From Delphi to the cloud

Introduction

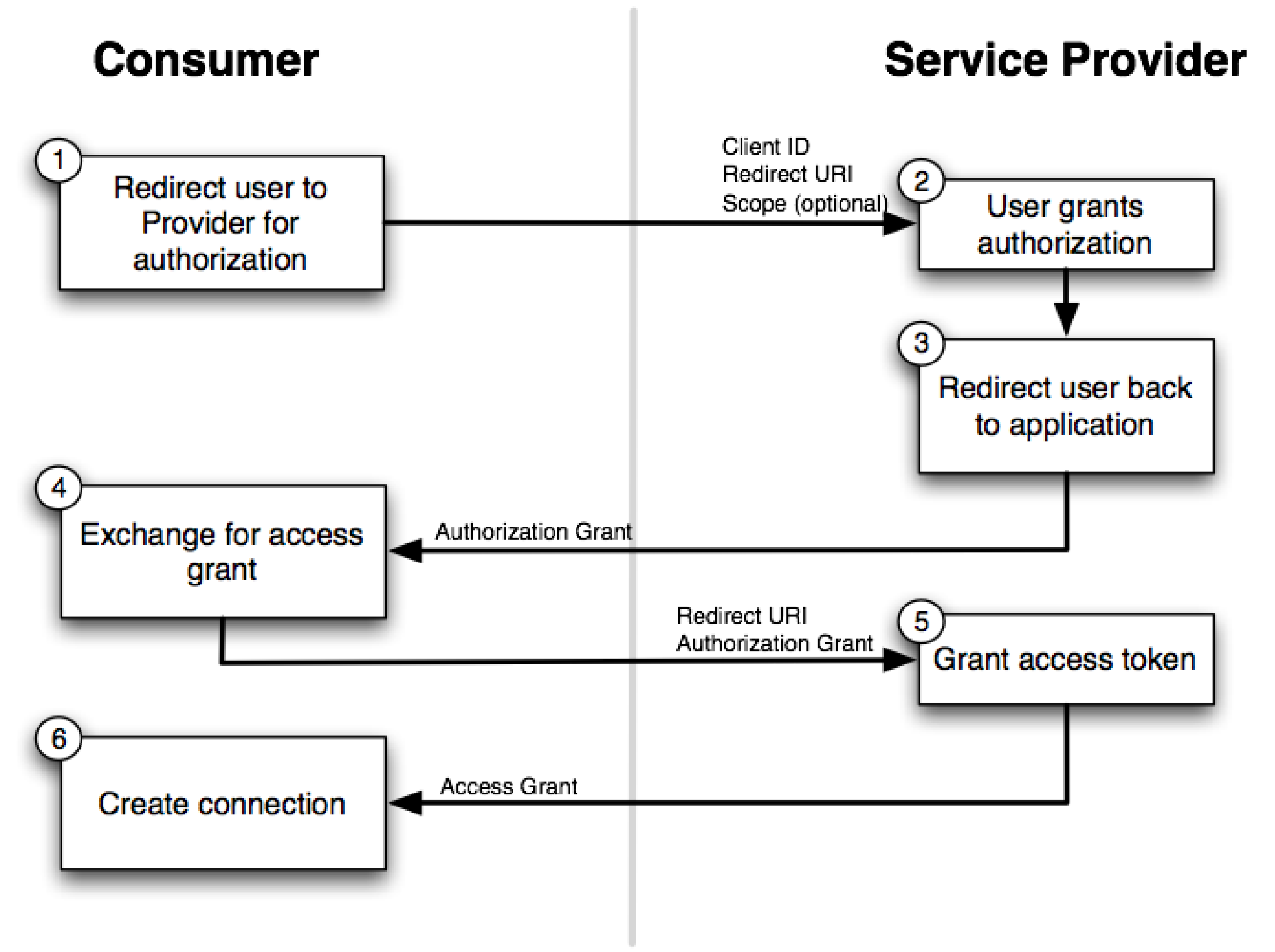
Increasingly data and services hosted in the cloud become accessible by authenticated REST APIs for client applications, be it web clients, mobile clients and thus also classic Windows desktop clients. There are several challenges though to consume this data and services. As soon as this is related to personal data, there is a need for authentication & authorization. The defacto standard protocol for authentication & authorization is OAuth. Once gained authorization, the data and services are typically accessed by HTTPS GET, PUT, POST, DELETE methods and exchange of JSON packets. In this article we'll present the general flow to use these services from a Delphi application and how components from TMS software help in making this a lot easier.

OAuth

The defacto protocol for authentication and authorization is OAuth. Meanwhile there is the OAuth v1.0 and OAuth v2.0 spec. More information can be found at: http://oauth.net/
In a nutshell, the OAuth protocol is a series of steps that need to be done for a consumer of a REST API to do authentication and gain authorization. This involves:

- registering for a key of the application with the service provider
- identifying the application that will access the APIs by means of a key
- inform the access rights the app wants
- performing a login on a web page off the service provider
- retrieve an access token

Unfortunately, the OAuth 1.0 and OAuth 2.0 specification leave a lot of options for the service provider to implement. Each service provider implements the protocol in slightly different ways so creating a universally usable Delphi component for every OAuth service implementation is not really possible.



We have created a class though, TCloudBase that provides a lot of helper methods to perform these variations of OAuth implementations.

Interface of TCloudBase with some helper methods:

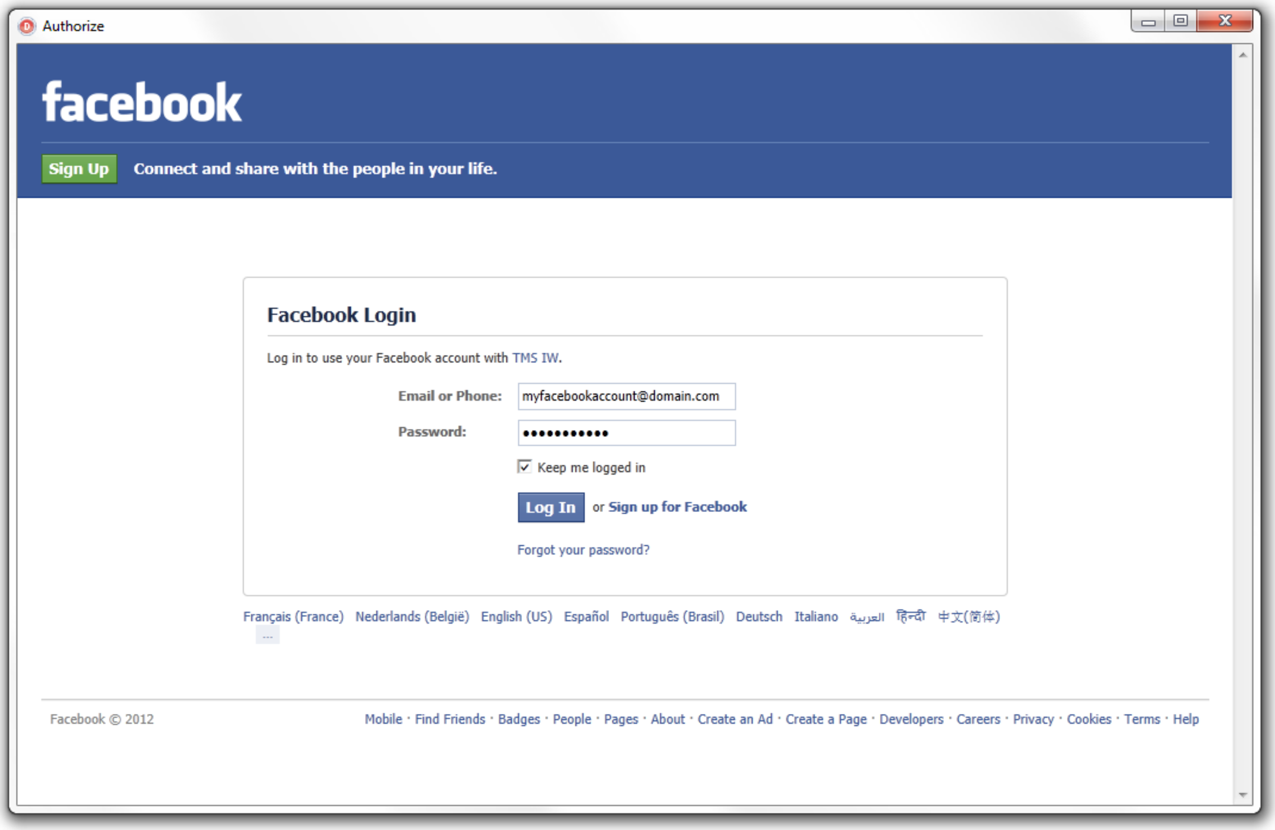
```
TCloudBase = class(TComponent)
public
  // method to start authentication
  procedure DoAuth; virtual;

  // helper methods to sign HTTPS REST API calls
  function GenerateNonce: string;
  function GenerateTimeStamp: string;
  function GenerateSignatureBase(): string;
  function GetSignature(): string;

  // helper methods to perform HTTPS REST API calls
 function HttpsDelete(): Integer;
```

```
  function HttpsPost(): Integer;
  function HttpsUpdate(): Integer;
  function HttpsPut(): string
  function HttpsGet(): string;
published
  // class property to hold application key & secret for OAuth
 property App: TCloudApp;
end;
```

In TCloudBase is a virtual method DoAuth that should be called to perform the authentication process. In a typical implementation, first an authorization token will be requested from the service. To do this, a HTTPS GET call with request parameters that identify the app with its key is executed and depending on the service this can return an authorization token via plain text, XML or JSON. A next step in the authentication is letting the user perform a login on a web page of the service provider. This is for security reasons a required step. The login on this page is done either via a TWebBrowser instance assigned to TCloudBase or, when nothing is assigned, a built-in dialog with a web browser that is started from TCloudBase.


Authorize
facebook
Sign Up
Connect and share with the people in your life.

Facebook Login

Log in to use your Facebook account with TMS IW.

Email or Phone: myfacebookaccount@domain.com

Password: •••••••••••

☑ Keep me logged in

Log In
or Sign up for Facebook
Forgot your password?
Français (France) Nederlands (België) English (US) Español Português (Brasil) Deutsch Italiano العربية हिन्दी 中文(简体) ...
Facebook © 2012
Mobile · Find Friends · Badges · People · Pages · About · Create an Ad · Create a Page · Developers · Careers · Privacy · Cookies · Terms · Help

After a successful login, the service provider typically redirects to a callback URL that is then intercepted and as parameter of this redirect URL is in some cases the access token included, in other cases, another HTTPS GET request should be done to the service in order to obtain the access token.

Once the access token is obtained, this access token is used to sign further HTTPS requests. For some services, the access token will not expire, in other cases it has a limited lifetime. In some cases, the service provider offers a refresh token with which a new access token can be automatically obtained, for some service providers this is not the case. As you can start to understand, all these subtle differences make it impossible to provide a universal component to access whatever cloud service.

Signed REST requests

After obtaining the access token, we can start performing various HTTPS requests to the service. The HTTPS request URLs typically need to be signed. This means that authorization info must be added sometimes via request parameters, sometimes via the request header. With this info the service can determine that caller is authorized. Unfortunately, also here, different service providers expect the signing in different ways. The most common way to perform the signing with an authorization section added in the HTTPS request header where a signature is calculated based on the app key, the access token and the URL itself. According to the OAuth protocol the signature can be encoded with PLAINTEXT, HMAC-SHA1 or RSA-SHA1 methods. A typical authorized HTTPS request is:

```
GET /photos?size=original&file=vacation.jpg HTTP/1.1
Host: photos.example.net:80
Authorization: OAuth realm="https://photos.example.net/photos",
   oauth_consumer_key="dpf43f3p2l4k3l03",
   oauth_token="nnch734d00sl2jdk",
   oauth_nonce="kllo9940pd9333jh",
   oauth_timestamp="1191242096",
   oauth_signature_method="HMAC-SHA1",
   oauth_version="1.0",
   oauth_signature="tR3%2BTy81lMeYAr%2FFid0kMTYa%2FWM%3D"
```

Here, the request wants to download a photo vacation.jpg from a users account from service photos.example.net on behalf of the user.

With the TMS TCloudBase class, this header would be created with following code:

```
var
  url,sigbase: string;

  url := 'https://photos.example.net/photos?size=original&file=' + FileName;

  // generate the authorization request parameters (via property RequestParams)
with the app key and obtained access token:
  AddOAuthRequestParams(App.Key, Token_Access);
  // get the base signature string
  sigbase := GetOAuthSigBase(url,App.Key,'GET');
  // calculate the signature with HMAC-SHA1 encryption
  sig := GetSignature(sigbase, App.Secret, Token_Access_Secret);
  // add the signature to the request parameters
  RequestParams.Values['oauth_signature'] := URLEncodeRFC3986(sig);
  // create the HTTP request header
 hdr :=  'Authorization: OAuth ' + RequestParamText;
  // perform the actual signed HTTPS request
  resdat := HTTSGet(url,hdr);
```

Direct route to cloud storage with TMS components

While the TCloudBase component is designed to get you started to implement access to any web service, we have already created several components descending from TCloudBase that hide any of the complexities of OAuth handling and signed REST API accesses. These first 3 components we created are for cloud storage services: TAdvGDrive, TAdvSkyDrive and TAdvDropBox. These components allow you to access your Google Drive, Windows SkyDrive or DropBox. Note that these components descend from TCloudStorage that in turn descends from TCloudBase. The TCloudStorage class is an abstract class that adds file handling methods and a collection to hold the tree of files on the cloud storage service.

To start using any of these components, first register your application at Google, Windows Live or DropBox. There, you obtain the application key and secret. To start getting access to your favorite cloud storage, all you need to do is:

```
begin
  CloudStorage.App.Key := '************';
  CloudStorage.App.Secret := 'xxxxxxxxxxxxxx';
  CloudStorage.DoAuth;
end;
```

after the authentication is successful, the event OnReceivedAccessToken is triggered and from there, we can obtain the entire tree of files on the cloud storage with GetDriveInfo.

```
procedure TForm1.CloudStorageReceivedAccessToken(Sender: TObject);
begin
  CloudStorage.GetDriveInfo;
end;
```

The entire tree is now available via CloudStorage.Drive which is a collection of files & folder with properties:

```
 TCloudItem = class(TCollectionItem)
  private
  public
    // name of the file or folder
   property FileName: string;
    // when the item is a folder, the folder collections contains all files in
this folder
    property Folder: TCloudItems;
    // size of the file when it is a file type item
    property Size: int64;
    // is ciFile when the item is a file, ciFolder when the item is a folder
   property ItemType: TCloudItemType;
  end;
```

Further, following methods are available to download, delete or upload a file:

```
  // Downloads the item ci and saves as filename TargetFile
  TCloudStorage.Download(ci: TCloudItem; const TargetFile: string);
  // Deletes the online storage file ci
 TCloudStorage.Delete(ci: TCloudItem);
  // Uploads local file FileName in the cloud storage folder Folder
  TCloudStorage.Upload(Folder: TCloudItem; const FileName:string): TCloudItem;
```

The progress of these file uploads or downloads can be tracked with the events TCloudStorage.OnDownloadProgress , TCloudStorage.OnUploadProgress.

Access to other cloud services

While the current release just offers access to 3 cloud storage services, the team at TMS software is already working on several more Delphi components to make it easy to access other services. Components will be added to access Twitter, Facebook, Windows Live Contacts, Windows Live Calendar, Google Calendar and many many more.

For example, to post on your Facebook or Twitter account from Delphi, following code can be used:

Twitter sample:

```
  AdvTwitter1.App.Key := '********************';
  AdvTwitter1.App.Secret := 'xxxxxxxxxxxxxxxxxxxxxxxxxxxx';
  AdvTwitter1.DoAuth;
```

After authorization, a tweet can be done with:

```
 AdvTwitter1.Tweet('Hello Twitter from Delphi');
```

Facebook sample:

```
  AdvFacebook1.App.Key := '**********************';
  AdvFacebook1.App.Secret := 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';
  AdvFacebook1.DoAuth;
```

After authorization, a post on your Facebook stream can be done with:

```
  AdvFacebook1.Post('Hello Facebook from Delphi');
```

Conclusion

While OAuth 1.0, OAuth 2.0 were attempts to standardize authentication and authorization for cloud services, different implementations from different services make it quite hard to write universal Delphi code to access this. Then the REST APIs itself from the cloud services also have all their particularities. This makes it hard and time consuming to write and test all the code needed to access a cloud service from Delphi. With the TMS Cloud Pack, it is our goal to make your life easier and offer user friendly Delphi components that encapsulate all these particularities so you can focus on your application functionality rather than on all specific details related to cloud services. As the number of cloud services is still rising every day, it will be a matter of picking these services that interest you the most. We look forward to your feedback & comments on what cloud services you would like us to provide Delphi components for.

TMS Cloud Pack for Delphi XE, XE2, XE3

Single developer license	Site license
€ 50	€ 175

For more information visit:
http://www.tmssoftware.com/site/cloudpack.asp