [FlatPress](#) by [MIN.at](#). [XHTML 1.0 Strict](#)