Some investigations on cloud services.

**1. Introduction**

This article is to share my investigations about some of cloud services and suggest the way to integrate them into Exiv2. The investigated cloud services are Google Drive and Dropbox, which are the most popular services.

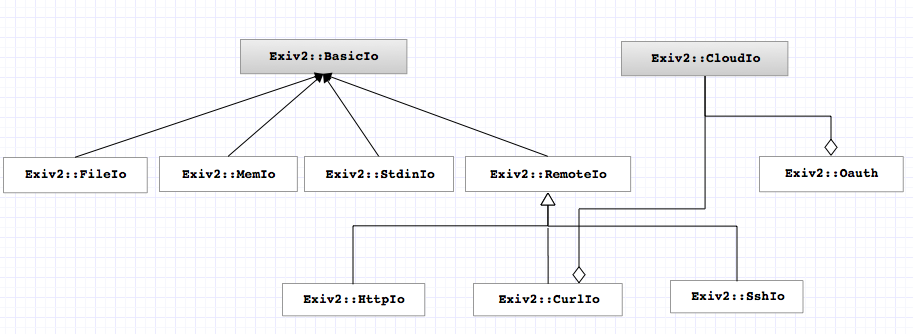
As enterprises integrate their existing IT infrastructures and IT resources with the sharable cloud paradigm, it is imperative for cloud enablers to provide a uniform API that these enterprises can use to tailor the cloud to their business processes. Hence, the way we integrate Google Drive and Dropbox into Exiv2 should be applicable to other cloud services.

**2. Architecture**

Most of cloud services provide Representational State Transfer (RESTful) API to allow users to perform the operations (view, edit, delete, share, …) on their resources. The API is mostly based on the HTTP/HTTPS protocol and GET, POST, PUT, and DELETE requests may be all used. The API presupposes no particular structure in the Uniform Resource Identifier (URI) space. The starting point is a URI, supplied by the cloud service provider, that identifies the cloud itself. The cloud's representation contains URIs for the other resources in the cloud. Operations on the cloud resources are performed by making an HTTP request against the URI of the resource.

Before accessing the API, the applications are required to ask users to authenticate as well as authorize the permission to access the user data. Most of the cloud services use OAuth 2.0 to do that.

The following is the way we should integrate the cloud services to Exiv2.

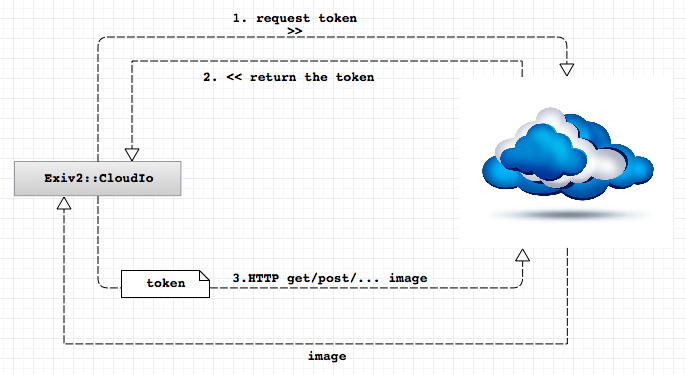


*Fig. 1.1 The architecture of CloudIo*

The CloudIo should contains 2 components: Exiv2::Oauth and Exiv2::CurlIo.

The Exiv2:Oauth is to request and receive credentials to access the API. The credentials in this case are often the tokens. The tokens are attached to the request headers that are sent to RESTFUL API to get the data.

The Exiv2::CurlIo is to retrieve the data. It should be similar as we access a particular image URL because the API is mostly based on the HTTP/HTTPS protocol (We should easily get byte-ranges and PUT byte-ranges)



*Fig. 1.1 the flow of CloudIo*

Note: the above diagram indicates briefly the flow of CloudIo. It should be much more complicated in details. The step 1 & 2 should contain more sub-steps before the token is returned (e.g. the step when user provides username, password).

**3. Google Drive**

The documentation of Google Drive (https://developers.google.com/drive/) is very clear and detail. I don’t intend to repeat it at here. I just want to highlight some key points.

+ Google Drive uses Oauth 2.0, more details at <https://developers.google.com/drive/about-auth>

+ The API reference is at <https://developers.google.com/drive/v2/reference/>. It's all done with fairly normal HTTP GETs, POSTs, PUTs.

+ It’s very easy to test Google Drive API because they provide APIs Explorer below each page (e.g <https://developers.google.com/drive/v2/reference/files/list>).

+ For each file on Google Drive, it has a unique Id. To download the file, we first need to request to

*GET https://www.googleapis.com/drive/v2/files/fileId*

The response contains basic info, some metadata of the file, the URL to download the file.

+ We can upload and update the file via PUT and POST method (Maximum file size: 10GB)

**4. Dropbox API**

+ The RESTFUL API of dropbox is at <https://www.dropbox.com/developers/core/docs>

It’s very clear and detail. It’s very similar as Google Drive and it's all done with fairly normal HTTP GETs, POSTs, PUTs.

+ Dropbox also uses Oauth 2.0