TMS Cloud Pack
DEVELOPERS GUIDE

August 2012

Web: http://www.tmssoftware.com
Email: info@tmssoftware.com

Index

Availability

TMS Cloud Pack is a set of VCL components for Win32 & Win64 application development and is available for Embarcadero Delphi XE, XE2 and XE3.

Online references

TMS software website:
http://www.tmssoftware.com

TMS Cloud Pack page:
http://www.tmssoftware.com/site/cloudpack.asp

Purchase a license

The TMS Cloud Pack is separately available and also as part of the TMS VCL Subscription:

- TMS Cloud Pack : http://www.tmssoftware.com/site/panels.asp
- TMS VCL Subscription: http://www.tmssoftware.com/site/vdsub.asp

Terms of use

With the purchase of TMS Cloud Pack, you are entitled to our consulting and support services to integrate the Google GDrive service, the Microsoft SkyDrive or the DropBox service in Delphi applications and with this consulting and support comes the full source code needed to do this integration. As TMS Cloud Pack uses the Google Gdrive service, the Microsoft SkyDrive service and the DropBox service, you're bound to the terms of these services that can be found at:

http://www.google.com/apps/intl/en/terms/user_terms.html

http://windows.microsoft.com/en-US/windows-live/microsoft-service-agreement?SignedIn=1

http://windows.microsoft.com/en-AU/windows-live/code-of-conduct

https://www.dropbox.com/terms

TMS software is not responsible for the use of TMS Cloud Pack components. The purchase of TMS Cloud Pack does not include any license fee that you might possibly be required to pay to Google, Microsoft or DropBox. It will depend on your type of usage of these services whether a license fee needs to be paid to Google, Microsoft or DropBox.

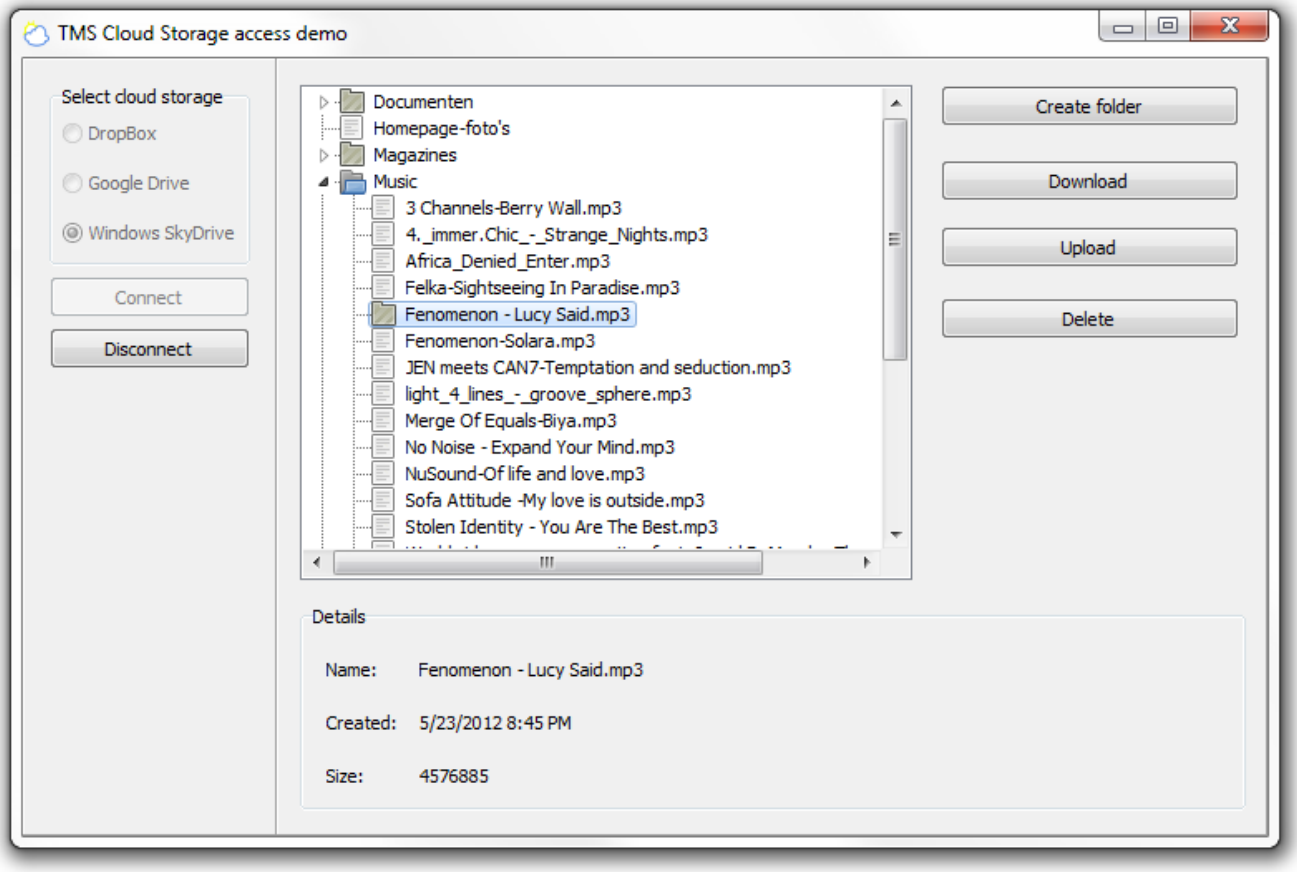
It is the sole responsibility of the user or company providing the application that integrates the Google, Microsoft or DropBox service to respect the Google, Microsoft and DropBox terms and conditions. TMS software does not take any responsibility nor indemnifies any party violating the Google, Microsoft or DropBox service terms & conditions.

Limited warranty

TMS software cannot guarantee the current or future operation & uptime of the Google Gdrive, Microsoft SkyDrive or DropBox service. TMS software offers the consulting and support for TMS Cloud Pack in good faith that the Google Drive, Microsoft SkyDrive and DropBox service is a reliable and future-proof service. In no case, TMS software shall offer refunds or any other compensations in case the Google Drive, Microsoft SkyDriven, DropBox service terms/operation changes or stops.

Main features

- Set of VCL components to offer easy access from Windows applications to cloud services
- Component to get access to DropBox storage
- Component to get access to Google Drive storage
- Component to get access to Microsoft SkyDrive storage
- Built-in support for OAuth 1.0 & 2.0 handling
- Built-in support for use of refresh tokens for use with one time authentication

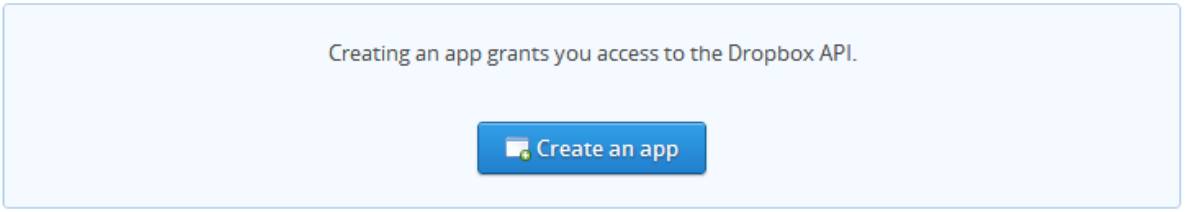


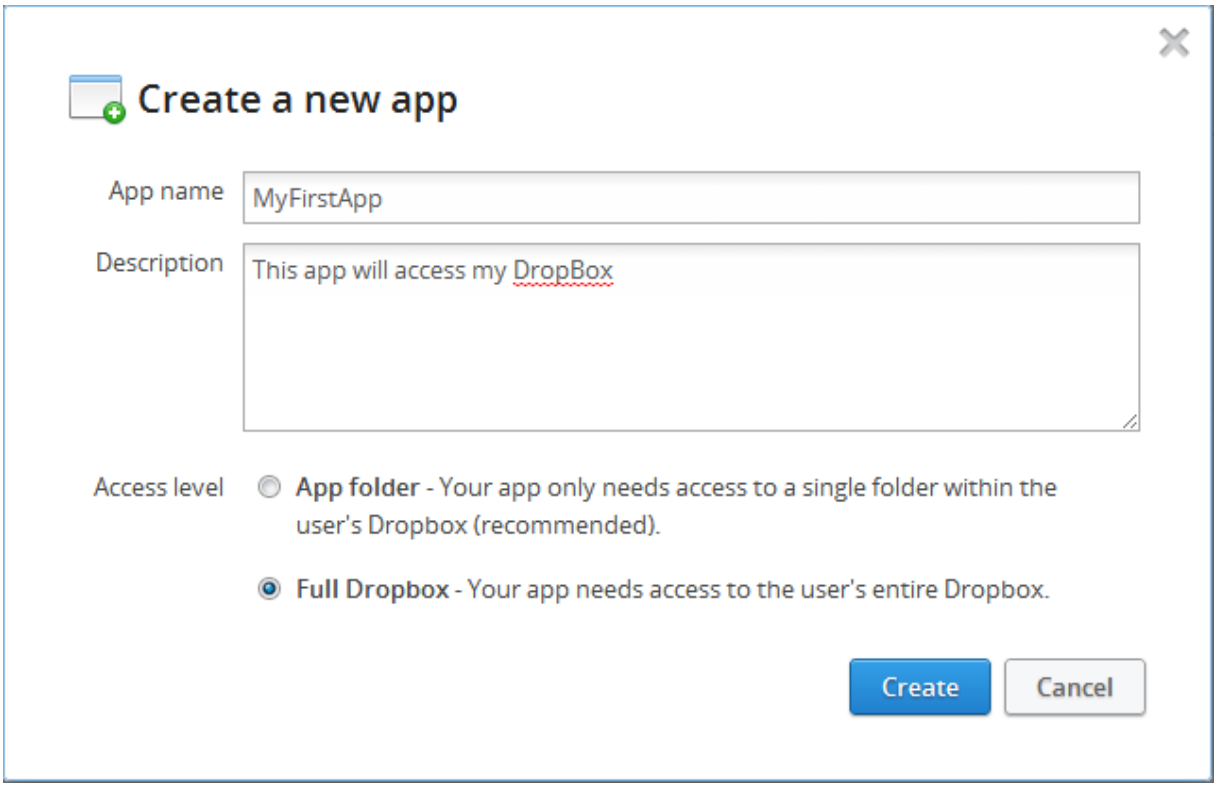
Registering your application

A first step will be to register your application so you can obtain an application key and secret at the different cloud services.

DropBox

To create an application for DropBox, you'll need a DropBox account. Login and request the application key and secret via the link: https://www.dropbox.com/developers/apps





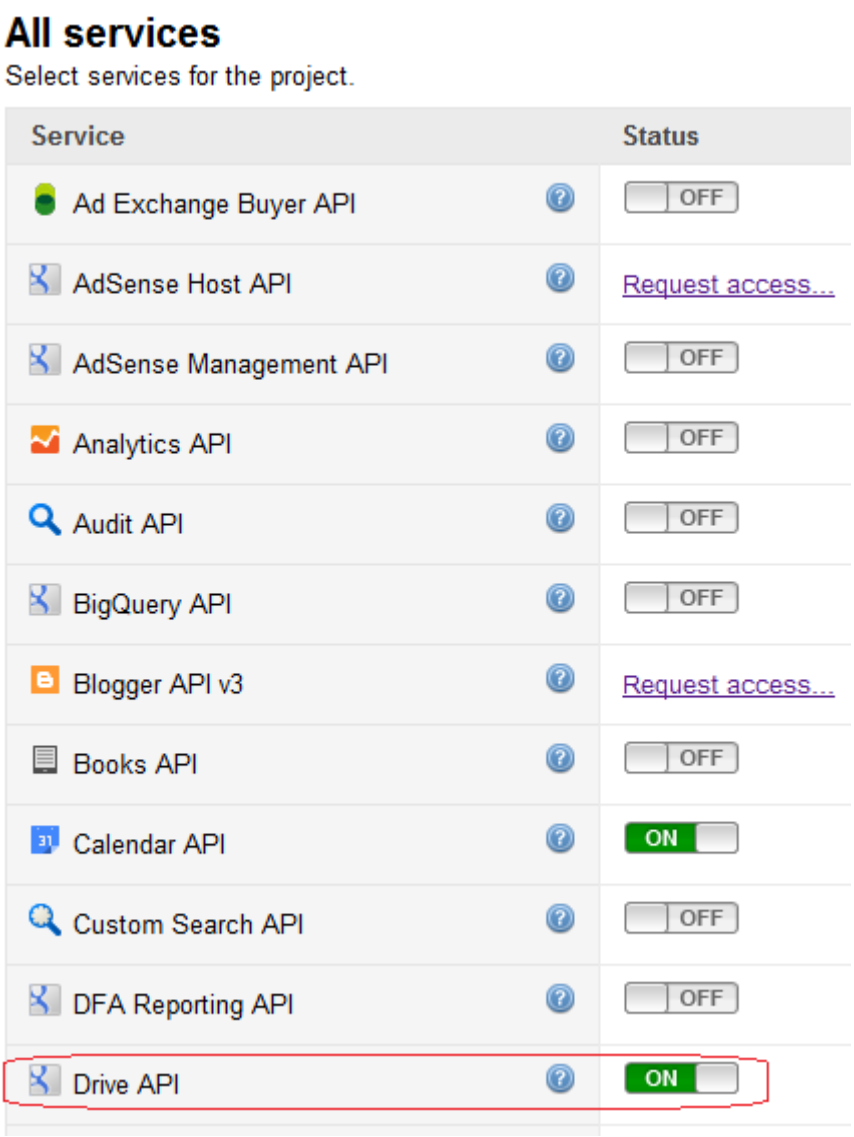
When confirming the creation of this app, you will see on the next page the application key and application secret. This is needed for using the TMS Cloud access components.

Google Drive

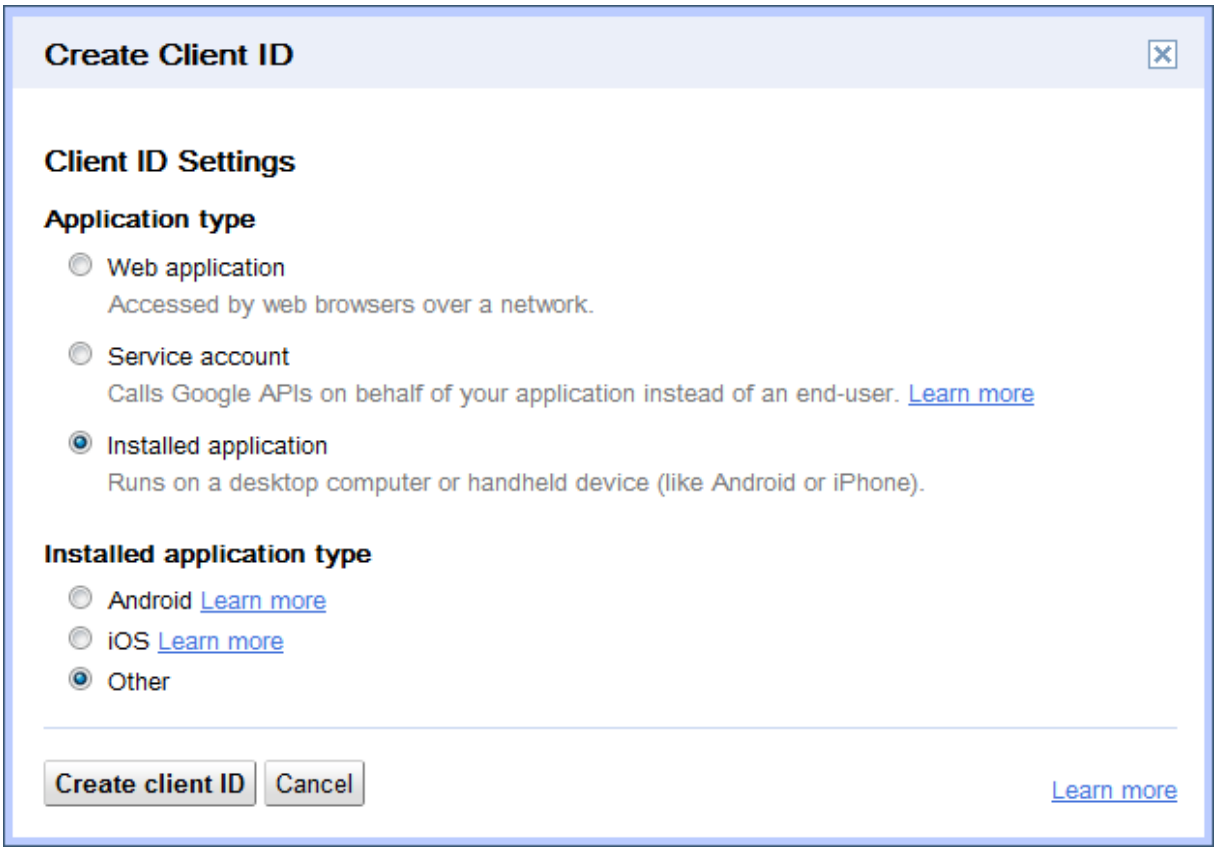
To create an application for Google's APIs, go to the API console:

https://code.google.com/apis/console/

and from the Console, choose “API Access” and “Create (another) client ID”. Make sure that under “Services”, the Drive API is turned on.



Make sure to select type Application type : ‘Installed application’ and type ‘other’:

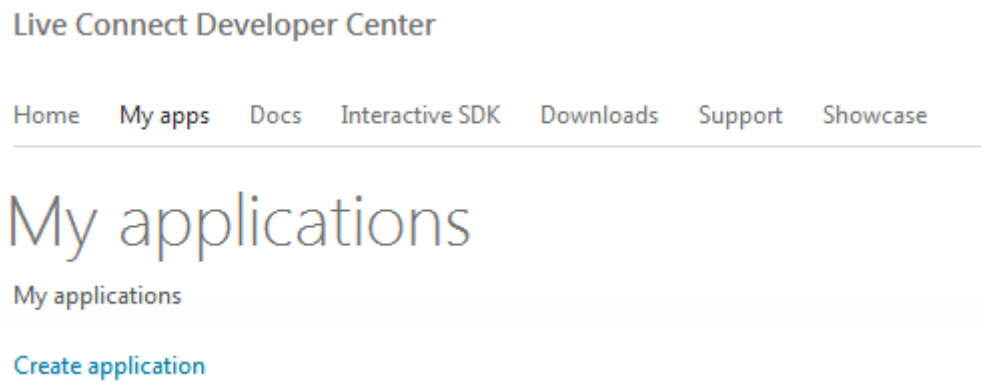


Microsoft SkyDrive

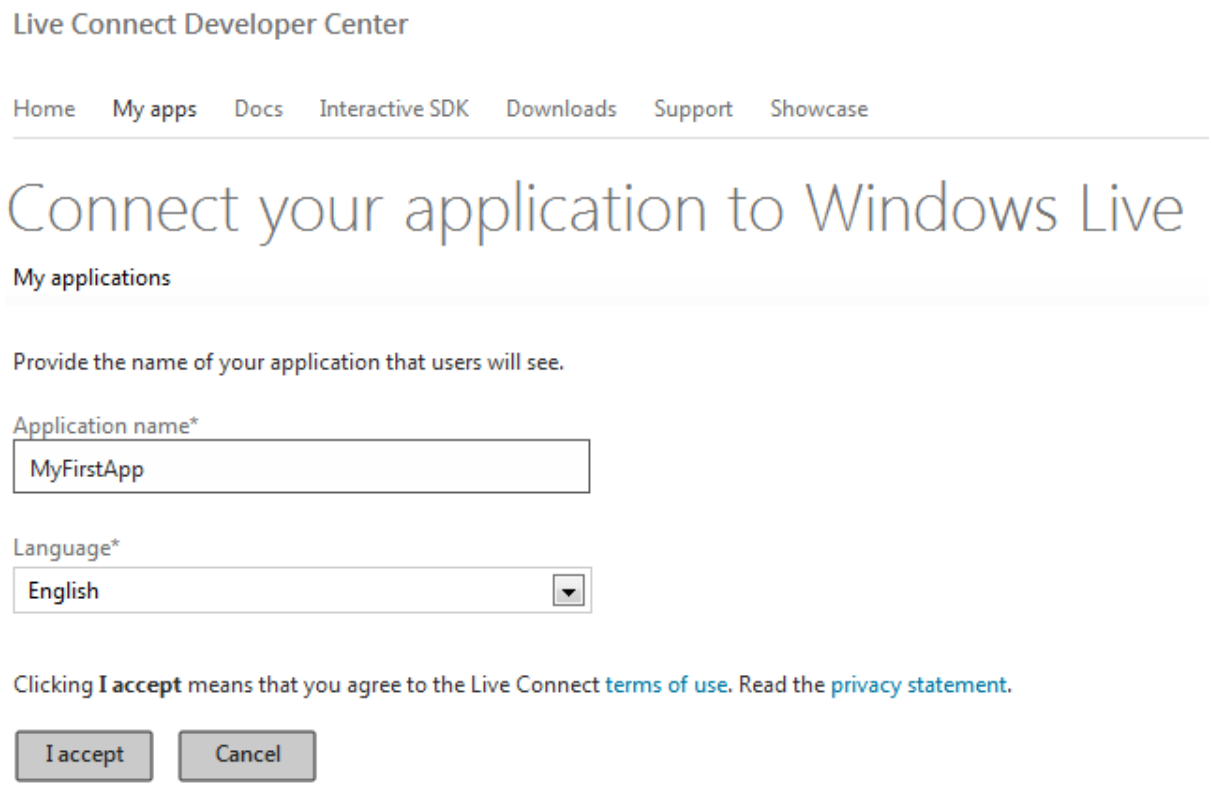
Login to Windows Live and go to the URL:

https://manage.dev.live.com/Applications/Index

and from this page select: “Create application”.



Then choose an application name and language for your application:



Accepting will show a page with your application client ID and client secret.

Getting started

Once your application is registered and you have an application ID or client ID and application secret or client secret, you can get started to use the TAdvSkyDrive, TAdvDropBox and TAdvGDrive components to access your cloud storage. All 3 components work in a similar way.

1) Drop the component on the form

2) Setup the client ID, client secret via the .App.Key and .App.Secret property

3) Call the .DoAuth method

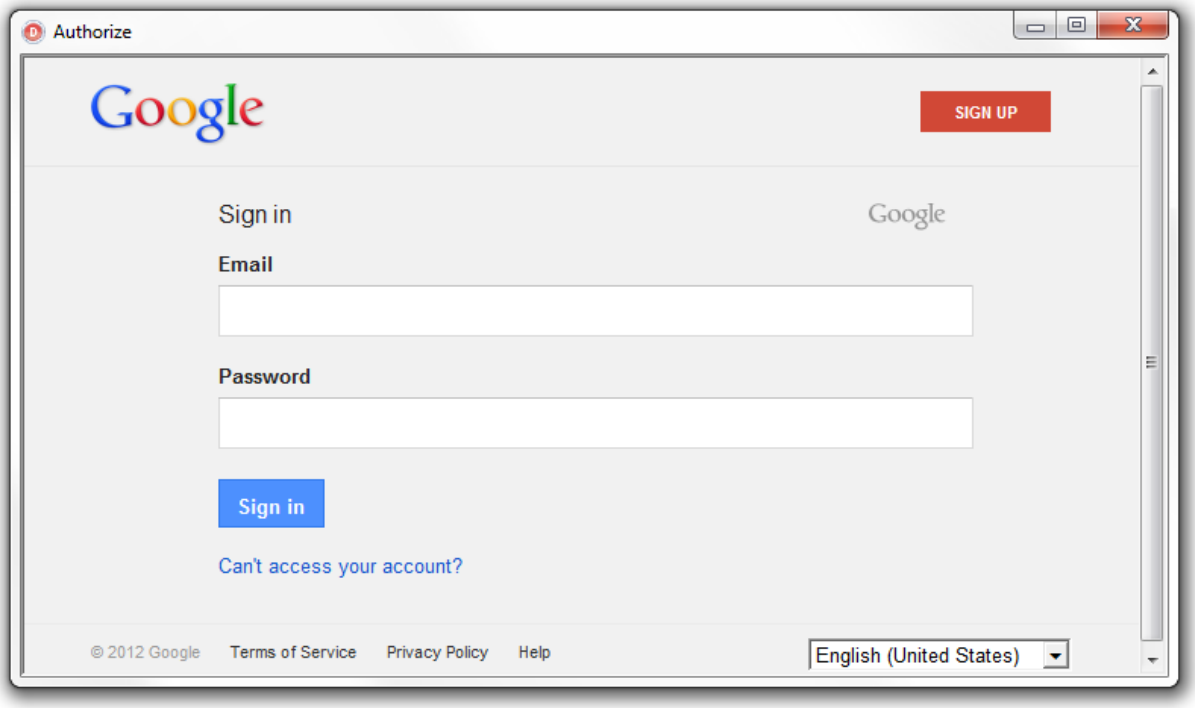
Code:

```
AdvGDrive1.App.Key := 'xxxxxxxxx.apps.googleusercontent.com';

AdvGDrive1.App.Secret := 'yyyyyyyyyyyyyyyy';

AdvGDrive1.DoAuth;
```

And this will show the Google login screen:



The size of the login screen and the caption are controlled by the properties:

```
AuthFormSettings.Caption: string;
AuthFormSettings.Height: integer;
AuthFormSettings.Width: integer;
```

When the login screen has been closed without a successful authentication, the event OnAuthFormClose will be triggered.
Alternatively, it is also possible to use any other TWebBrowser instance as login screen. To do this, just assign this TWebBrowser instance to TAdvXXXDrive.AuthBrowser: TWebBrowser.

When the user has provided the correct credentials, the event OnReceivedAccessToken will be triggered and from that moment, the component has access to the online cloud APIs.

File organisation

The file structure of a cloud storage service typically has a hierarchical organization in folders and files. This is represented in the cloud access component as a collection of the type TCloudItems. The common structure of this collection is:

TCloudItems = class(TCollection)

in this collection are items of the type:

TCloudItem = class(TCollectionItem)

with properties:

property FileName: string;
Holds the filename of a file or folder

property Folder: TCloudItems;
When the ItemType is itFolder, this contains in turn a collection of files and folders

property Size: int64;
Holds the size of the item

property ItemType: TCloudItemType;
Defines whether the item is a file or folder (itFile, itFolder)

property CreationDate: TDateTime;
Holds the creation date of the file or folder

property ModifiedDate: TDateTime;
Holds the timestamp when the file was last modified

property Tag: integer;
Integer property that can be freely used.

These are the common file/folder properties. Note that for different cloud storage services, there might be different extra properties available.

Calling the method TAdvXXXDrive.GetDriveInfo will use the cloud storage API to query the list of all files and will store this hierarchically in the Drive: TCloudItems collection property.

A helper method is available to immediately visualize this file structure in a treeview. This can be done by calling TAdvXXXDrive.FillTreeView(TreeView: TTreeView);

File operations

Following operations are available:

Create a folder

Delete a file or folder

Download a file

Upload a file

Creating a folder

Creating a folder is simple. You can either create a folder in the root by calling:

TAdvXXXDrive.CreateFolder(nil, 'New folder'): TCloudItem;

or create a subfolder in a specific folder. To do this, you need the instance of the TCloudItem representing the folder and use it as first parameter of the CreateFolder() call.

TAdvXXXDrive.CreateFolder(ParentFolderItem, 'New folder'): TCloudItem;

This function returns a new TCloudItem instance representing the created folder.

Delete a file or folder

Deleting a file can be done by calling the function TAdvXXXDrive.Delete(CloudItem): Boolean. This function returns true on a successful delete. The parameter is the TCloudItem instance that represents the file or folder to be deleted

Download a file

Downloading a file is equally simple. Call the function TAdvXXXDrive.Download(CloudItem, TargetFileName): Boolean. For a successful download, this function returns true. Note that the progress of the download can be tracked via the event OnDownloadProgress.

Upload a file

Uploading a file means creating the file on the cloud storage and uploading its data. The function that is used for upload is: TAdvXXXDrive.Upload(Folder: TCloudItem; FileName:string): TCloudItem.

The file is uploaded in the root (when Folder parameter is nil) or in the folder as specified by the Folder TCloudItem. The local file that will be uploaded is set via the FileName parameter. When successful, this function returns an instance of the new created file. Note that this item will also automatically be added in the TAdvXXXDrive.Drive collection. The progress during the upload can also be tracked via the OnUploadProgress event.

Authentication persistence

Internally, after authentication with the cloud service, the component has obtained an access token and for some services also a refresh token. This access token and refresh token can be used at a later time to access the cloud service again without the need for authentication. The components can:

Test if the access token is still accepted

Save the tokens in encrypted form in an INI file or registry key

Load the tokens from an INI file or registry key

The component has methods:

LoadTokens: load & decrypt access and refresh token from INI file or registry

SaveTokens: encrypt and save access and refresh tokens to INI file or registry

TestTokens: Boolean: performs a test if the access token is still accepted

RefreshAccess: Boolean: tries to get a new access token with the refresh token

and the property:

TokensAsString: string; encrypted string holding all token values

A typical flow is:

```
var
  acc: boolean;

if Storage.App.Key <> '' then
begin
  // load tokens from registry
  Storage.LoadTokens;
  // test if the access token is still accepted
  acc := Storage.TestTokens;
  // when not, try to get a new access token with the refresh token
  if not acc then
```

```
    acc := Storage.RefreshAccess;
  // when no new access token can be obtained, do new authentication
  if not acc then
    Storage.DoAuth
end
```

and the access token, refresh token is saved from the OnReceivedAccessToken event:

```
procedure TForm1.CloudAccess1ReceivedAccessToken(Sender: TObject);
var
  cs: TCloudStorage;
begin
  Authenticated := true;

  if (Sender is TCloudStorage) then
  begin
    cs := Sender as TCloudStorage;
    cs.SaveTokens;
  end;
end;
```

The property TokensAsString can be used to save and load the token values in a DB field for example.

Tips and FAQ

Logging

For debugging purposes, it can be interesting to see the log of all HTTPS access. Set CloudComponent.Logging = true and by default a log file will be generated under \My Documents. To override the default log file, set the filename with the public property CloudComponent.LogFileName: string;

Scopes

Many REST APIs implement Scopes as part of the authentication. The scopes specify what parts of the API the client needs access to and thus needs to authenticate for. In the TMS cloud storage acces components these scopes are automatically preset to perform full read/write access to the files.

For Microsoft SkyDrive, these scopes are:

wl.signin : consent to login on Live services

wl.basic : basic authentication
wl.offline_access : request a refresh token
wl.skydrive : read access to user SkyDrive files
wl.skydrive_update : write access to user SkyDrive files

For Google Drive the scopes are:

https://www.googleapis.com/auth/drive : read access
https://www.googleapis.com/auth/drive.file : full read/write access

If you want to limit access as read-only for example for the Windows SkyDrive or Google Drive, remove the specifiers wl.skydrive_update from the TAdvSkyDrive scopes or remove from https://www.googleapis.com/auth/drive.file from the TAdvGDrive scopes.