TMS LCL Cloud Pack
DEVELOPERS GUIDE

May 2016

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

Contents

Availability

TMS LCL Cloud Pack is a set of LCL components for Win32/Win64/Linux/MacOS application development and is available for Lazarus fpc.

Online references

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

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

Purchase a license

The TMS LCL Cloud Pack is available separately and is also part of the TMS Cloud Studio.

There is also a version of TMS Cloud Pack for FireMonkey, TMS Cloud Pack for VCL, IntraWeb and Visual Studio .NET & ASP.NET:

- TMS Cloud Pack for VCL: http://www.tmssoftware.com/site/cloudpack.asp
- TMS Cloud Pack for FireMonkey: http://www.tmssoftware.com/site/TMSLCLcloudpack.asp
- TMS IntraWeb Cloud Pack: http://www.tmssoftware.com/site/tmsiwcloud.asp
- TMS Cloud Pack for .NET: http://www.tmssoftware.com/site/tmscloudnet.asp

Terms of use

With the purchase of TMS LCL Cloud Pack, you are entitled to our consulting and support services to integrate the Amazon Cloud Drive, Google GDrive, Microsoft SkyDrive, DropBox, Box, Flickr, Google Calendar, Google Contacts, Google Picasa, Google Mail, Google Analytics, Microsoft Live Calendar, Microsoft Live Contacts, Wunderground weather, Google Location Lookup, Google Search Lookup, Google DataStore, Facebook, Twitter, LinkedIn, PushOver, Instagram, FourSquare, FreeGEOIp, YouTube, Pryv, CloudConvert, Barcodes4me, HiDrive service in Lazarus applications and with this consulting and support comes the full source code needed to do this integration. As TMS LCL Cloud Pack uses the Amazon Cloud Drive, Google GDrive, Microsoft OneDrive, DropBox, Box, Flickr, Google Calendar, Google Contacts, Google Picasa, Google DataStore, Microsoft Live Calendar, Microsoft Live Contacts, Wunderground weather, Google Location Lookup, Google Search Lookup, Facebook, Twitter, LinkedIn, PushOver, Instagram, FourSquare, FreeGEOIp, YouTube, Pryv, CloudConvert, HiDrive, Trello, Stripe 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
http://developers.facebook.com/policy/
http://twitter.com/tos
http://www.wunderground.com/members/tos.asp
https://m.box.com/static/html/terms.html
http://info.yahoo.com/legal/us/yahoo/utos/utos-173.html
http://developer.linkedin.com/documents/linkedin-apis-terms-use
https://pushover.net/terms
http://instagram.com/legal/terms/
https://foursquare.com/legal/terms
https://www.youtube.com/static?template=terms
http://pryv.com/terms-of-use/
http://www.amazon.com/gp/help/customer/display.html/?nodeId=201376540&ref_=cd_tou_fp
https://cloudconvert.com/terms
https://dev.strato.com/hidrive/terms-of-service
https://trello.com/legal
https://stripe.com/terms

TMS software is not responsible for the use of TMS LCL Cloud Pack components. The purchase of TMS LCL Cloud Pack does not include any license fee that you might possibly be required to pay to Amazon, Google, Microsoft, DropBox, Box, Flickr, Wunderground, Facebook, Twitter, LinkedIn, PushOver, Instagram, FourSquare, YouTube, Pryv, CloudConvert, Barcodes4me, HiDrive, Trello. It will depend on your type of usage of these services whether a license fee needs to be paid.

It is the sole responsibility of the user or company providing the application that integrates the Amazon, Google, Microsoft, DropBox, Box, Flickr, Facebook, Twitter, Wunderground, LinkedIn, PushOver, Instagram, FourSquare, YouTube, Pryv, CloudConvert, Barcodes4me service to respect the Google, Microsoft, DropBox, Facebook, Twitter, Wunderground, LinkedIn, PushOver, Instagram,

YouTube, Pryv, CloudConvert terms and conditions. TMS software does not take any responsibility nor indemnifies any party violating the Google, Microsoft, DropBox, Box, Flickr, Facebook, Twitter, Wunderground, LinkedIn, PushOver, Instagram, FourSquare, YouTube, Pryv, CloudConvert, Barcodes4me, HiDrive, Trello, Stripe service terms & conditions.

Limited warranty

TMS software cannot guarantee the current or future operation & uptime of the Amazon, Google Drive, Microsoft OneDrive, Microsoft Live Calendar, Microsoft Live Contacts, DropBox, Box, Flickr, Facebook, Twitter, Wunderground, Google Contacts, Google Calendar, Google Picasa, LinkedIn, PushOver, Instagram, FreeGEOIP, YouTube, Pryv, CloudConvert, Barcodes4me, HiDrive, Trello services.
TMS software offers the consulting and support for TMS LCL Cloud Pack in good faith that the Amazon, Google Drive, Google DataStore, Microsoft OneDrive, Microsoft Live Calendar, Microsoft Live Contacts, DropBox, Box, Flickr, Facebook, Twitter, Wunderground, Google Contacts, Google Calendar, Google Picasa, Google Mail, Google Analytics, LinkedIn, PushOver, Instagram, FreeGEOIP, YouTube, Pryv, CloudConvert, Barcodes4me, HiDrive, Trello service is a reliable and future-proof service.
In no case, TMS software shall offer refunds or any other compensation in case the Amazon, Google Drive, Microsoft OneDrive, Microsoft Live Calendar, Microsoft Live Contacts, DropBox, Box, Flickr, Facebook, Twitter, Wunderground, Google Contacts, Google Calendar, Google Picasa, LinkedIn, PushOver, Instagram, FreeGEOIP, YouTube, Pryv, CloudConvert, Barcodes4me, HiDrive, Trello, Stripe service terms/operation changes or stops.

Main features

- Set of LCL components to offer easy access from Windows applications to cloud services
- Component to get access to Amazon cloud drive
- Component to get access to DropBox storage
- Component to get access to Box storage
- Component to get access to HiDrive storage
- Component to get access to CloudConvert API
- Component to get access to Google Drive storage
- Component to get access to Microsoft SkyDrive storage
- Component to get access to Facebook API
- Component to get access to Twitter API
- Component to get access to Google Contacts API
- Component to get access to Google Calendar API
- Component to get access to Google Picasa API
- Component to get access to Google Places API
- Component to get access to Google Mail API
- Component to get access to Google Analytics API
- Component to get access to Flickr API
- Component to get access to Windows Live Contacts API
- Component to get access to Windows Live Calendar API
- Component to get access to Wunderground weather forecast service
- Component to get access to LinkedIn API
- Component to get access to Instagram API
- Component to get access to FourSquare API
- Component to get access to YouTube API
- Component to get access to DropBox DataStore API
- Component to get access to Pryv API
- Component to get access to Barcodes4me API
- Component to get access to Trello API
- Component to get access to Stripe API
- Component to shorten URLs based on Google URL shortener service
- Component to send push messages to iOS devices running the PushOver client
- Component to determine machine location via the FreeGEOIP service
- Edit control with lookup capabilities based on web services lookup data
- Built-in support for OAuth 1.0 & 2.0 handling
- Built-in support for use of refresh tokens for use with one time authentication

Installing LCLTMSCloudLazarusPkg.lpk package

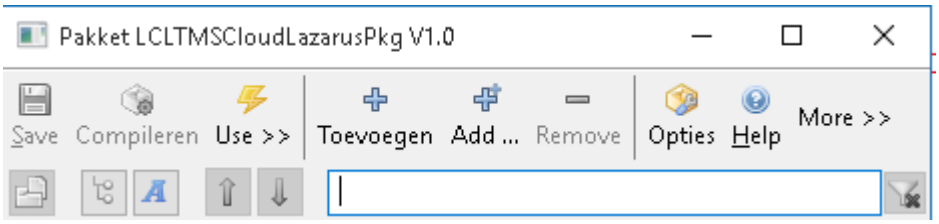
Manual installation

Trial Version:

This version includes a closed source package. It will be impossible to compile your package.

Click “Use >>”

Click “Install”

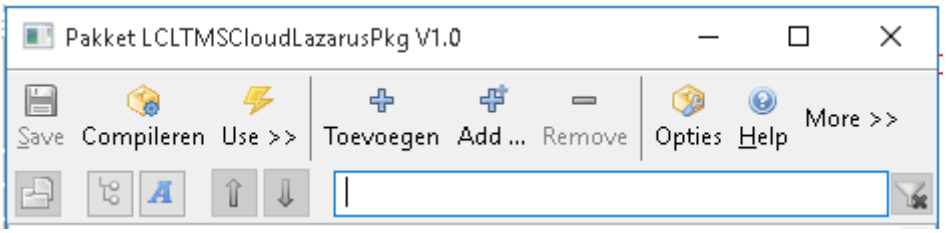


Registered Version:

“Compile” your package before use

Click “Use >>”

Click “Install”



Automatic installation

Please find the script TMSInstall in the directory.

Windows
- Open the terminal
  Windows + R & type cmd
- navigate to the TMS LCL Cloud Pack directory
  cd /path/to/directory
- type *TMSLCLCompile_old.bat LCLTMSCloudLazarusPkg PathToDirectory PathToLazarus*
ex: *TMSLCLCompile_old.bat LCLTMSCloudLazarusPkg "C:\Temp\TMS LCL Cloud Pack" "C:\lazarus"*

Raspberry Pi 2
- open the terminal (Menu > Accessories > Terminal)
- navigate to the TMS LCL Cloud Pack directory (*cd /path/to/directory*)
- type *sudo bash ./TMSInstall.sh PackageName "path/to/directory" "path/to/lazarus"*
ex: *sudo bash ./TMSInstall.sh LCLTMSCloudLazarusPkg "/home/pi/Documents/TMS LCL Cloud Pack" /usr/local/lib/lazarus/1.5/*

Extra package for Raspberry Pi 2

Raspberry Pi 2 has no openssl lib installed. Please follow these instructions:

- open the terminal (Menu > Accessories > Terminal)

- Extract the compressed package "openssl-0.9.8zg.tar.gz"

sudo tar zxvf openssl-0.9.8/

- Go inside the extracted folder

cd openssl-0.9.8zg

- type following command:

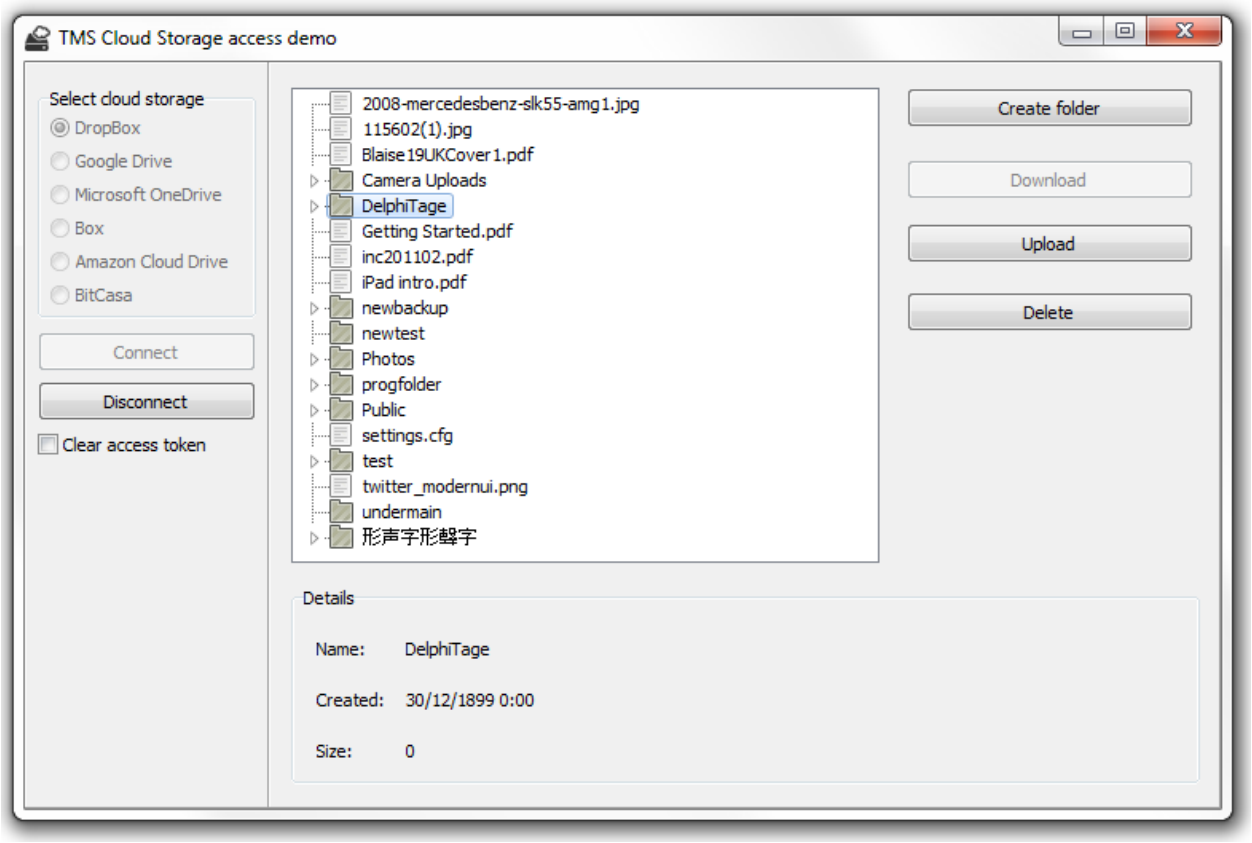
sudo make all

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.

Please refer to our online documentation.

Getting started with cloud storage access



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 TTMSLCLCloudSkyDrive/TTMSLCLCloudOneDrive, TTMSLCLCloudDropBox, TAmazonCloudDrive, TTMSLCLCloudBoxNetDrive, TTMSLCLCloudGDrive, TTMSLCLCloudHubic and TTMSLCLCloudHiDrive components to access your cloud storage. All storage 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.

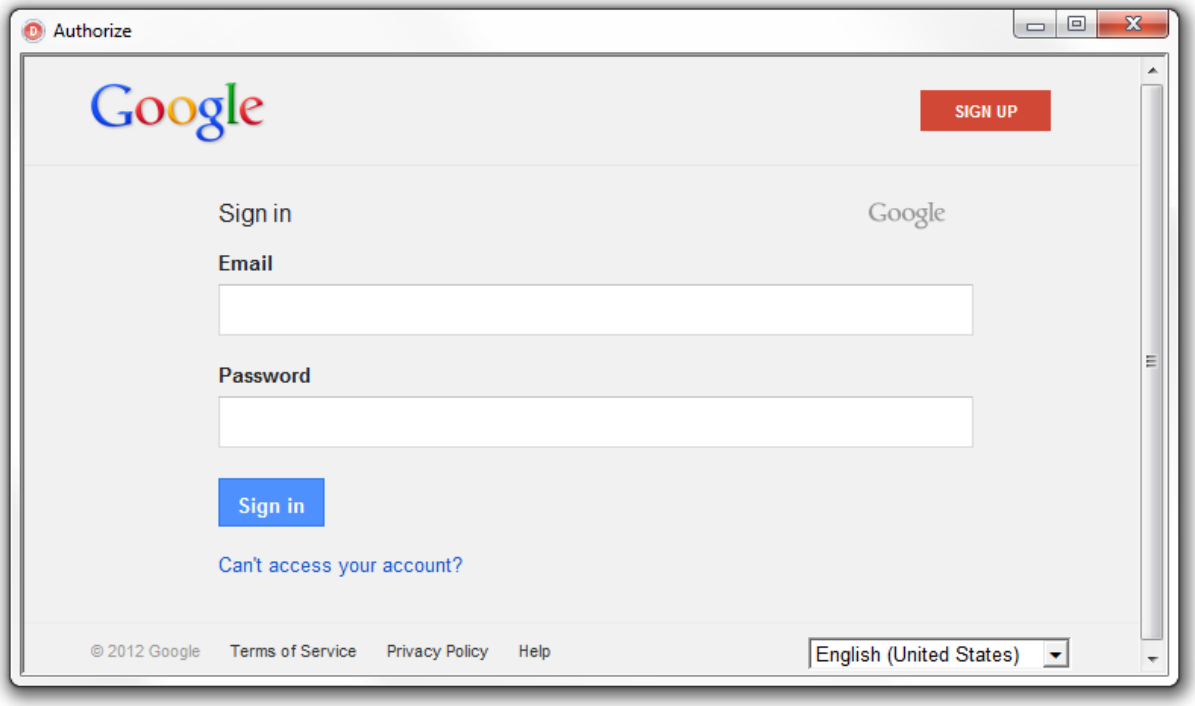
Code:

```
TMSLCLCloudGDrive1.App.Key := 'xxxxxxxxx.apps.googleusercontent.com';

TMSLCLCloudGDrive1.App.Secret := 'yyyyyyyyyyyyyyyy';
```

```
TMSLCLCloudGDrive1.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 TTMSLCLCloudXXXDrive.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 TTMSLCLCloudXXXDrive.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 TTMSLCLCloudXXXDrive.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:

TTMSLCLCloudXXXDrive.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.

TTMSLCLCloudXXXDrive.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 TTMSLCLCloudXXXDrive.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 TTMSLCLCloudXXXDrive.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: TTMSLCLCloudXXXDrive.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 TTMSLCLCloudXXXDrive.Drive collection. The progress during the upload can also be tracked via the OnUploadProgress event.

Public shared files

TTMSLCLCloudDropBox, TTMSLCLCloudSkyDrive/TTMSLCLCloudOneDrive, TTMSLCLCloudBoxNetDrive, TTMSLCLCloudGDrive, TTMSLCLCloudHubic and TTMSLCLCloudHiDrive have a built-in function to create a public share URL for a file on the cloud storage.

To create & get the share URL, you can use either:

property TCloudItem.Share: string

Property returning a public accessible HTTP URL to download the file.

or

function TCloudStorage.GetShare(CloudItem): string

With the property

property TCloudItem.Shared: Boolean

it can be retrieved whether the file is shared or not.

Notes:

- **TTMSLCLCLOUDGDrive:**
  For TTMSLCLCloudGDrive, the permissions are organized in a slightly different way:

  TGDriveItem.PublicShare := true;
  Set this item as public accessible.

  TGDriveItem.PublicShare := false;
  Remove the public accessible permission for this item.

  The property **TGDriveItem.Shared: boolean** will return true when the file can be shared with other users.
  The property **TGDriveItem.DownloadURL: string** can return the HTTP URL for such shared file.
  The property **TGDriveItem.AlternateURL: string** can return a link for opening the file using a relevant Google editor or viewer.

The property **TGDriveItem.WebContentURL: string** can return a link for downloading the content of the file. In cases where the content is shared publicly, the content can be downloaded without credentials.

- **TTMSLCLCloudHiDrive:**
  For TTMSLCLCloudHiDrive, shared URLs can expire after the shared file has been downloaded a specific number of times or after a specific amount of time. The number of times a shared file can be downloaded or the amount of time before the shared link expires depends on the HiDrive account type of the authenticated user.
  To re-enable an expired share, first delete the existing share and then create a new share for the same THiDriveItem.

  function THiDriveItem.DeleteShare: Boolean

  or

  function TTMSLCLCloudHiDrive.DeleteShare(ShareURL: string): Boolean

  Function to delete an existing share. The share will be deleted if it is expired or not.

CloudStorage specific settings

TTMSLCLCloudDropBox has a few extra settings to take control of specific behavior of the DropBox service. The public property ExternalBrowser: Boolean is added. When setting this to true, the authentication/authorization step can be done via the default browser.
Secondly, a property TMSLCLCloudDropBox.FileLimit: integer is added. This controls how many files the DropBox API can return for a folder file listing. The default value is 10000. If you have more than 10000 files in a DropBox folder, this can be increased to a maximum value of 25000. The maximum value of 25000 is a limitation of the DropBox API itself.

Note:
The TTMSLCLCloudHubic component is derived from the TTMSLCLCloudCustomOpenStack class which partially implements the OpenStack Object Storage API v1.

Information about the OpenStack Object Storage API can be found here:
http://developer.openstack.org/api-ref-objectstorage-v1.html

TTMSLCLCloudTwitter

Usage

TTMSLCLCloudTwitter is a component that provides access to the Twitter API service. It allows to Tweet messages with an optional image, Retweet messages from other users and retrieve existing messages, followers & friends from a Twitter account.

Organisation

Properties:

Profile: TTwitterProfile; Class property that contains the Twitter profile information for the currently authenticated user.

ScreenName: string; The screen name, handle, or alias that this user identifies themselves with.
ID: integer; The unique identifier for this user.
CreatedAt: TDateTime; The UTC datetime that the user account was created on Twitter.
ListedCount: integer; The number of public lists that this user is a member of.
Name: The name of the user, as they've defined it.
StatusCount: integer; The number of tweets this user has set as favorite in the account's lifetime.
FollowersCount: integer; The number of followers this account currently has.
FriendsCount: integer; The number of users this account is following.
GeoEnabled: Boolean; When true, indicates that the user has enabled the possibility of geotagging their Tweets.
ImageUrl: string; An URL pointing to the user's avatar image.
Protected: Boolean; When true, indicates that this user has chosen to protect their Tweets.
Status: The user's most recent tweet or retweet.
StatusCount: integer; The number of tweets (including retweets) issued by the user.
TimeZone: string; A string describing the timezone this user declares themselves within.
URL: string; A URL provided by the user in association with their profile.

Followers: TProfileList; Contains the list of Twitter followers.
Friends: TProfileList; Contains a list of Twitter friends.
Statuses: TStatusList; Contains a list of Twitter Status messages.

User: TTwitterProfile; The user who posted this Tweet.
Text: string; The actual text of the status update.
Favorited: Boolean; Indicates whether this Tweet has been set as a favorite by the user.
CreatedAt: TDateTime; Time when this Tweet was created.
ID: uint64; The unique identifier for this Tweet.
InReplyToID: integer; If the represented Tweet is a reply, this field will contain the original

Tweet's author ID.
InReplyToScreenName: string; If the represented Tweet is a reply, this field will contain the screen name of the original Tweet's author.
InReplyToStatusID: integer; If the represented Tweet is a reply, this field will contain the original Tweet's ID.
MediaURL: string; Link to the image that is connected to this Tweet (if any).
Source: string; Utility used to post the Tweet, as an HTML-formatted string.

Mentions: TStatusList; Contains a list of Twitter mentions.
UserTimeLine: TStatusList; Contains a list of Twitter messages that are on the user's timeline.

Methods:

Tweet(const Msg: string): string;
Create a new Tweet on the user's timeline.

Example:

```
TMSLCLCloudTwitter1.Tweet('Hello World!');
```

TweetWithMedia(const Msg, FileName: string): string;
Create a new Tweet with included image on the user's timeline.

Retweet(const ID: Int64): string;
Retweet an existing Tweet on the user's timeline.

DirectMessage(const Msg: string; ScreenName: string): string;
Send a direct message to another Twitter user.

GetFollowers: integer;
Fill the list of Followers.

GetFriends(UserID: string = ''): integer;
Fill the list of Friends.

GetStatuses(Count: integer = 100; SinceID: Int64 = -1; MaxID: Int64 = -1; UserID: integer = -1; UserName: string = ''): integer;
Fill the list of Statuses. If a UserID or UserName is provided a list of Statuses from the specific user is retrieved.

GetMentions(Count: integer = 100; SinceID: Int64 = -1; MaxID: Int64 = -1): integer;
Fill the list of Mentions.

Search(const Query: string; Count: integer = 100; SinceID: integer = -1): TStatusList;
Return a list of Status messages that contain the Query string value.

GetAccountInfo: boolean;
Fill the user's Profile.

GetProfileInfo(const ID: integer; Profile: TTwitterProfile): boolean; overload;
GetProfileInfo(const ID: string; Profile: TTwitterProfile): boolean; overload;
GetProfileInfo(Profile: TTwitterProfile): boolean; overload;
Fill a TTwitterProfile based on the profile ID.

GetProfileListInfo(ProfileList: TProfileList): boolean;
Fill a list TTwitterProfiles based on the profile IDs.

Example:

```
TMSLCLCloudTwitter1.GetFollowers;
TMSLCLCloudTwitter1.GetProfileListInfo(TMSLCLCloudTwitter1.Followers);
```

TTMSLCLCloudFacebook

Usage

TTMSLCLCloudFacebook is a component that provides access to the Facebook API service. It allows to post a status update (with optional URL and image), upload an image, retrieve a list of friends, retrieve a user's Feed and Like/Unlike a feed item.

Organisation

Properties:

APIVersion: TFacebookAPIVersion; Indicates which version of the Facebook API should be used (av10 for v1.0, av21 for v2.1, or av21 for v2.4).

Profile: Class property that contains the Facebook profile info for the currently authenticated user.

ID: string; The user's Facebook ID.
FullName: string; The user's full name.
FirstName: string; The user's first name.
LastName: string; The user's last name.
MiddleName: string; The user's middle name.
Gender: TFacebookGender; The user's gender.
Link: string; The URL of the profile for the user on Facebook.
UserName: string; The user's Facebook username.
BirthDay: string; The user's birthday.
Email: string; The proxied or contact email address granted by the user.
TimeZone: integer; The user's timezone.
Locale: string; The user's locale.
Verified: Boolean; The user's account verification status.
UpdatedTime: TDateTime; The last time the user's profile was updated.
ImageURL: string; The URL of the user's profile pic.
HomeTown: TFacebookObject; The user's hometown.
Location: TFacebookObject; The user's current city.
RelationShip: string; The user's relationship status.
SignificantOther: TFacebookProfile; The user's significant other.
Website: string; The URL of the user's personal website.
Likes: TFacebookObjectList; List of the Facebook pages the user has liked.

Feed: TFeed; List of the user's feed items.

ID: string; The post ID.
User: TFacebookProfile; Information about the user who posted the message.
Text: string; The actual message text.

ImageURL: string; If available, a link to the picture included with this post.
Link: string; The link attached to this post.
Summary: string; The name of the link.
Caption: string; The caption of the link.
Description: string; A description of the link.
CreatedTime: TDateTime; The time the post was initially published.
Story: string; Text of stories not intentionally generated by users.
ObjectID: string; The Facebook object id for an uploaded photo or video.
UpdatedTime: TDateTime; The time of the last comment on this post.
Likes: TFacebookProflieList; List of users that liked this post.
ToUsers: TFacebookProfileList; List of profiles mentioned or targeted in this post.
Comments: TFacebookCommentList; List of comments add to this post.

ID: string; The ID of the comment.
User: TFacebookProfile; The user that created the comment.
Text: string; The comment text.
CreatedTime: TDateTime; The time the comment was created.
Likes: TFacebookProfileList; List of users that liked this comment.
UserLikes: boolean; Indicates if the currently authenticated user has liked the comment.

Albums: TFacebookAlbumList; List of the user's album items.

ID: string; The ID of the album.
Title: string; The title of the album.
From: TFacebookProfile; The profile that created this album.
Link: string; A link to this album on Facebook.
Description: string; The description of the album.
CoverPhotoID: string; The album cover photo ID.
CreatedTime: TDateTime; The time the photo album was initially created.
UpdatedTime: TDateTime; The last time the photo album was updated.

Pictures: TFacebookPictureList; List of pictures that are in this album.

ID: string; The ID of the picture.
Summary: string; The user provided caption given to this picture.
From: TFacebookProfile; The profile (user or page) that posted this picture.
ImageURL: string; A direct URL link to the picture on Facebook.
Link: string; A link to the picture on Facebook.
CreatedTime: TDateTime; The time the picture was initially published.
UpdatedTime: TDateTime; The last time the picture or its caption was updated.

FriendList: TFacebookProfileList; Contains a list friend profiles.

FriendsCount: Integer; The total count of friends

PageList: TFacebookPageList; Contains a list of Facebook pages the currently authenticated user has administrative rights to.

ID: string; The page Facebook ID.
AccessToken: string; The access token that must be used to post to this page.
Summary: string; The page name.
Category: string; The page category.
Permissions: TStringList; The list of permissions that are available for the currently authenticated user.
Likes: integer; The number of likes the Page has received. (See GetLikes)

Methods:

GetUserInfo;
Fill the user's "Profile".

GetFriends;
Fill the "FriendList" with the user's Facebook Friends.

GetProfileInfo(const ID: string; Profile: TFacebookProfile): boolean;
Fill a TFacebookProfile based on the profile ID.

GetLikes(Profile: TFacebookProfile): boolean; overload;
Fill a TFacebookProfile.Likes list with the Facebook Pages that the user has "liked".

GetLikes(FeedItem: TFacebookFeedItem): boolean; overload;
Fill a TFacebookFeedItem.Likes list with TFacebookProfiles that have "liked" the TFacebookFeedItem.

GetLikes(Comment: TFacebookComment): boolean; overload;
Fill a TFacebookComment.Likes list with TFacebookProfiles that have "liked" the TFacebookComment.

GetLikes(Page: TFacebookPage): boolean; overload;
Set a TFacebookPage.Likes property with the total number of likes the Page has received. (requires the "read_insights" scope)

GetAlbums(Profile: TFacebookProfile): boolean;
Fill a TFacebookProfile.Albums list with the Facebook Photo Albums connected to the TFacebookProfile.

GetAlbums(ID: string): boolean;
Fill a TFacebookProfile.Albums list with the Facebook Photo Albums connected to the provided ID. The ID can be of a Facebook Profile or a Facebook Page.

GetPages(): boolean;
Fill "PageList" with the Facebook Pages the user has administrative access to.

GetPictures(Album: TFacebookAlbum): boolean;
Fill a TFacebookAlbum.Pictures list with the Pictures connected to the TFacebookAlbum.

GetFeed(Profile: TFacebookProfile; Count: integer = 100; Offset: integer = 0;Since: TDateTime = 0): integer;
Fill a TFacebookProfile.Feed with a list of Facebook Feed messages for the TFacebookProfile.

GetComments(FeedItem: TFacebookFeedItem): boolean;
Fill a TFacebookFeedItem.Comments with a list of comments for the TFacebookFeedItem.

GetImageURL(const ImageID: string): string;
Return a direct URL to a TFacebookPicture.ImageID.

Post(const Msg, Link, Image: string): string;
Post a message on the user's Facebook Feed with an optional URL link and image URL.

Post(const Msg, Link, Image: string; Page: TFacebookPage): string;
Post a message on the user's Facebook Page with an optional URL link and image URL.

PostComment(ID: string;const Msg: string): string;
Post a comment on one of the user's Facebook feed items based on a TFacebookFeedItem ID.

PostImage(const Msg, FileName: string): string; overload;
Post an image to the Facebook Album that is connected to your application registered with Facebook. (If the Facebook Album does not exist, it will automatically be created.)

PostImage(const Msg, FileName: string; Album: TFacebookAlbum): string; overload;
Post an image to the TFacebookAlbum.

PostImage(const Msg, FileName: string; Album: TFacebookAlbum; Page: TFacebookPage): string; overload;
Post an image to the TFacebookAlbum of a specific Facebook Page.

Like(ID: string): string;
Like a Facebook Feed message/comment based on a TFacebookFeedItem / TFacebookComment ID.

UnLike(ID: string): string;
Unlike a Facebook Feed message/comment based on a TFacebookFeedItem / TFacebookComment ID.

Example:

Post a message that contains an image from a Facebook Album to the user's feed:
First upload an image to a Facebook album then retrieve the direct remote URL of the image and use this to post a new message.

```
ImageID := TMSLCLCloudFacebook1.PostImage('Description',
OpenDialog1.FileName);
ImageURL := TMSLCLCloudFacebook1.GetImageURL(ImageID);
TMSLCLCloudFacebook1.Post('Message', 'http://www.mywebsite.com', ImageURL);
```

TTMSLCLCloudFlickr

Usage

TTMSLCLCloudFlickr is a component that facilitates access to the Flickr service. It allows retrieving your galleries, sets, and photostream as well as creating / deleting sets, uploading / downloading photos to sets and to your photostream. Photos can be uploaded / updated with a title / description and a geo location. Comments on sets and photos can also be retrieved.

Organisation

Properties:

property UserID: String: The ID retrieved after login, used to access user related content.
property UserName: String The user name retrieved after login.
property Galleries: TFlickrGalleries: The collection of galleries, loaded after calling GetGalleries.
property PhotoStream: TFlickrPhotos: The collection of photos from the photostream, loaded after calling GetPhotoStream.
property Sets: TFlickrSets: The collection of sets, loaded after calling GetSets.

Gallery Properties:
property ID: String: The ID of the set, used to access the flickr api for galleries specifically.
property URL: String: The direct URL of the gallery, to display in a browser window.
property Owner: String: The User ID of the owner of this gallery.
property DateCreate: String: The date the gallery is created.
property Title: String: The title of the gallery.
property Description: String: The description of the gallery.
property Photos: TFlickrPhotos: The collection of photos inside the gallery.

Set Properties:
property ID: String: The ID of the set, used to access the flickr api for sets specifically.
property Primary: String: The ID of the primary photo inside the set.
property Title: String: The title of the set.
property Description: String: The description of the set.
property Photos: TFlickrPhotos: The collection of photos inside the set.
property Comments: TFlickrComments: The comments of the set.

Photo Properties:
property ID: String: The ID of the photo, used to access the flickr api for photos specifically.
property Owner: String: The User ID of the owner of the photo.

property Title: String: The title of the photo
property Description: String: The description of the photo.
property Sizes: TFlickrSizes: The various sizes in which the photo exists and can be downloaded.
property Latitude: Double: The latitude of the geo location of the photo.
property Longitude: Double: The longitude of the geo location of the photo.
property Tags: TFlickrTags: The tags of the photo.
property Comments: TFlickrComments: The comments of the photo.

Tag Properties:

property ID: String: The ID of the tag.
property Author: String: The User ID of the author of the tag.
property Value: String: The value of the tag.

Comment Properties:

property ID: String: The ID of the comment.
property Author: String: The User ID of the author of the comment.
property AuthorName: String: The User Name of the author of the comment.
property Value: String: The value of the comment.
property DateCreate: String: The date the comment was created.

Size Properties:

property Size: String: Identifier for the size in which the photo can be downloaded.
property Width: Integer: The width of the photo for the specific size.
property Height: Integer: The height of the photo for the specific size.
property URL: String: The URL of the photo for the specific size to display in a browser window.
property DownloadURL: The download URL of the photo.

Methods:

```
function AddFolderToSet(AFolder, ATitle: String; ADescription: String): TFlickrSet;
```

Adds a folder of images to a new or existing set with a specific title and description.

```
Sample:
TMSLCLCloudFlickr1.AddFolderToSet('C:\folder\*.jpg', 'new set', 'set
description');
```

```
function AddPhotoToSet(AFileName, ATitle, ADescription: String):
TFlickrSet;
```

Adds an image to a new or existing set with a specific title and description.

```
Sample:
TMSLCLCloudFlickr1.AddPhotoToSet('C:\folder\image1.jpg', 'new set', 'set
description');
```

```
function CreateGallery(ATitle: String; ADescription: String):
TFlickrGallery;
```
Creates and returns a new gallery with a specific title and description.

```
Sample:
TMSLCLCloudFlickr1.CreateGallery('new gallery', 'gallery description');
```

```
function CreateSet(ATitle, ADescription, APrimaryPhotoID: String):
TFlickrSet;
```

Creates and returns a new set with a specific title, description and primary photo ID. The Primary Photo ID is required when creating a new set and can be retrieved by first uploading a photo to the photo stream.

```
Sample:
var
  APhoto: TFlickrPhoto;
begin
  APhoto := TMSLCLCloudFlickr1.UploadPhotoToStream('C:\folder\image1.jpg',
'new image', 'image description');
  if Assigned(APhoto) then
    TMSLCLCloudFlickr1.CreateSet('new set', 'set description', APhoto.ID);
```

```
function DownloadPhoto(ATargetFile, APhotoURL: String): Boolean; overload;
```
Downloads a photo to a target location. The APhotoURL is retrieved after calling GetSizes on a photo object and selecting the DownloadURL property depending on the size in the sizes collection.

```
function FindGalleryByID(AGalleryID: String): TFlickrGallery;
```
Returns a gallery by ID.

```
function FindGalleryByTitle(AGalleryTitle: String): TFlickrGallery;
```
Returns a gallery by title.

```
function FindGalleryByDescription(AGalleryDescription: String):
TFlickrGallery;
```
Returns a gallery by description.

```
function FindSetByID(ASetID: String): TFlickrSet;
```
Returns a set by ID.

```
function FindSetByTitle(ASetTitle: String): TFlickrSet;
```
Returns a set by title.

```
function FindSetByDescription(ASetDescription: String): TFlickrSet;
```
Returns a set by description.

```
function FindPhotoByID(APhotoID: String): TFlickrPhoto;
```
Returns a photo by id.

```
function FindPhotoInGalleryByID(APhotoID: String): TFlickrPhoto;
```
Returns a photo by id in the galleries collection.

```
function FindPhotoInSetByID(APhotoID: String): TFlickrPhoto;
```
Returns a photo by id in the sets collection.

```
function FindPhotoInStreamByID(APhotoID: String): TFlickrPhoto;
```
Returns a photo by id in the streams collection.

```
function GetAccountInfo: Boolean;
```
Returns a Boolean if the call has completed correctly and sets the User ID and User Name properties necessary for API access such as loading the sets / galleries. This function needs to be called after the authorization is complete to make sure subsequent calls function correctly.

```
function GetGalleries(AUserID: String = ''): Boolean;
```
Returns a Boolean if the call has completed correctly and fills the Galleries collection.

```
function GetSets(AUserID: String = ''): Boolean;
```
Returns a Boolean if the call has completed correctly and fills the Sets collection.

```
function GetPhotoStream(AUserID: String = ''): Boolean;
```
Returns a Boolean if the call has completed correctly and fills the PhotoStream collection.

```
function LastRequestData: AnsiString;
```
Returns the last requested data string. Can be used for debugging purposes.

```
procedure Logout;
```
This procedure can be used to force a user-logout when the user has checked the 'Keep me signed in' checkbox at the login page.

```
function PerformGetURL(AMethod: String; AParameters: TStringList = nil):
TJSONValue;
```
Performs a Post URL with a given method and an optional parameters list. Returns a TJSONValue object when performed correctly. Can be useful for unsupported flickr api calls in the TTMSLCLCloudFlickr.

Sample:

```
var
  jv: TJSONValue;
  jo, joUser: TJSONValue;
begin
  Result := False;
  jv := PerformGetURL('flickr.test.login');
```

```
  if Assigned(jv) then
  begin
    try
      jo := GetJSONValue(jv as TJSONObject, 'user');
      if Assigned(jo) and (jo is TJSONObject) then
      begin
        joUser := GetJSONValue(jo as TJSONObject, 'id');
        if Assigned(joUser) then
        begin
          FUserID := joUser.Value;
          Result := UserID <> '';
        end;

        joUser := GetJSONValue(jo as TJSONObject, 'username');
        if Assigned(joUser) and (joUser is TJSONObject) then
          FUserName := (joUser as
TJSONObject).Get('_content').JsonValue.Value;
      end;
    finally
      jv.Free;
    end;
  end;
```

```
function PerformPostURL(AMethod: String; AParameters: TStringList = nil):
TJSONValue;
```
Same functionality as the PerformGetURL, but uses a HTTP POST instead.

```
function UploadPhotoToStream(AFileName: String; ATitle: String = '';
ADescription: String = ''):TFlickrPhoto;
```
Uploads and returns photo to the photostream.

Gallery Methods:

```
function AddPhoto(APhotoID: String; ACreate: Boolean = False):
TFlickrPhoto; overload;
```
Adds a Photo with a specific ID to a gallery, if the Photo does not exist in the collection, the ACreate parameter can be used to create an Object of this Photo with the specific Photo ID.

```
function AddPhoto(APhoto: TFlickrPhoto): TFlickrPhoto; overload;
```
Adds a photo object to a gallery, the function internally uses the ID property of the Photo to upload.

```
procedure DownloadToFolder(const AFolder: string);
```
Download all the photos from a gallery to a specific folder.

```
function GetPhotos: Boolean;
```
Returns a Boolean if the call has completed correctly and fills the Galleries collection.

Set Methods:

```
function AddAndUploadPhoto(const AFileName: String; ATitle: String = '';
ADescription: String = ''): TFlickrPhoto;
```
Adds a photo to the set and uploads the photo to the photostream. A set cannot be empty and needs a primary photo to exist. The photo is first uploaded to the photo stream and then added to the set.

```
procedure AddFolder(const AFolder: String);
```
Adds a list of images from a folder to a set.

```
function AddPhoto(const APhotoID: String; ACreate: Boolean = False):
TFlickrPhoto; overload;
```
Adds a photo with a specific Photo ID to a set, if the photo does not exist in the collection, the ACreate parameter can be used to create a photo object.

```
function AddPhoto(APhoto: TFlickrPhoto): TFlickrPhoto; overload;
```
Adds a photo object to the set, the Photo ID property is used to add the photo to the set.

```
procedure DownloadToFolder(const AFolder: string);
```
Download all the photos from a set to a folder.

```
function GetComments: Boolean;
```
Retrieves the comments and fills the comment collection of a set.

```
function GetPhotos: Boolean;
```
Retrieves the photos and fills the photo collection of a set.

```
function Remove: Boolean;
```
Remove a set.

Photo Methods:

```
function DownloadLargest(ATargetFile: String = ''): Boolean; overload;
```
Download the largest photo of the list of photo sizes.

```
function DownloadSmallest(ATargetFile: String = ''): Boolean; overload;
```
Download the smallest photo of the list of photo sizes.

```
function Download(ASize: String; ATargetFile: String = ''): Boolean;
overload;
```
Download the photo with a specific size of the list of photo sizes.

```
function FindSizeByLabel(ASizeLabel: String): TFlickrSize;
```
Return the photo size with a specific size label.

```
function GetGeoLocation: Boolean;
```
Gets the geo location and sets the Latitude and Longitude properties.

```
function GetInfo: Boolean;
```

Gets the information and fills the properties of a photo.

```
function GetSizes: Boolean;
```
Gets the downloadable sizes of the photo and fills the sizes collection.

```
function GetTags: Boolean;
```
Gets the tags and fills the tags collection of the photo.

```
function GetComments: Boolean;
```
Gets the comments and fills the comments collection of a photo.

```
function Remove: Boolean; overload;
```
Remove a photo.

```
function SetTags: Boolean;
```
Updates / sets the tags on a photo, found in the Tags collection.

```
function SetGeoLocation: Boolean;
```
Updates / sets the geo-location on a photo, found in the Latitude and Longitude properties.

```
function SetMeta: Boolean;
```
Updates / sets the title and description on a photo, found in the Title and Description properties.

TTMSLCLCloudFourSquare

Usage

TTMSLCLCloudFourSquare is a component that facilitates access to the FourSquare service. It allows searching venues by keyword, category and location. Venue details can be retrieved as well as venue photos, tips and specials. The component also enables retrieving a FourSquare user's profile information and a list of places the authenticated user has checked in to.

Organisation

Properties:

Categories: TFourSquareCategories; Contains a list of FourSquare categories and sub-categories.

ID: string; The category ID.
Summary: string; The name of the category.
PluralName: string; The plural name of the category.
ShortName: string; The short name of the category.
IconURL: string; The URL of the icon image for the category.
Primary: Boolean; Indicates if this is the primary category for the parent venue object.
SubCategories: TFourSquareCategories; Contains a list of sub-categories.

Location: TFourSquareLocation; Contains information about the geographical location where the venues in the Venues collection are located.

Summary: string; The description of the location.
CountryCode: string; The country code related to the location.
Center: TFourSquareCoordinates; The center latitude and longitude coordinates for the location.
Bounds: TFourSquareBounds; The North East and South West coordinates for the location. All venues currently listed in Venues are located within these bounds.

UserProfile: TFourSquareUserProfile; Contains the profile information for a FourSquare user.

ID: string; The user ID.
FirstName: string; The first name of the user.
LastName: string; The last name of the user.
ImageURL: string; The URL for the profile image of the user.
FriendsCount: integer; The number of friends the user has on FourSquare.
CheckInsCount: integer; The number of times the user has checked in at a venue.
HomeCity: string; The home city of the user.
Gender: TFourSquareGender; The gender of the user.
Email: The email address of the user.
PhoneNumber: The phone number of the user.
FacebookID: The Facebook ID of the user.

TwitterID: The Twitter ID of the user.
Bio: The description of the user.
CheckIns: TFourSquareCheckIns; Contains a list of locations where the user has checked in.

ID: string; The CheckIn ID.
Created: TDateTime; The time when this CheckIn was created.
Comment: string; The comment for this CheckIn.
Venue: TFourSquareVenue; The venue related to this CheckIn.

Venues: TFourSquareVenues; Collection that contains a list of FourSquare venues.

ID: string; The venue ID.
Summary: string; The name of the venue.
Address: The address of the venue.
CrossStreet: string; The main street nearby the venue.
PostalCode: string; The postal code of the venue.
City: string; The city of the venue.
State: string; The state of the venue.
Country: string; The country of the venue.
CountryCode: string; The country code of the venue.
Latitude: double; The latitude coordinate of the venue.
Longitude: double; The longitude coordinate of the venue.
Phone: string; The phone number of the venue.
URL: string; The FourSquare URL of the venue.
Categories: TFourSquareVenueCategories; A list of FourSquare categories related to the venue.
Website: string; The website URL of the venue.
CheckInsCount: integer; The number of times a user has checked in at the venue.
UsersCount: integer; The number of unique users that have checked in at the venue.
TipCount: integer; The number of tips that have been posted for the venue.
HereNowCount: integer; The number of users that is currently checked in at the venue.
Tips: TFourSquareTips; A list of tips that has been posted for the venue.

ID: string; The tip ID.
Summary: string; The tip content text.
Created: TDateTime; The time the tip was posted.
ImageURL: string; The URL for the image related to the tip.
User: TFourSquareUserProfile; The user who has posted the tip.
LikesCount: integer; The number of likes this tip has received.
Liked: Boolean; Indicates if the currently authenticated user has liked the tip.

Photos: TFourSquarePhotos; A list of photos related to the venue.

ID: string; The photo ID.
Created: TDateTime; The time the photo was created.
ImageURL: string; The URL for the image.

Hours: TFourSquareHours; Contains information about the opening hours of the venue.

Status: string; Describes when the venue will open (when IsOpen = fiOpen) or when the venue will close (when IsOpen = fiClosed).
IsOpen: TFourSquareIsOpen; Indicates if the venue is currently open, closed or if opening hours are unknown.

Rating: double; The FourSquare rating of the venue.
Specials: TFourSquareSpecials; A list of FourSquare specials that are available at the venue.

ID: string; The special ID.
Title: string; The title of the special.
SpecialType: TFourSquareSpecialType; The type of the special.
Summary: The description of the special.
Description: The description of how to unlock the special.
FinePrint: The specific rules of the special.

Methods:

GetUserProfile(Profile: TFourSquareUserProfile = nil): string;
Gets the user profile information for the Profile.ID or for the authenticated user if no Profile is provided. *

GetCheckIns: string;
Gets a list of CheckIns for the currently authenticated user. Fills up the UserProfile.CheckIns collection. *

GetCategories: string;
Gets a list of FourSquare Venue Categories and Sub-Categories. Fills up the Categories collection. *

GetNearbyVenues(Latitude, Longitude: double; Location: string = ''; Keyword: string = ''; CategoryID: string = ''; ResultCount: integer = 10): string;
Gets a list of nearby venues based on Latitude and Longitude coordinates or Location. *

GetVenue(Venue: TFourSquareVenue): string;
Gets extra information about a venue using the Venue.ID value. *

GetSimilarVenues(VenueID: string): string;
Gets similar venues (in the same location) based on the VenueID value. Fills up the Venues collection. *

GetTips(Venue: TFourSquareVenue; Sort: TFourSquareSort = ftRecent; ResultCount: integer = 100; ResultOffset: integer = 0; Width: TFourSquareSize = fs100px; Height: TFourSquareSize = fs100px): string; *
Gets the Tips that have been posted for a venue. Fills up the Venues[].Tips collection. The size of the image linked in Venues[].Tips[].ImageURL is based on the Width & Height parameters.

GetPhotos(Venue: TFourSquareVenue; Width: TFourSquareSize = fs100px; Height: TFourSquareSize = fs100px) : string; *
Gets the Photos that have been posted for a venue. Fills up the Venues[].Photos collection. The size of the image linked in Venues[].Photos[].ImageURL is based on the Width & Height parameters.

CheckIn(VenueID: string; Comment: string = '');
Performs a CheckIn (with optional comment text) for the currently authenticated user a the venue specified through the VenueID.

* If the action is not executed successfully, the error type and error description are returned as a string value.

TTMSLCLCloudGCalendar

Usage

TTMSLCLCloudGCalendar is a component that provides access to the Google calendar service. It allows to retrieve a list of Google calendars and read, create, update & delete Google calendar events.

Organisation

Properties:

Calendars: TGCalendars; Class property that contains a list of Google calendars.

ID: string; The calendar ID.
Description: string; The description of the calendar.
Location: string; The location of the calendar.
Summary: string; The name of the calendar.
TimeZone: string; The time zone of the calendar.
Color: TGCalendarColor; The default event background color of the calendar. The index of the color type corresponds to the ID of the CalendarColors item.
BackgroundColor: TColor; The custom event background color of the calendar. If BackgroundColor is different from clNone the Color value may be ignored.
ForeGroundColor: TColor; The custom event text of the calendar.

Items: TGCalendarItems; Class property that contains a list of Google calendar events.

ID: string; The event ID.
ETag: string; The ETag of the event.
CalendarID: string; The ID of the calendar this event belongs to.
Description: string; The description of the event.
Summary: string; The name of the event.
StartTime: TDateTime; The start time of the event.
EndTime: TDateTime; The end time of the event.
IsAllDay: Boolean; Indicates if this is an all-day event.
Location: string; The location of the event.
Visibility: TVisbility; Indicates if the event is public, private or confidential.
Recurrence: string; The recurrency rule for a recurring event. (Not available for instances of the recurring event)
RecurringID: string; For an instance of a recurring event, this is the event ID of the parent event.
Sequence: integer; Indicates the number of times this event has been updated.
TimeZone: string; The time zone of the event. Required when adding a new recurring event if IsAllDay is false.
Color: TGItemColor; The background event color. The index of the color type corresponds to

the ID of the ItemColors item. If Color is icDefault, the Color/BackgroundColor value of the Calendar this event belongs to is used.
Attendees: TGAttendees; List of attendees for the event.

Summary: string; Name of the attendee.
Email: string; Email of the attendee.
Status: TResponseStatus; Indicates if the attendee has responded to the invitation and if the invitation was declined, tentatively accepted or accepted.

Reminders: TGReminders; List of reminders for the event.

Minutes: integer; The number of minutes before the start of the event when the reminder is initiated.
Method: TReminderMethod; Indicates which method is used to send the reminder (rmPopup, rmEmail, rmSMS).

ItemColors / CalendarColors: TGColors; Class property that contains a list of pre-defined Google calendar/event colors.
ID: integer; The color ID.
BackgroundColor: TColor; The background color definition.
ForegroundColor: TColor; The foreground color definition.

Methods:

GetCalendars;
Fill the list of calendars.

GetCalendar(ID: string; FromDate, ToDate: TDate); overload;
GetCalendar(ID: string; ChangedSince: TDateTime); overload;
GetCalendar(FromDate, ToDate: TDate); overload;
GetCalendar(ID: string); overload;
Fill the Items list with calendar events from a Google calendar for a certain timespan. If no ID is provided, the events are retrieved from the default Google calendar. If no FromDate/ToDate is specified the events are retrieved for the default timespan. When ChangedSince is used, only events that changed since the specified date/time are retrieved.

GetColors;
Fill the list of ItemColors and CalendarColors with the predefined values from the Google Calendar service.

GetItemByID(CalendarID, ItemID: string): TGCalendarItem;
Retrieve a single event based on the ID of the Google calendar and the event ID.

Add(Item: TGCalendarItem);
Add a new event to the calendar as specified by Item.CalendarID.

Update(Item: TGCalendarItem);
Update an event.

Example:

```
ci: TGCalendarItem;

ci.Summary := 'Item Name';
ci.Description := 'Item Description';
ci.Location := 'Item Location';
TMSLCLCloudGCalendar1.Update(ci);
```

Delete(Item: TGCalendarItem);
Delete an item.

TTMSLCLCloudGContacts

Usage

TTMSLCLCloudGContacts is a component that provides access to the Google contacts service. It enables to read, update, create and delete contacts. It also allows to read, update, create and delete contact groups.

Organisation

Properties:

Contacts: TGContacts; Contains a list of Google Contact items.

ID: string; The ID of the contact.
Birthday: TDateTime; The birthday of the contact.
Company: string; The company name related to the contact.
CustomItems: TGCustomItems; Contains a list of custom item data.

Key: string; The id for the custom item.
Value: string; The value for the custom item.

Emails: TGEmails; Contains a list of email addresses for the contact.

Address: string; The email address.
CustomType: string; The type description if EmailType is set to ftCustom.
EmailType: TGFieldType; The type of email address.
Primary: boolean; Indicates if this is the contact's primary email address.

FirstName: string; The first name of the contact.
MiddleName: string; The middle name of the contact.
LastName: string; The last name of the contact.
FullName: string; The full name of the contact.
NickName: string; The nickname of the contact.
OwnerName: string; The ownername of the contact.
Groups: TGContactGroups; Contains a list of Google Groups this contact belongs to.

ID: string; The ID of the group that the contact is assigned to.

ImageURL: string; The image for the contact.
InstantMessengers: TGInstantMessengers; Contains a list of instant messenger IDs for the contact.

CustomType: string; The type description if InstantMessengerType is set to itCustom.
ID: string; The ID for this instant messenger type.
InstantMessengerType: TGIMType; The type of instant messenger.

JobTitle: string; The job title of the contact.
NamePrefix: string; The name prefix of the contact.
NameSuffix: string; The name suffix of the contact.
Notes: string; The notes for the contact.
PhoneNumbers: TGPhoneNumbers; Contains a list of phone numbers for the contact.

CustomType: string; The type description if PhoneType is set to ptCustom.
Number: string; The phone number.
PhoneType: TGPhoneType; The type of phone number.
Primary: boolean; Indicates if this is the contact's primary phone number.

PostalAddresses: TGPostalAddresses; Contains a list of postal addresses for the contact.

Address: string; The full postal address.
AddressType: TFieldType; The type of postal address.
City: string; The city value of the address.
Country: string; The country value of the address.
CustomType: string; The type description if AddressType is set to ftCustom.
Neigborhood: string; The neighborhood value of the address.
POBox: string; The PO Box value of the address.
PostCode: string; The post code value of the address.
Primary: Boolean; Indicates if this is the contact's primary postal address.
Region: string; The region value of the address.
Street: string; The street value of the address.

Relations: TGRelations: Contains a list of relations of the contact.

CustomRelation: string; The type description if Relation is set to grCustom.
Relation: TGRelationType; The type of relation.
Value: string; The value text (Name) of the relation.

Title: string; The title of the contact.
Websites: TGWebSites: Contains a list of websites for this contact.

CustomType: string; The type description if WebsiteType is set to wtCustom.
URL: string; The URL of the website.
WebsiteType: TGWebsiteType; The type of website.

Groups: TGGroups; Contains a list of Google Group items.

ID: string; The ID of the group.
Summary: string; the group description.

Methods:

GetContactGroups;
Fill the list of groups.

GetContacts;
Fill the list of contacts.

Add(Item: TGContact);
Add a new Google contact item.

Example:

```
gc: TGContact;

gc.FirstName := edFirstName.Text;
gc.LastName := edLastName.Text;
TMSLCLCloudGContacts1.Add(gc);
```

Update(Item: TGContact);
Update a Google contact item.

Delete(Item: TGContact);
Delete a Google contact item.

AddGroup(Item: TGGroup);
Add a new Google group item.

UpdateGroup(Item: TGGroup);
Update a Google group item.

DeleteGroup(Item: TGGroup);
Delete a Google group item.

UpdateImage(Item: TGContact; Filename: string);
Add or update the image assigned to a Google contact item.

DeleteImage(Item: TGContact);
Delete the image assigned to a Google contact item.

TTMSLCLCloudGPlaces

Usage

TTMSLCLCloudGPlaces is a component that provides access to the Google places service. It enables to read places, navigate to a certain location and view all nearby places with their information.

Properties:

Places: TGPlacesItems : contains a list of all retrieved places
Place: TGPlacesItem: Contains all the info of a place

PlaceId: string : The unique place identifier
Title: string : The place title
Lat: double : The latitude of a place
Long: double : The longitude of a place
Icon: string : The accompanying icon
Open: Boolean : If the place is currently open
Vicinity: string : The vicinity of the place
Rating: string : The current rating
Phone: string : A universal phone number
Website: string : The places website
Photos: TPhotoItems : All photos
Types: TStringList : All describing types
Address: TGPlacesAddress : The full address

Photos: TPhotoItems : Contains a list of all photos of a place
Photo: TGPhotoItem : Contains all info of a photo

Reference: string : The photo identifier

Address: TGPlacesAddress : Contains all parts of the full address

FormattedAddress: string : A long string of full address info
Parts: TStringList : The address of a place in its separate parts

LastError: string : returns the last received error message if a search fails

Methods:

SearchNearby(ALong, ALat: double; AType: string = ''): boolean;
- Searches all nearby places (optionally from a type) by coordinates
Result is true when the request succeeds

SearchByText(ADiscription: string; AType: string): boolean;
- Searches all nearby places by a text string
Result is true when the request succeeds

HasNextPage; Boolean;
- Informs whether there are more matching places found

GetNextPlacesPage;
- Fetches the next page of places

AutoComplete(Search: string): string;
- Will try to autocomplete your search query

GetDetails(PlaceItem: TGPlacesItem);
- Gets the full Place information of the Place item passed as parameter

TTMSLCLCloudGTasks

Usage

TTMSLCLCloudGTasks is a component that provides access to the Google tasks service. It enables to read, update, create and delete taskslists and tasks. It also allows to read, update, create and delete task children.

Organisation

Properties:

TaskLists: TGTaskListItems : contains a list of all tasklists
TaskList: TGTaskListItem : Contains all info of a tasklist

Id: string : The tasklist identifier
Title: string : The tasklists title
Updated: TDateTime : When the tasklist was last updated
TaskItems: TGTaskItems: all tasks under this tasklist

Tasks: TGTaskItems : Contains a list of all tasks under a tasklist
Task: TGTaskItem : Contains all info of a (child)task

Id: string : The task identifier
Title: string : The tasks title
Updated: TDateTime : When the task was last updated
Position: integer : The tasks position inside the list
Status: string : The current status of the task : “Completed” | “needsaction”
Due: TDateTime : The tasks due date
Completed: TDateTime : The date and time when a task was completed

Methods:

GetTaskListItems(MaxResults: integer);
- Fill the list of tasklists

AddTaskList(TaskListName: string);
- Add a new task list item

UpdateTaskList(TaskListItem: TGTaskListItem);
- Updates a task list item

DeleteTaskList(TaskListItem: TGTaskListItem);
- Removes a tasklist

HasNextPage: boolean
- Informs whether there another page of tasklists

GetNextTaskListItems;
- Gets the next page of tasklists

GetTaskItems(MaxResults: integer; overloads..);
- Fetches all tasks for a specific tasklist

HasNextPage: Boolean;
- Informs whether there is another page of tasks

GetNextTaskItems;
- Fetches the next page of tasks

AddTaskToList(TaskItem: TGTaskItem);
- Add a task to the list

UpdateTask (TaskItem: TGTaskItem);
- Updates the specific task

DeleteTask(TaskItem: TGTaskItem);
- Removes a task from the tasklist

TTMSLCLCloudPicasa

Usage

TTMSLCLCloudPicasa is a component that facilitates access to the Google Picasa service. It allows retrieving your albums and photos as well as creating / deleting albums, uploading / downloading photos to albums. Photos can be uploaded / updated with a title / description and a geo location. Comments on photos can also be retrieved. The component also enables searching photos in public albums of other users.

Organisation

Properties:

Albums: TPicasaAlbums; Collection that contains a list of Picasa albums.

ID: string; The album ID.
Title: string; The title of the album.
ImageURL: string; The link to the image connected to this album.
MaxPhotoSize: TPicasaPhotoSize; The maximum size of the photo when downloaded using TPicasaPhoto.ImageURL. Set to psOriginal to retrieve the photo in its original size (default).
Photos: TPicasaPhotos; Contains a list of photos that the album contains.
Summary: string; The summary of the album.
Updated: TDateTime; The time when the album was last updated.

Photos: TPicasaPhotos; Collection that contains a list of Picasa photos.

ID: string; The photo ID.
AlbumID: string; The ID of the album the photo belongs to.
Author: TPicasaUser; The author of the photo.
Comments: TPicasaComments; Contains a list of comments for this photo.
FileName: string; The filename of the photo.
ImageURL: string; The link to the image file of the photo. (The size of the image is defined in TPicasaAlbum.MaxPhotoSize.
Latitude: double; The latitude value of the geolocation of the photo.
Longitude: double; The longitude value of the geolocation of the photo.
Summary: string; The description of the photo.
Tags: TStringList; The tags that have been set for the photo.
ThumbnailURL: string The link to the thumbnail image file of the photo.
Updated: TDateTime; The time when the photo was last updated.

Comments: TPicasaComments; Collection that contains a list of Picasa comments.

ID: string; The comment ID.
Author: TPicasaUser; The author of the comment.
PublishDate: TDateTime: The time when the comment was published.

Text: string; The text of the comment.
Title: string; The title of the comment.

Author: TPicasaUser; Class property that contains user information.

ID: string; The user id.
FullName: string; The full name of the user.
NickName: string; The nickname of the user.

Methods:

GetAlbums: Boolean;
Fill the list of albums, accessible via TMSLCLCloudPicasa.Albums

Album[].GetPhotos: TPicasaPhotos;
Fill the list of Picasa photos assigned to the album, accessible via TMSLCLCloudPicasa.Albums[Index].Photos

CreateAlbum(Album: TPicasaAlbum): Boolean;
Create a new Picasa album with properties set via Album parameter on the Picasa web service.

Example:

```
var
  Album: TPicasaAlbum;

  Album := TMSLCLCloudPicasa1.Albums.Add;
  Album.Title := edAlbumTitle.Text;
  Album.Summary := edAlbumDescription.Text;
  TMSLCLCloudPicasa1.CreateAlbum(Album);
```

DeleteAlbum(Album: TPicasaAlbum): Boolean;
Delete an existing Picasa album and all photos assigned to the album.

DeletePhoto(Photo: TPicasaPhoto): Boolean;
Delete an existing Picasa photo.

UploadPhoto(Album: TPicasaAlbum; FileName: string; Summary: string; string = ''; Tags: string = '';
Latitude: double = -1; Longitude: double = -1): TPicasaPhoto;
Upload a new photo to an existing Picasa album.

Example:

```
  if (OpenDialog1.Execute) then
  begin
    TMSLCLCloudPicasa1.UploadPhoto(TMSLCLCloudPicasa1.Albums[i],
      OpenDialog1.FileName, Description, Tags,
      Latitude, Longitude;
  end;
```

UpdatePhoto(Photo: TPicasaPhoto): Boolean;
Update an existing Picasa photo.

SearchPhotos(Keywords: string;var MoreResults: Boolean;const Page: integer = 0;const PageSize: integer = 10): integer;
Search in the available public Picasa albums for photos that match the provide keyword(s). MoreResults returns true if more photos are available, false otherwise. Returns the number of photos retrieved and retrieved photos are accessible via the collection property TMSLCLCloudPicasa.SearchResults: TPicasaPhotos

GetComments(Photo: TPicasaPhoto): TPicasaComments;
Fill the list of Picasa comments that have been made for the photo.

DownloadPhoto(TargetFile: string;Photo: TPicasaPhoto);
Download a photo to a local folder.

DownloadToFolder(const Folder: string;Album: TPicasaAlbum); overload;
Download all photos assigned to the album to a local folder.

DownloadToFolder(const Folder: string;Photos: TPicasaPhotos); overload;
Download all photos within the photos collection to a local folder.

AddFolderToAlbum(Folder, Title, Summary: string): TPicasaAlbum;
Upload all files from a local folder to a new or existing Picasa album.
If an album exists with a Title equal to the Title parameter value, the photos are added to the existing album, if not a new album is automatically created.

FindAlbumByTitle(AlbumTitle: string): TPicasaAlbum;
Returns a Picasa album for which the Title matches the AlbumTitle parameter value.

Events:

OnBeforeAddPhoto: TOnBeforeAddPhotoEvent;
Event fired before a photo is added with the UploadPhoto/AddFolderToAlbum method.

OnAfterAddPhoto: TOnAfterAddPhotoEvent;
Event fired after a photo has been added with the UploadPhoto/AddFolderToAlbum method.

OnBeforeDownloadPhoto: TOnBeforeDownloadPhotoEvent;
Event fired before a photo is downloaded with the DownloadPhoto/DownloadToFolder method.

OnAfterDownloadPhoto: TOnAfterDownloadPhotoEvent;
Event fired after a photo has been downloaded with the DownloadPhoto/DownloadToFolder method.

TTMSLCLCloudYouTube

Usage

TTMSLCLCloudYouTube is a component that facilitates access to the Google YouTube service. It allows retrieving your videos as well as uploading new videos. Photos can be uploaded with a title and description. The component also enables liking / unliking a video.

Organisation

Properties:

Videos: TYouTubeVideos; Collection that contains a list of YouTube videos.

ID: string; The video ID.
CategoryID: string; The id of the category the video belongs to.
ChannelID: string; The id of the YouTube channel the video belongs to.
ChannelTitle: string; The title of the YouTube channel the video belongs to.
Description: string; The description of the video.
ImageURL: string; The link to the thumbnail preview image of the video.
Link: string; The link to the video on YouTube.
PublishDate: TDateTime; The time when the video was published.
Rating: TYouTubeRating; The rating of the video. (Requires calling GetRating first)
Title: string; The title of the video.

Methods:

GetAllVideos: integer;
Fill the list of videos with all video items that belong to the authenticated user, accessible via TMSLCLCloudYouTube.Videos. MaxResults defines the maximum number of items that are returned.

GetFirstVideos(MaxResults: integer): integer;
Fill the list of videos with the first page of video items, accessible via TMSLCLCloudYouTube.Videos. MaxResults defines the maximum number of items that are returned. Returns the total number of videos.

GetNextVideos(MaxResults: integer): integer;
Adds the next page of video items to the list of videos, accessible via TMSLCLCloudYouTube.Videos. MaxResults defines the maximum number of items that are returned. Returns the total number of videos.

GetDetails(AVideo: TYouTubeVideo);
Get detailed information for a specific video. This fills the CategoryID and ChannelTitle properties for a specific video.

GetRating(AVideo: TYouTubeVideo): TYouTubeRating;
Get the authenticated user's rating for a specific YouTube video. This also assigns the Rating property of the video.

SetRating(AVideo: TYouTubeVideo; ARating: TYouTubeRating);
Set the authenticated user's rating for a specific YouTube video. This also assigns the Rating property of the video.

DeleteVideo(AVideo: TYouTubeVideo);
DeleteVideo(AVideoID: string);
Delete an existing YouTube video.

UploTMSLCLCloudideo(AFileName, ATitle, ADescription: string): TYouTubePhoto;
Upload a new video to a YouTube account.

Example:

```
  if OpenDialog1.Execute then
  begin
     TMSLCLCloudYouTube1.UploTMSLCLCloudideo(OpenDialog1.FileName,
edTitle.Text, edDescription.Text);
  end;
```

UpdateVideo(AVideo: TYouTubeVideo);
Update an existing YouTube video. Properties than can be updated are Title, Description and CategoryID.

Events:

OnBeforeAddVideo: TOnBeforeAddVideoEvent;
Event fired before a photo is added with the UploadPhoto/AddFolderToAlbum method.

OnAfterAddPhoto: TOnAfterAddVideoEvent;
Event fired after a photo has been added with the UploadPhoto/AddFolderToAlbum method.

TTMSLCLCloudInstagram

Usage

TTMSLCLCloudInstagram is a component that facilitates access to the Instagram service. It allows retrieving your photos, your followers (and their photos) and the profiles you are following (and their photos) as well as retrieving the tags, likes and comments for a photo. The component also enables searching photos by tag, username or location.

Organisation

Properties:

Users: TInstagramUsers; Collection that contains a list of Instagram users.

ID: string; The user ID.
Bio: string; The description of the user.
FullName: string; The full name of the user.
UserName: string; The name of the user.
Following: integer; The number of users the user is following.
Followers: integer; The number of users that is following the user.
Photos: integer; The number of photos the user has posted.

Photos: TInstagramPhotos; Collection that contains a list of Instagram photos.

ID: string; The photo ID.
Caption: string; The caption of the photo.
Location: TInstagramLocation; The location of the photo.

ID: string; The location ID.
Summary: string; The description of the location.
Latitude: string; The latitude of the location.
Longitude: string; The longitude of the location.

Filter: string; The filter used on the photo.
Tags: TStringList; The tags associated to the photo.
Comments: TInstagramComments; The comments for the photo.
CommentsCount: integer; The number of comments the photo has received.
Likes: TInstagramUsers; A list of users that has liked the photo.
LikesCount: integer; The number of likes the photo has received.
UserLikes: Boolean; Indicates if the currently authenticated user has liked the photo.
Link: string; A link to the photo on Instagram.
CreatedTime: TDateTime; The time the photo was created.
Images: TInstagramImages; The Images associated with the photo.
From: TInstagramUser; The user that posted the photo.

Tags: TInstagramTags; Collection that contains a list of Instagram tags.

Summary: string; The name of the tag.
MediaCount: integer; The number of Instagram photos that have received this tag.

Comments: TInstagramComments; Collection that contains a list of Instagram comments.

ID: string; The comment ID.
From: TInstagramUser; The user who posted the comment.
Text: string; The text content of the comment.
CreatedTime: TDateTime; The time the comment was posted.

Images: TInstagramImages; Collection that contains a list of Instagram images.

URL: string; The URL of the image.
Width: integer; The width of the image.
Height: integer; The height of the image.

Methods:

Follow(ID: string): string;
Follow the Instagram user associated with the ID.

UnFollow(ID: string): string;
Don't follow the Instagram user associated with the ID.

PostComment(ID, Text: string): string;
Post a comment to the Instagram photo associated with the ID.

DeleteComment(MediaID, CommentID: string): string;
Delete the comment (using CommentID) associated with the MediaID.

Like(ID: string): string;
Like the Instagram photo associated with the ID.

UnLike(ID: string): string;
UnLike the Instagram photo associated with the ID.

DownloadPhoto(ATargetFile, APhotoURL: String): Boolean;
Download a photo to a local folder.

GetProfile(UserID: string; User: TInstagramUser);
Retrieve the full profile information for an Instagram user.

GetFeed(Count: integer = 0; MaxID: string = ''; MinID: string = ''): Boolean;
Retrieve the Instagram feed for the currently authenticated user. Fills up the Photos collection.

GetMediaByUser(UserID: string = ''; Count: integer = 0; MaxID: string = ''; MinID: string = ''): Boolean;
Retrieve photos posted by the Instagram user associated with the UserID. Fills up the Photos collection.

GetMediaByTag(HashTag: string; MaxTagID: string = ''; MinTagID: string = ''): string;
Retrieve photos tagged with “HashtTag”. Fills up the Photos collection.

GetMediaByLocation(Latitude, Longitude: Double): Boolean;
Retrieve photos for a certain Location.

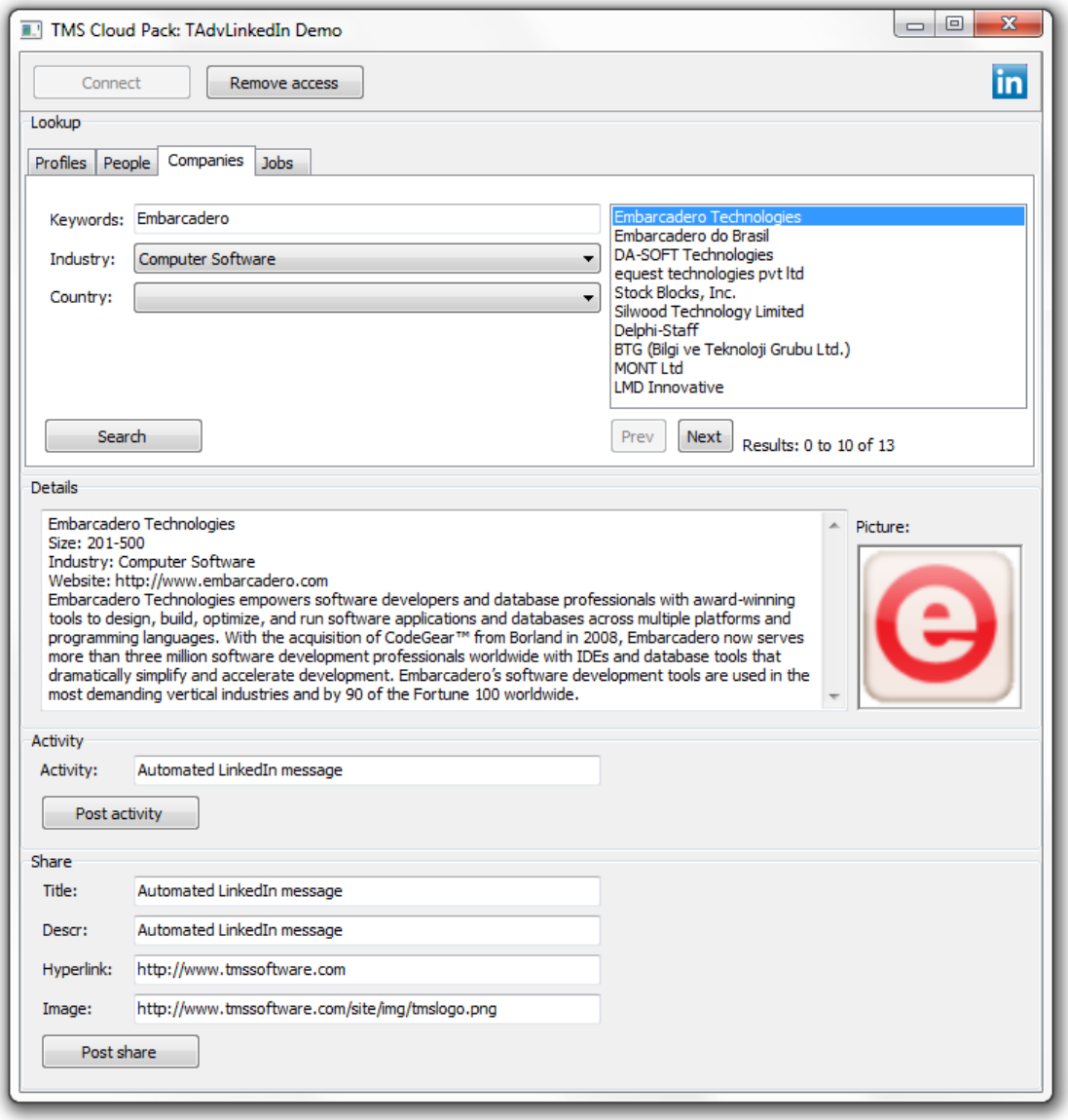
GetFollowing(UserID: string = '');
Retrieve the users that a user is following. Fills up the Users collection.

GetFollowers(UserID: string = '');
Retrieve the users that are following a user. Fills up the Users collection.

SearchUsers(Keyword: string);
Search Instagram users by keyword. Fills up the Users collection.

SearchTags(Keyword: string);
Search Instagram tags by keyword. Fills up the Tags collection.

TTMSLCLCloudLinkedIn



Usage

TTMSLCLCloudLinkedIn is a component that provides access to the LinkedIn API service. It allows to post an activity and share a message (with optional URL and image), retrieve a list of connections. It also enables searching for People, Companies and Jobs that are listed on LinkedIn.

Organisation

Properties:

DefaultProfile: TLinkedInProperty; Class property that contains the Facebook profile info for the currently authenticated user.

ID: string; The user's LinkedIn ID.
FormattedName: string; The user's formatted name.
FirstName: string; The user's first name.
LastName: string; The user's last name.
MaidenName: string; The user's maiden name.
EmailAddress: string; The contact email address granted by the user.
ConnectionsCount: integer; The number of connections of the user.
ConnectionsCapped: boolean; Indicates if the ConnectionsCount value is capped. (LinkedIn won't communicate if a user has 500+ connections)
Positions: TLinkedInPositions; List of the user's job positions.

ID: string; The ID for this position.
Company: TLinkedInCompany; The company related to this position.
EndMonth: integer; The month indicating when the position ended.
EndYear: integer; The year indicating when the position ended.
IsCurrent: Boolean; Indicates if this is the current position of the user.
Title: string; The job title held at this position, as indicated by the user.
Summary: string; The summary of the user's position.
StartMonth: integer; The month indicating when the position began.
StartYear: integer; The year indicating when the position began.

Location: string; The location of the user.
CountryCode: string; The country code of the user.
CurrentShare: TLinkedInShare; The current share of the user.

ID: string; The ID for this share.
Author: TLinkedInAuthor; The author for this share.

ID: string; The ID for the author.
FirstName: string; The first name of the author.
LastName: string; The last name of the author.

Comment: string; The comment for this share.
Description: string; The description for this share.
TimeStamp: TDateTime; The time this share was posted.
Title: string; The title of this share.
URL: string; The URL for this share.

PublicProfileURL: string; The URL for the user's LinkedIn profile.
Distance: integer; The degree distance of the fetched profile from the user who fetched the

profile.
LastModified: TDateTime; The time when the user's profile was last modified.
Associations: string; The associations of the user.
Honors: string; The honors of the user.
ProposalComments: string; The description of how the user approaches proposals.
Publications: TLinkedInPublications; A list of the user's publications.

ID: string; The ID for this publication.
PublicationDate: TDateTime; The time this publication was published.
PublicationTitle: string; The title of this publication.

Patents: TLinkedInPatents; A list of the user's patents.

ID: string; The ID for this patent.
PatentDate: TDateTime; The time related to this patent.
PatentTitle: string; The title of this patent.

Languages: TLinkedInLanguages; A list of the user's languages.

ID: string; The ID for this language.
LanguageName: string; The name of this language.

Skills: TLinkedInSkills: A list of the user's skills.

ID: string; The ID for this skill.
SkillName: The name of this skill.

Certifications: TLinkedInCertifications; A list of the user's certifications.

ID: string; The ID for this certification.
CertificationName: The name of this certification.

Educations: TLinkedInEducations; A list of the user's educations.

ID: string; The ID for this education.
Degree: string; The description of the degree received at this institution.
SchoolName: string; The name of the school as indicated by the user.

Courses: TLinkedInCourses; A list of the user's courses.

ID: string; The ID for this course.
CourseName: string; The name of this course.
Number: string; The course number assigned, as entered by the user.

VolunteerExperiences: TLinkedInVolunteerExperiences; A list of the user's volunteer experiences.

ID: string; The ID of this volunteer experience.
Organization: string; The name of the organization the user has volunteered with.
Role: string; The role the member has performed as a volunteer.

Recommendations: TLinkedInRecommendations; A list of the recommendations the user has received.

ID: string; The ID of this recommendation.
RecommendationText: string; The text of the recommendation received.
RecommendationType: string; Indicates the type of recommendation that was selected by the person making the recommendation.
Recommender: TLinkedInProfile; The profile of the person who made the recommendation.

Following: TLinkedInObjects; A list of the items the user is following.

ID: string; The ID for this object.
ObjectName: string; The name of this object.
ObjectType: TLinkedInObjecType; The type of this object.

JobBookmarks: TLinkedInJobBookmarks; A list of jobs that the user has bookmarked.

ID: string; The ID for this job bookmark.
Company: TLinkedInCompany; The company this job is listed for.
Description: string; The description of this job.
IsApplied: boolean; Indicates if the user has applied for the job.
IsActive: boolean; Indicates if the job listing is active.
PositionTitle: string; The title of the position.
SavedTimeStamp: TDateTime; The time this job bookmark was created.

Groups: TLinkedInGroups; A list of LinkedIn groups the user is a member of.

ID: string; The ID for this group.
GroupName: string; The name of this group.
MembershipState: string; The state of the user's membership to the specified group.

BirthDate: TDateTime: The user's birth date.
Resources: TLinkedInResources; A list of URLs the user has chosen to share on their LinkedIn profile.

ID: string; The ID for this resource.
ResourceName: string; The name of the resource.
URL: string; The URL of the resource.

PhoneNumbers: TLinkedInPhoneNumbers; A list of the user's phone numbers.

PhoneNumber: string; The phone number.
PhoneType: TLinkedInPhoneType; The type of this phone number.

IMAccounts: TLinkedInIMAccounts; A list of the user's instant messaging accounts.

AccountName: string; The name of this IM account.
AccountType: TLinkedInIMType; The type of this IM account.

MainAddress: string; The user's address.
Headline: string; The user's headline.
Industry: string; The industry the LinkedIn member has indicated their profile belongs to.
Interests: string; A description of the user's interests.
PictureUrl: string; An url to the user's profile picture.
Specialties: string; A description of the user's specialties.
Summary: string; A description of the user's professional profile.

Connections: TLinkedInConnections; A list of user profiles the currently authenticated user is connected with.

Classes:

TLinkedInCompany:

ID: string; The ID for the company.
BlogURL: string The URL for the company blog.
CompanyName: string; The name of the company.
CompanyType: string; The type of the company.
Description: string; The description of the company.
ImageURL: string; The URL of the company logo image.
Industries: TLinkedInObjects; A list containing the names of the company's industries.
Locations: TLinkedInLocations; A list of the company's locations.

Street1: string; The first line of the street address.
Street2: string; The second line of the street address.
City: string; The city for this location.
State: string; The state for this location.
PostalCode: string; The postal code for this location.
RegionCode: string; The region code for this location.

CountryCode: string; The country code for this location.
Phone1: string; The company phone number for this location.
Phone2: string; The second company phone number for this location.
Description: string; The company location description.

Size: string; The number range of employees at the company.
Specialties: TStringList; A list of the company's specialties.
StockExchange: string; The stock exchange the company is in.
Ticker: string; The company ticker identification for the stock exchange.
TwitterID: string; The ID for the company Twitter feed.
Website: string; The company website address.

TLinkedInJob:

ID: string; The ID for this job.
Company: TLinkedInCompany; The hiring company for this job.
CountryCode: string; The country code for this job.
Description: string; The description for the position.
ExperienceLevel: string; The experience level for the position.
Industries: TLinkedInObjects; A list of industries related to this position.
IsActive: boolean; Indicates of the job listing is active.
JobURL: string; The URL for the job listing.
JobPoster: TLinkedInProfile; The user who posted the job listing.
Location: string; The location for this job.
Position: string; The description of the job position.
PostingDate: TDateTime; The date of the job listing.
Salary: string; The salary for this job listing.
SkillsAndExperience: string; The description of the skills and experience needed for the listed position.

TCompanyResult:

ID: string; The ID for this company.
CompanyName: string; The name of the company.

TJobResult:

ID: string; The ID for this job.
Company: TLinkedInCompany; The company for this job.
Description: string; The description for this position.
Location: string; The location for this job.
Position: string; The description of the job position.

TPeopleResult:

ID: string; The ID for this person.
FirstName: string; The first name of this person.
LastName: string; The last name of this person.

Methods:

Activity(const: Msg: string): boolean;
Post an activity message on the stream of the user.

GetDefaultProfile;
Fill the DefaultProfile class property with the profile information of the user.

GetConnections;
Fill the Connections list with the connections of the user.

GetProfile(const ID: string): TLinkedInProfile;
Get the profile information of a user based on the user ID.

GetCompanyInfo(const ID: string): TLinkedInCompany;
Get the company information based on the ID of the company.

GetJobInfo(const ID: string): TLinkedInJob;
Get the job information based on the ID of the job.

SearchCompany(SP: TCompanySearchParam; var ResultCount; const Page: integer): TCompanyResults;
Search for companies based on the values in TCompanySearchParam.
Returns a list of companies.

SearchJob(SP: TJobSearchParam; var ResultCount; const Page: integer): TJobResults;
Search for job listings based on the values in TJobSearchParam.
Returns a list of jobs.

SearchPeople(SP: TPeopleSearchParam; var ResultCount; const Page: integer): TPeopleResults;
Search for people based on the values in TPeopleSearchParam.
Returns a list of people.

Example:

```
var
  sp: TPeopleSearchParam;
  pr: TPeopleResults;
  resultcount, maxresult: integer;

  sp.Keywords := edKeywords.Text;
  sp.FirstName := edFirstName.Text;
  sp.LastName := edLastName.Text;
```

```
  sp.CompanyName := edCompany.Text;
  sp.CountryCode := icUnitedStates;

  pr := TMSLCLCloudLinkedIn1.SearchPeople(sp, resultcount, 0);
```

Share(const Title, Msg, HyperLink, ImageLink: string): boolean;
Share a message with optional title, hyperlink and imagelink on the stream of the user.

Example:

```
  TMSLCLCloudLinkedIn1.Share(edTitle.Text, edDescription.Text,
edHyperlink.Text, edImageLink.Text);
```

TTMSLCLCloudLiveCalendar

Usage

TTMSLCLCloudLiveCalendar is a component that provides access to the Windows Live calendar service. It allows to retrieve a list of Windows Live calendars and read, create, update & delete Windows Live calendar events.

Organisation

Properties:

Calendars: TLiveCalendars; Contains a list of Live calendars.

ID: string; The calendar ID.
Summary: string; The name of the calendar.
Description: string; The description of the calendar.
ReadOnly: Boolean; Indicates if events can be added/updated for this calendar.

Items: TLiveCalendarItems; Contains a list Live calendar events.

ID: string; The event ID.
CalendarID: string; The ID of the calendar this event belongs to.
Summary: string; The name of the event.
Description: string; The description of the event.
StartTime: TDateTime; The start time of the event.
EndTime: TDateTime; The end time of the event.
IsAllDay: Boolean; Indicates if this is an all-day event.
Location: string; The location of the event.
Visibility: TVisbility; Indicates if the event is public, private or confidential.
IsRecurrent: Boolean; Indicates if this is a recurring event.
Recurrence: string; Description of the event recurrence.

Methods:

GetCalendars;
Fill the list of Calendars.

GetCalendar(ID: string; FromDate, ToDate: TDate); overload;
GetCalendar(FromDate, ToDate: TDate); overload;
GetCalendar(ID: string); overload;
Fill the Items list with calendar events from a Live Calendar for a certain timespan. If no ID is provided, the events are retrieved from the default Live Calendar. If no FromDate/ToDate is specified the events are retrieved for the default timespan.

GetItemByID(CalendarID, ItemID: string): TLiveCalendarItem;
Retrieve a single event based on the ID of the Live Calendar and the event ID.

Add(Item: TLiveCalendarItem);
Add a new event to the Calendar as specified by Item.CalendarID.

Update(Item: TLiveCalendarItem);
Update an event.

Example:

```
ci: TLiveCalendarItem;

ci.Summary := 'Item Name';
ci.Description := 'Item Description';
ci.Location := 'Item Location';
TMSLCLCloudLiveCalendar1.Update(ci);
```

Delete(Item: TLiveCalendarItem);
Delete an event.

TTMSLCLCloudLiveContacts

Usage

TTMSLCLCloudLiveContacts is a component that provides access to the Windows Live contacts service. It enables to read and create contacts.

Organisation

Properties:

Items: TLiveContactItems; Contains a list of Live Contact items.

ID: string; The ID of the contact.
FirstName: string; The first name of the contact.
LastName: string; The last name of the contact.
FullName: string; The full name of the contact.
Gender: TGender; The gender of the contact.
IsFriend: Boolean; Indicates if the contact is a friend.
IsFavorite: Boolean; Indicates if the contact has been added as a favorite.
UserID: string; The user ID of the contact.
BirthDay: integer; The day part of the contact's birthday.
BirthMonth: integer; The month part of the contact's birthday.

Methods:

GetContacts;
Fill the list of Items.

Add(Item: TLiveContactItem);
Add a new Live contact item.

Example:

```
ci: TLiveContactItem;

ci.FirstName := edFirstName.Text;
ci.LastName := edLastName.Text;
TMSLCLCloudLiveContacts1.Add(ci);
```

TTMSLCLCloudURLShortener

Usage

TTMSLCLCloudURLShortener is a component that enables to make a short version of a long URL.

Organisation

Methods:

ShortenURL(URL: string): string;

Returns a short version of the provided URL.

TTMSLCLCloudWeather

Usage

TTMSLCLCloudWeather is a component that uses the Wunderground.com weather service to get information on the current weather status for a location, a 4 day weather forecast or a 10 day weather forecast.

Organisation

Properties:

property Condition: TWeatherCondition;

Class property that holds various properties returning the current weather condition for a location. This class property is filled with information when GetConditions() is called.

property Location: TWeatherLocation;
Exact location information, including longitude, latitude, altitude.

property TempC: double;
Temperature in degrees Celcius

property TempF: double;
Temperature in Fahrenheit

property Weather: string;
Textual weather description

property WindDir: string;
Wind direction in text format

property WindDegrees: integer;
Wind direction in degrees

property WindMph: double;
Wind speed in miles per hour

property WindKph: double;
Wind speed in kilometer per hour

property FeelsLikeC: double;
Corrected temperature feeling in degrees Celcius

property FeelsLikeF: double;
Corrected temperature feeling in Fahrenheit

property PressureMb: double;
Air pressure in millibar

property PressureIn: double;
Air pressure in inch of Mercury

property DewPointF: double;
Dew point in Fahrenheit

property DewPointC: double;
Dew point in degrees Celcius

property VisibilityMi: double;
Visibility in Miles

property VisibilityKm: double;
Visibility in kilometer

property UV: string;
UV index

property Icon: string;
Name of the icon as text

property IconURL: string;
URL of icon representing the weather condition

property Precip1hrMetric: string;
Precipitation

property TextForeCast: TTextWeatherForeCast;

Collection of TTextWeatherForeCastItem items that contain the forecast text for the next 4 or 10 days, depending on whether GetForecast() or GetForeCast10Day() was called.

The TTextWeatherForeCastItem exposes following properties:

property Icon: string;
Name of the icon as text

property IconURL: string;
URL of the image that represents the weather condititon

property Title: string;
Title of the weather forecast.

property Text: string;
Textual weather description with numbers in non metric format.

property TextMetric: string;
Textual weather description with numbers in metric format.

property ForeCast: TWeatherForeCast;

Collection of TWeatherForeCastItem items that contain the weather conditions for the next 10 days for a specific location.
The TWeatherForeCastItem is a class that exposes following properties:

property Date: TDateTime;
Date of the forecast

property TempHighF: double;
Max temperature of the day in Fahrenheit

property TempHighC: double;
Max temperature of the day in Celcius

property TempLowF: double;
Min temperature of the day in Fahrenheit

property TempLowC: double;
Min temperature of the day in Celcius

property Conditions: string;
Textual weather condition description

property IconURL: string;
URL of icon representing the weather

property MaxWindMph: double;
Max wind speed in miles per hour

property MaxWindKmh: double;
Max wind speed in kilometer per hour

property MaxWindDir: string;
Direction where wind is coming most from

property MaxWindDegrees: integer;
Direction in degrees where wind is coming most from

property AveWindMph: double;
Average wind speed in miles per hour

property AveWindKmh: double;
Average wind speed in kilometer per hour

property AveWindDir: string;
Average wind direction

property AveWindDegrees: integer;
Average wind direction in degrees

property AveHumidity: integer;
Average humidity during the day

property MaxHumidity: integer;
Max humidity during the day

property MinHumidity: integer;
Min humidity during the day

property SnowAllDay: TWeatherSnow;
Snow expectations for all day

property SnowDay: TWeatherSnow;
Snow expectations for day

property SnowNight: TWeatherSnow;
Snow expectations for night

property QPFAllDay: TWeatherQPF;
Rain forecast for all day

property QPFDay: TWeatherQPF;
Rain forecast for day

property QPFNight: TWeatherQPF;
Rain forecast for night

Methods:

procedure GetConditions(ACountry, ACity: string);

Retrieves the weather conditions for the location and fills the TTMSLCLCloudWeather.Conditions property with the info retrieved.

procedure GetForecast(ACountry, ACity: string);

Retrieves the weather 4 day forecast for the location and fills the TTMSLCLCloudWeather.Forecast collection and TTMSLCLCloudWeather.TextForecast collection property with the info retrieved.

procedure GetForecast10Day(ACountry, ACity: string);

Retrieves the weather 10 day forecast for the location and fills the TTMSLCLCloudWeather.Forecast collection and TTMSLCLCloudWeather.TextForecast collection property with the info retrieved.

TTMSLCLCloudPushOver

Usage

TTMSLCLCloudPushOver is a component that sends push messages to the PushOver client running on iOS devices. This service allows to send free messages to one or more devices with the PushOver client from a Windows application.

Organisation

TTMSLCLCloudPushOver descends from TCloudBase and exposes following properties and methods:

Properties

property PushOverMessage: TPushOverMessage

Class property giving access to all settings associated with a PushOver message:

property User: string;
PushOver ID of the user.

property Title: string;
Title of the message to send. Can be up to 50 characters.

property Device: string;
Device name of the user where to send the message. Only required when the user has multiple devices and the message needs to be sent to a single device only.

property Message: string;
Content of the message itself. Can be up to 500 characters.

property URL: string;
Optional URL to associate with the message.

property URLTitle: string;
Optional title to set for the URL that will be sent with the message

property Priority: TMessagePriority;
Can be any of following values: (mpNone,mpQuiet,mpHighPriority,mpConfirmation);
mpNone is the default property value for normal priority push message
mpQuiet is message without popup & sound on the device
mpHighPriority set when message needs to be sent with high priority
mpConfirmation sends a push message with receipt confirm prompt

property TimeStamp: TDateTime;
When different from zero, sends a timestamp different from the server timestamp when the message is sent.

property Sound: TMessageSound;
Select the sound to associate with the push message. The value can be:

```
TMessageSound = (msDefault,msBike,msBugle,msCashRegister,msClassical,msCosmic,
  msFalling,msGamelan,msIncoming,msIntermission,msMagic,msMechanical,msPianoBar,
  msSiren,msSpaceAlarm,msTugBoat,msAlien,msClimb,msPersistent,msEcho,msUpDown,msNone);
```

Methods

function PushMessage(AUser, AMessage: string): boolean;

Sends a simple text message to user with PushOver ID AUser.

function PushMessage(AUser, ATitle, AMessage: string): boolean;

Sends a simple title and text message to user with PushOver ID AUser.

function PushMessage(AUser, ADevice, ATitle, AMessage: string): boolean;

Sends a simple title and text message to device ADevice belonging to user with PushOver ID AUser

function PushMessage(AMessage: TPushOverMessage): boolean;

Sends a message with all properties as defined by AMessage.

Sample code

```
begin
  // set the PushOver application ID
  TMSLCLCloudPushOver1.App.Key := APPID;
  // fill in the message details
  TMSLCLCloudPushOver1.PushOverMessage.User := edUser.Text;
  TMSLCLCloudPushOver1.PushOverMessage.Title := edTitle.Text;
  TMSLCLCloudPushOver1.PushOverMessage.Message := edMemo.Lines.Text;
  TMSLCLCloudPushOver1.PushOverMessage.URL := edURL.Text;
  TMSLCLCloudPushOver1.PushOverMessage.Device := edDevice.Text;
  TMSLCLCloudPushOver1.PushOverMessage.Sound :=
TMessageSound(integer(edSound.Items.Objects[edSound.ItemIndex]));
  // Send the message

TMSLCLCloudPushOver1.PushMessage(TMSLCLCloudPushOver1.PushOverMessag
e);
end;
```

TTMSLCLCloudCloudConvert

Usage

TTMSLCLCloudCloudConvert is a component that can convert one file format to another file format. The input file is uploaded to the API service and after the conversion the output file is downloaded. The available conversion types can be consulted at: https://cloudconvert.com/formats

Methods

function ConvertFile(FileName: string; InputFormat: string; OutputFormat: string; Converter: string = ''; Options: TStringList = nil): string;

Method to convert the file with FileName from InputFormat to OutputFormat. Optionally the Converter value can be added to select which converter to use if multiple converters are available for the desired conversion. Optionally Options can be defined for the conversion. The result is the the URL of the converted file. Details of the conversion are returned in the ConvertResults property.

function ConvertAndDownload(InputFileName: string = ''; OutputFileName: string = ''; InputFormat: string = ''; OutputFormat: string = ''; Converter: string = ''; Options: TStringList = nil): boolean;

Method to convert the InputFileName to OutputFileName. If no InputFormat and/or OutputFormat is provided, the file format is automatically determined based on the filename extension(s). Optionally the Converter value can be added to select which converter to use if multiple converters are available for the desired conversion. Optionally Options can be defined for the conversion. Returns true if the conversion succeeded, false otherwise. Details of the conversion are returned in the ConvertResults property.

procedure Download(Url, TargetFile: String);

Method to download a converted file. The result of the ConvertFile method or the ConvertResults.OutputFile.Url can be used as the Url parameter.

Sample code

1) **Convert a DOC file to a PDF file**

```
TMSLCLCloudCloudConvert1.ConvertAndDownload('test.doc', 'test.pdf');
```

2) **Convert the first 5 pages of a PDF file to PNG files**
   Note: Converting a multi-page file (such as a PDF file) to an image format will result in a separate image file for each converted page. The images are returned in a single ZIP file.

```
  options := TStringList.Create;
  options.Add('page_range=1-5');
  TMSLCLCloudCloudConvert1.ConvertAndDownload('test.pdf', 'test.zip', '',
'png', '', options);
  options.Free;
```

Conversion details returned by TTMSLCLCloudCloudConvert are:

TMSLCLCloudCloudConvert.ConvertResults.ID: The conversion ID
TMSLCLCloudCloudConvert.ConvertResults.StarTime: The timestamp when the conversion started
TMSLCLCloudCloudConvert.ConvertResults.EndTime: The timestamp when the conversion was finished
TMSLCLCloudCloudConvert.ConvertResults.ExpiryTime: The timestamp when the converted file download expires
TMSLCLCloudCloudConvert.ConvertResults.InputFile: The input file details
TMSLCLCloudCloudConvert.ConvertResults.OutputFile: The output file details
TMSLCLCloudCloudConvert.ConvertResults.OutputFile.FileName: The output file name
TMSLCLCloudCloudConvert.ConvertResults.OutputFile.FileSize: The output file size
TMSLCLCloudCloudConvert.ConvertResults.OutputFile.Url: The output file url
TMSLCLCloudCloudConvert.ConvertResults.OutputFile.Files: The list of files contained in the output file (only if the output file is a ZIP file)

TTMSLCLCloudBarcode

Usage

TTMSLCLCloudBarcode is a component that can generate barcodes and QR codes. The barcode or QR code is provided to the service and an URL to an image with the barcode or QR code is returned or saved as an image file.

Properties

BarcodeOptions: Configure the barcode settings.

Height: integer; The height of the barcode
ReverseColor: Indicates if the barcode is displayed as white on black instead of black on white
ShowBorder: Boolean; Indicates if a border is displayed around the barcode
ShowText: Indicates if the text value of the barcode is displayed beneath the barcode
Width: Boolean; The width of the barcode

ImageType: TBarcodeImageType; Set the image format that is used when a barcode is generated. Supported image formats are GIF, JPG and PNG.

QRCodeOptions: Configure the QR code options
ECCLevel: TQRCodeECC; Set the Error Correction Capibility Level of the QR code
Size: TQRCodeSize; Set the size of the QR code

Methods

function GetBarcode(AValue: string; AFileName: string; AType: TBarcodeType = btC39): string;

Method to generate a barcode based on AValue. Optionally the TBarcodeType can be set with AType. The supported barcode types are: ‘Code 39’, ‘Code 128a’, ‘Code 128b’, ‘Code128c’ and ‘2 of 5 interleaved’. The result is saved as an image file with AFileName as filename. The image format can be set with the ImageType property and the available options can be configured in BarcodeOptions.

function GetQRcode(AValue: string; AFileName: string): string;

Method to generate a QR code based on AValue. The result is saved as an image file with AFileName as filename. The image format can be set with the ImageType property and the available options can be configured in QRcodeOptions.

function GetBarcodeURL(AValue: string; AType: TBarcodeType = btC39): string;

Method to generate a barcode based on AValue. Optionally the TBarcodeType can be set with AType. The supported barcode types are: ‘Code 39’, ‘Code 128a’, ‘Code 128b’, ‘Code128c’ and ‘2 of 5 interleaved’. The result is an URL to an image file with the barcode. The image format can be set with the ImageType property and the available options can be configured in BarcodeOptions.

function GetQRcodeURL(AValue: string): string;

Method to generate a QR code based on AValue. The result is an URL to an image file with the barcode. The image format can be set with the ImageType property and the available options can be configured in QRcodeOptions.

Sample code

1) **Display a barcode with a TTMSLCLCloudCloudImage that contains the value '12345'**

```
TMSLCLCloudCloudImage1.URL := TMSLCLCloudBarcode1.GetBarcodeURL('12345');
```

2) **Generate an image file with a QR code with a width and height of 210 pixels that contains the value 'http://www.tmssoftware.com'**

```
TMSLCLCloudBarcode1.QRcodeOptions.Size := qs210;
TMSLCLCloudBarcode1.GetQRCode('http://www.tmssoftware.com', 'QRcode.png');
```

TTMSLCLCloudIPLocation

Usage

TTMSLCLCloudIPLocation is a component that can determine the geolocation of an IP address. It can determine the geolocation of the machine where it is executed, the geolocation of any IP address or the geolocation of a server based on its domain name.

Sample code

```
  if TMSLCLCloudIPLocation1.GetIPLocation then
    Showmessage('I am located in country : ' +
TMSLCLCloudIPLocation1.IPInfo.CountryName);

  if TMSLCLCloudIPLocation1.GetIPLocation('107.128.206.99') then
    Showmessage('IP address is located in country : ' +
TMSLCLCloudIPLocation1.IPInfo.CountryName);

  if TMSLCLCloudIPLocation1.GetIPLocationFromServer('www.google.com') then
    Showmessage('Google.com is located in country : ' +
TMSLCLCloudIPLocation1.IPInfo.CountryName);
```

Details returned by TTMSLCLCloudIPLocation are:

TMSLCLCloudIPLocation.IPInfo.CountryCode: code of the country. Example: Brasil = BR
TMSLCLCloudIPLocation.IPInfo.CountryName: name of the country
TMSLCLCloudIPLocation.IPInfo.RegionCode: code of the region
TMSLCLCloudIPLocation.IPInfo.RegionName: name of the region
TMSLCLCloudIPLocation.IPInfo.ZIPCode: ZIP code
TMSLCLCloudIPLocation.IPInfo.City: name of the city
TMSLCLCloudIPLocation.IPInfo.Metrocode: code of the nearby metro
TMSLCLCloudIPLocation.IPInfo.Areacode: code of the nearby area
TMSLCLCloudIPLocation.IPInfo.Longitude: longitude coordinate of the geolocation
TMSLCLCloudIPLocation.IPInfo.Latitude: latitude coordinate of the geolocation

TTMSLCLCloudPryv

Usage

TTMSLCLCloudPryv is a component that provides access to the Pryv service.
The component supports retrieving, inserting, updating and deleting of streams and events.
Different types of events are supported: Text, Picture, File, Position (Geolocation) …

Properties

The **TTMSLCLCloudPryv** has following public properties:

Events: TList; The collection of events
Streams: TPryvStreams; The collection of streams and substreams

The Events collection can contain a number of different class objects, all derived from **TPryvObject**.

The **TPryvObject** has following properties:

ID: string; The ID of the event
StreamID: string; The ID of the stream this event belongs to
DateTime: TDateTime; The datetime associated with the event
Description: string; The description of the event
Tags: TStringList; The list of tags associated with the event
Created: TDateTime; The datetime when the event was created
Updated: TDateTime; The datetime when the event was last updated

The **TPryvText** has following properties (in addition to all properties from TPryvObject):

Content: string; The content text of the event

The **TPryvValue** has following properties (in addition to all properties from TPryvObject):

Content: string; The measurement value of the event
UnitValue: string; The measurement unit of the event

This class can be used to hold any event type that is defined by a measurement unit (i.e: “length/cm”) and a measurement value (i.e.: ‘100’).
A list of currently available numerical event types can be found at: http://pryv.github.io/event-types/#directory-numerical-types.
Besides the documented event types, it’s also possible to define custom event types.

Sample:
This code snippet demonstrates how to add a new Event with a custom type ('mymeasurement/myvalue') to an existing Stream:

```
var
  val: TPryvValue;
begin

   val := TPryvValue.Create;
   TMSLCLCloudPryv1.Events.Add(val);
   val.Content := '10';
   val.StreamID := TMSLCLCloudPryv1.Streams[0].ID;
   val.UnitValue := 'mymeasurement/myunit';
   val.DateTime := Now;
   val.Tags.CommaText := 'custom, value';
   val.Description := 'my custom value';
   TMSLCLCloudPryv1.AddEvent(val);
end;
```

The **TPryvPosition** has following properties (in addition to all properties from TPryvObject):

Latitude: double; The latitude coordinate of the event
Longitude: double; The longitude coordinate of the event

The **TPryvPicture** has following properties (in addition to all properties from TPryvObject):

FileName: string; The FileName of the image file associated with the event
ImageURL: string; The url of the image file associated with the event

The **TPryvFile** has following properties (in addition to all properties from TPryvObject):

FileName: string; The FileName of the file associated with the event
FileURL: string; The url of the file associated with the event

The **TPryvStreamItem** has following properties:

ID: string; The stream id
ParentID: string; The id of the parent stream (only available for substreams)
Summary: string; The name of the stream
Created: TDateTime; The datetime when the stream was created
Updated: TDateTime; The datetime when the stream was last updated
SubStreams: TPryvStreams; The collection of substreams for this stream

Methods

AddStream(Stream: TPryvStreamItem);
Creates a new stream with the values defined in the Stream parameter. This method can be used to add streams as well as substreams.

DeleteStream(Stream: TPryvStreamItem);
Deletes an existing stream.

GetStreams;
Fills the Streams collection with the existing streams and substreams of the currently authenticated user.

UpdateStream(Stream: TPryvStreamItem);
Updates an existing stream with the values defined in the Stream parameter.

AddEvent(Event: TPryvObject);
Creates a new event with the values defined in the Event parameter. This method can be used to add Events of types TPryvText, TPryvValue and TPryvPosition.

AddEvent(Event: TPryvFile; FileName: string);
Creates a new event with the values defined in the Event parameter. This method can only used to add Events of type TPryvFile. The local file referenced in the FileName parameter will be uploaded to the Pryv service and associated with the Event.

AddEvent(Event: TPryvPicture; FileName: string);
Creates a new event with the values defined in the Event parameter. This method can only used to add Events of type TPryvPicture. The local image file referenced in the FileName parameter will be uploaded to the Pryv service and associated with the Event.

DeleteEvent(Event: TPryvObject);
Deletes an existing event.

GetEvents(Streams: TStringArray);
GetEvents(Streams: TStringArray; Tags: TStringArray; MaxResults: integer; Page: integer; SortAscending: Boolean);
GetEvents(FromTime: TDateTime; ToTime: TDateTime; MaxResults: integer; Page: integer; SortAscending: Boolean);
Fills the Events collection with the existing events of the currently authenticated user.
Parameters:
Streams: If assigned, only events that belong to the specified stream ids and their sub-streams will be returned.
Tags: If assigned, only events that have any of the provided tags assigned will be returned.
MaxResults, Page: can be used to retrieve paged results
SortAscending: If true, events will be sorted from oldest to newest
FromTime: The start time of the timeframe events are retrieved for
ToTime: The end time of the timeframe events are retrieved for

UpdateEvent(Event: TPryvObject);
Updates an existing event with the values defined in the Event parameter.

Note that for TPryvPicture the FileName and ImageURL properties can not be updated, for TPryvFile the FileName and FileURL properties can not be updated.

Download(Event: TPryvFile; TargetFile: string);
Download(Event: TPryvPicture; TargetFile: string);
The (image) file associated with the Event will be downloaded to the TargetFile.

Sample

This code snippet shows how to add a new Event of type Position to a new Stream:

```
Var
  it: TPryvStreamItem;
  pos: TPryvPosition;

begin
  it := TMSLCLCloudPryv1.Streams.Add;
  it.Summary := edStreamName.Text;
  TMSLCLCloudPryv1.AddStream(it);

  pos := TPryvPosition.Create;
  TMSLCLCloudPryv1.Events.Add(pos);
  pos.Latitude := StrToFloat(edLatitude.Text);
  pos.Longitude := StrToFloat(edLongitude.Text);
  pos.StreamID := it.ID;
  pos.DateTime := Now;
  pos.Tags.CommaText := 'location';
  pos.Description := 'position description';
  TMSLCLCloudPryv1.AddEvent(pos);
end;
```

TTMSLCLCloudTrello

Usage

TTMSLCLCloudTrello is a component that provides access to the Trello service. Trello is the free, flexible and visual way to organize anything with anyone.
The component supports retrieving, inserting, updating and deleting of boards, lists, cards, attachments and checklists.

TTMSLCLCloudCustomTrello

Methods
GetBoardLabels(ABoard: TTMSLCLCloudTrelloBoard);
Retrieves the list of all available labels of the board. The labels can be read after retrieval via Aboard.LabelNames.

GetBoardMemberships(ABoard: TTMSLCLCloudTrelloBoard);
Retrieves a list of all members (invited and confirmed) of the board. Aboard.Memberships.

GetMember(AUsername: string = 'me');
Gets the member details by the provided username. The member details are returned via TMSLCLCloudTrello.Member.

GetLists(ABoard: TTMSLCLCloudTrelloBoard);
Retrieves a list of all available board lists (open and closed). The lists can be read after retrieval via Aboard.Lists.

GetCards(AList: TTMSLCLCloudTrelloList);
Retrieves a list of all available card inside the provided list (open and closed).
The members, checklists, labels and actions are included in this call. The cards can be read via AList.Items[index]: TTMSLCLCloudTrelloCard.

SetBoardClosed(ABoard: TTMSLCLCloudTrelloBoard ; IsClosed: Boolean = True);
Toggles the closed property of the board.

SetListClosed(AList: TTMSLCLCloudTrelloList; IsClosed: Boolean = True);
Toggles the closed property of the list.

SetCardClosed(ACard: TTMSLCLCloudTrelloCard; IsClosed: Boolean = True);
Toggles the closed property of the card.

DeleteCard(ACard: TTMSLCLCloudTrelloCard);
Permanently removes the card from the list.

AddBoardMember(ABoard: TTMSLCLCloudTrelloBoard; AMemberEmail: string);
Adds or invites a member to the board by emailaddress.

AddCardComment(ACard: TTMSLCLCloudTrelloCard; AComment: string);
Adds a comment to the card.

AddCardChecklist(ACard: TTMSLCLCloudTrelloCard; AChecklistName: string);
Adds a new checklist to the card with empty checklist items. The checklist is accessible via the Card.CheckLists list.

AddCardChecklistItem(ACard: TTMSLCLCloudTrelloCard; AChecklist: TTMSLCLCloudTrelloCardCheckList; ACheckItemName: string);
Adds a new checklist item to the provided checklist inside a card.

AddCardLabel(ACard: TTMSLCLCloudTrelloCard; ALabel: TTMSLCLCloudTrelloBoardLabelNames);
Adds a label to the card. The label is accessible via the Card.Labels list.

AddCardAttachment(ACard: TTMSLCLCloudTrelloCard; AFileName: string);
Adds a file attachment to the card.

AddCardAttachmentUrl(ACard: TTMSLCLCloudTrelloCard; AUrl: string);
Adds an URL attachment to the card.

UpdateCardComment(AAction: TTMSLCLCloudTrelloCardAction; AComment: string);
Updates an existing comment of the provided card. Note that comments are named Actions in Trello and can be retrieved via the Card.Actions list.

UpdateCardDescription(ACard: TTMSLCLCloudTrelloCard; ADescription: string);
Updates the cards description.

UpdateCardDueDate(ACard: TTMSLCLCloudTrelloCard; ADateTime: TDateTime);
Updates the cards due date.

UpdateCardCheckListName(ACheckList: TTMSLCLCloudTrelloCardCheckList; AChecklistName: string);
Updates the provided checklists name.

UpdateCardCheckListItemName(ACard: TTMSLCLCloudTrelloCard; ACheckList: TTMSLCLCloudTrelloCardCheckList; AChecklistItem: TTMSLCLCloudTrelloCardCheckListItem; AChecklistItemName: string);
Updates the provided check items name.

UpdateCardLabel(ABoard: TTMSLCLCloudTrelloBoard; ACard: TTMSLCLCloudTrelloCard; ALabel: TTMSLCLCloudTrelloCardLabel; ALabelName: string);
Updates the name of the specified cards label.

DeleteCardComment(AAction: TTMSLCLCloudTrelloCardAction);
Deletes the specified cards comment.

DeleteCardLabel(ACard: TTMSLCLCloudTrelloCard; ALabelID: string);
Deletes the specified label.

DeleteCardAttachment(ACard: TTMSLCLCloudTrelloCard; AAttachment: TTMSLCLCloudTrelloCardAttachment);
Permanently deletes the specified attachment.

AddBoard(AName: string): string;
Adds a board by name. The board is accessible via the TMSLCLCloudTrello.Member.Boards list.

AddCardMember(ABoard: TTMSLCLCloudTrelloBoard; ACard: TTMSLCLCloudTrelloCard; AUsername: string): Boolean;
Adds a member to the specified card.

AddList(ABoard: TTMSLCLCloudTrelloBoard; AName: string): string;
Adds a list by name to the specified board.

AddCard(AList: TTMSLCLCloudTrelloList; AName: string): string;
Adds a card by name to the specified list.

Properties

Member: TTMSLCLCloudTrelloMember
The full member with all its boards.

TTMSLCLCloudTrelloMember

Methods

GetBoardByID(ABoardId: string): TTMSLCLCloudTrelloBoard;
Returns the full board by provided board id.

Properties

Username: string
The username of the member.

ID: string
The ID of the member.

Initials: string
The initials of the member. Ex: John Doe = JD

Membertype: string
The account type (normal, admin, disabled)

AvatarSource: string
The source of the avatar.

AvatarHash: string
The hash of the avatar.

Bio: string
The description on the members account.

FullName: string
The full written name of the member.

URL: string
The url to the members page.

Email: string
The emailaddress of the member.

GravatarHash: string
The hash of the gravatar.

IDBoards: TList<string>
A list of all members board ID's .

IDOrganizations: TList<string>
A list of all members Organization ID's.

Boards: TTMSLCLCloudTrelloBoardList
All the members boards

TTMSLCLCloudTrelloBoard

Methods

GetListByName(AName: string): TTMSLCLCloudTrelloList;
Gets the full board list by its name.

GetListById(AID: string): TTMSLCLCloudTrelloList;
Gets the full board list by its ID.

Properties

ID: string
The ID of the board.

Desc: string
The boards description.

Name: string
The name of the board.

Closed: Boolean
True if the board has been closed.

IDOrganization: string
The corresponding organization ID.

ShortLink: string
The shortened URL of the board

DateLastActivity: TDateTime
The date and time of the last activity

Starred: Boolean
If the board has been starred

URL: string
The URL of the board.

Memberships: TList<TTMSLCLCloudTrelloBoardMembership>
All members who are connected to the board.

LabelNames: TList<TTMSLCLCloudTrelloBoardLabelNames>
List of available labels.

DateLastView: TDateTime
The date and time of the last view

ShortURL: string
The shortened URL to the board.

Lists: TTMSLCLCloudTrelloListList
A list of all lists inside the board.

TTMSLCLCloudTrelloList

Methods

GetCardById(AID: string): TTMSLCLCloudTrelloCard;
Gets the full Card by its ID.

Properties

ID: string
The ID of the list.

Name: string
The name of the list.

Closed: Boolean
Indicates if the list has been closed.

Position: Single
The position of the list inside the board.

Subscribed: Boolean
If you are Subscribed.

Cards: TTMSLCLCloudTrelloCardList
All the Cards inside the current list.

TTMSLCLCloudTrelloCard

Methods

GetActionById(AID: string): TTMSLCLCloudTrelloCardAction;
Gets a cards action by its ID. This can be a comment, activity, member add notification, …

GetAttachmentById(AID: string): TTMSLCLCloudTrelloCardAttachment;
Gets the specific attachment by its ID.

getLabelById(AID: string): TTMSLCLCloudTrelloCardLabel;
Gets the specific Card label by its ID.

GetChecklistById(AID: string): TTMSLCLCloudTrelloCardCheckList;
Gets the specific Card Checklist by its ID.

Properties

ID: string
The ID of the card.

Closed: Boolean
Indicates if the Card has been closed.

DateLastActivity: TDateTime
The date and time of the last activity.

Desc: string
The description of the card.

DescData: string
The data of the card.

IDShort: Integer;
A shortened ID of the card.

Name: string
The name of the card.

Position: Single
The position of the card inside the list.

ShortLink: string
The shortened URL of the card.

Due: TDateTime
The due date of the card in date and time format.

Email: string
The emailaddress of the Card.

ShortURL: string
The shortened URL of the card.

Subscribed: Boolean
Indicates if you are subscribed to the card.

URL: string
The full URL of the card.

Attachments: TList<TTMSLCLCloudTrelloCardAttachment>
A list of all the attachments.

Members: TList<TTMSLCLCloudTrelloCardMember>
A list of all linked members.

Actions: TTMSLCLCloudTrelloCardActionList
A list of all actions of the card.

CheckLists: TTMSLCLCloudTrelloCardCheckListList
A list of all the checklists of the card.

Labels: TList<TTMSLCLCloudTrelloCardLabel>
A list of all the labels that are linked to this card.

TTMSLCLCloudTrelloCardChecklist

Methods

GetChecklistItemById(AID: string): TTMSLCLCloudTrelloCardCheckListItem;
Gets the specific checklist item by its ID.

Properties

ID: string
The ID of the checklist.

Name: string
The display name of the checklist.

Position: Single
The position of the checklist inside the card.

CheckItems: TTMSLCLCloudTrelloCardCheckListItemList
A list of all the items of this checklist.

TTMSLCLCloudTrelloCardChecklistItem

Properties

ID: string
The ID of the checklist item.

Name: string
The name of the checklist item.

NameData: string

The data of the checklist item.

Position: Single

The position of the checklist item inside the current checklist.

State: string

The current state of the checklist item.

TTMSLCLCloudTrelloCardAction

Properties
ID: string
The ID of the action

IDMemberCreator: string
The ID of the member that caused the action.

Text: string
The corresponding text of the action.

ActionType: string
The type of the action. Ex: CommentCard

Date: TDateTime
The date and time of the action.

TTMSLCLCloudGMail

Usage

TTMSLCLCloudGMail is a component that provides access to the Google Mail service. It enables to read, create, edit and send mails.

TTMSLCLCloudCustomGMail

Methods

procedure GetMails(Labels: string = 'INBOX'; AMaxResults: Integer = 50);
Retrieves all the mails from the specified mail folder. The emails can be read via the TTMSLCLCloudGMail.Mails list.

GetLabels;
Retrieves all the labels. After retrieval, the labels can be read via the TMSLCLCloudGMail.Label list.

SendMessage(AMessage: TTMSLCLCloudGMailMessage; ALabels: string = 'SENT'): Boolean;
Sends a message, with the option to assign one or more labels to the message. When multiple labels are used, the list needs to be a comma separated string.

UpdateMessageLabels(AMessageId, ALabels: string): Boolean;
Updates the messages labels. When multiple labels are used, the list needs to be a comma separated string.

Properties

Mails: TTMSLCLCloudGMailMessageList
All the mails from the folder specified when calling GetMails.

TTMSLCLCloudGMailMessage

Properties

ID: string
The (unique Google) ID of the email.

BCCRecipients: TStringList
The list of all BCC recipients.

Body: string
The body text of the message.

CCRecipients: TStringList
The list of all CC recipients.

FileName: string
The name of the associated file.

Headers: TList<TTMSLCLCloudGMailHeader>
All messages headers.

MessageType: TTMSLCLCloudGMailMessageType
The type of the Message. <plain text or HTML>

Snippet: string
A short substring of the actual body text.

Subject: string
The subject of the message.

ToRecipients: TStringList
The list of all recipients.

TTMSLCLCloudGMailLabel

Properties

ID: string
The (unique Google) ID of the label.

Name: string
The name of the label.

LabelType: string
The type of the label.

TTMSLCLCloudGAnalytics

Usage

TTMSLCLCloudGAnalytics is a component that provides access to the Google Analytics service. It enables to get insights into how visitors find and use your site, and how to keep them coming back.

TTMSLCLCloudCustomGAnalytics

Methods

GetData(AStartDate: string = 'today'; AEndDate: string = 'today'; AMaxResults: Integer = 1000): string;
Gets all the data that was requested by adding dimensions & metrics to the *RequestData* property.

Properties

Data: TGData
The returned data.

RequestData: TGRequestData
The place to distinguish the needed dimensions and metrics.

ErrorMessages: TGErrorMessages
Can set the error message for metrics and dimensions, when that error occurs.

For more information about metrics and dimensions, please see the analytics reporting api and analytics realtime reporting api

TGData

Property

Data: TList<TArray<string>>
A list of all requested data (in the same order as requested)

TTMSLCLCloudStripe

TTMSLCLCloudCustomStripe

Methods

GetProducts(ActiveState: TStripeProductActiveState = pasAll; ShippableState: TStripeProductShippableState = pssAll; ALimit: Integer = 10): string;
Gets all the products for the authorized account

CreateOrder(AOrder: TStripeOrderItem): string;
Creates an order with given properties

Properties

Orders: TStripeOrderItems
All created orders

Products: TStripeProductItems
All the products for the authorized account

UserID: string
The user identifier for the authorized account

TStripeProductItem

Properties

ProductID: string
The identifier of the product

Created: string
Stringified timestamp of the product

Updated: string
Stringified timestamp of the product

LiveMode: Booelan
True if the product is available at live mode

Productname: string
The name of the product

Caption: string
The products caption

Description: string
The products description

Active: Boolean
True if the product is active

Shippable: Boolean
True if the product is shippable

Url: string
The available url of the product

SKUs: TStripeProductSKUItems
A collection of the products stock keeping units.

TStripeProductSKUItem

Properties

SKUID: string
The identifier of the sku

Created: string
Stringified timestamp of the sku

Updated: string
Stringified timestamp of the sku

LiveMode: Booelan
True if the product is available at live mode

ProductID: string
The parent product identifier

Active: Boolean
True if the sku is active

Price: Integer
The price of the sku in cents

Currency: string
A string notification of the currency, for example: eur, usd, …

TStripeOrderItem

Properties

Amount: Integer
The amount of this order

ApplicationFee: Integer
The fee of the application in cents

Charge: Integer
The carge of the order in cents

OrderId: string
The order identifier

Created: string
Stringified timestamp of the Order

Updated: string
Stringified timestamp of the order

Status: TStripeCustomer
The status of the order

Currency: string
The currency of the order

Email: string
The email of the customer of the order

Items: TStripeOrderItemItems
All items of the order

Shipping: TStripeShipping
The shipping properties

SelectedShippingMethod: string
The selected Shipping method

TStripeOrderItemItem

Properties

Amount: Integer
The amount of items

Currency: string
The currency of the item

Description: string
The visible description of the item

ParentID: string
The parent order identifier

Quantity: Integer
The quantity of item orders

ItemType: string
The type of the item

TTMSLCLCloudImage

TTMSLCLCloudImage is a helper component to display images from an URL. TTMSLCLCloudImage descends from the LCL component TImage and inherits all its properties & behaviour. With TTMSLCLCloudImage you get an additional property TTMSLCLCloudImage.URL: string
When setting the URL property to a valid URL to an image (PNG,JPEG,BMP,GIF), it will load the image from the URL and display it. To clear the image, set the URL property to an empty string.

Example:

TMSLCLCloudImage.URL := 'http://www.tmssoftware.com/site/img/tmslogo.png'

TTMSLCLCloudmyCloudData

Usage

TTMSLCLCloudmyCloudData is a component that provides seamless access to the myCloudData.net service that allows to create tables with meta data of choice to store data in the cloud. A user can have access to one or more tables. After login, the collection of tables available to the user is returned with TTMSLCLCloudmyCloudData.GetTables and accessible via TTMSLCLCloudmyCloudData.Tables. A table on myCloudData.net is represented by the class TMyCloudDataTable. A table has metadata, entities, filters, sort order and shares. The table entities are represented by the class TMyCloudDataEntity. The shares are represented by the class TMyCloudDataShare. The meta data for a table is retrieved via Table.GetMetaData. The entities are retrieved via the Table.Query function and the list of shares is retrieved with Table.Shares. The filter is accessible via the collection Table.Filters and the sort ordering can be setup via the collection Table.SortOrder.

Organisation

TMyCloudDataEntity

The TMyCloudDataEntity class is the class that wraps an entity (in database terminology also often referred to as record). The TMyCloudDataEntity class has following methods & properties:

Properties

property Value[AName: string]: Variant;
property Blob[AName: string]: TMyCloudDataBlob;

Methods

procedure Update;
procedure Insert;
procedure Delete;

To get or set a value for a field within the entity, you can use Value[AName: string]: Variant.

Example:

```
entity.Value['NAME'] := 'Bill Gates;
entity.Value['NETWORTH'] := 46546114646;
entity.Value['BIRTHDATE'] := DatePicker.Date;
```

All supported field types: int/int64/double/string/date/datetime/time/boolean can be get or set this way. The binary blob fields can be accessed as:

```
blob := entity.Blob['BIN'];
```

With blob an instance of the class TMyCloudDataBlob. This explained further in the paragraph covering this class.

TMyCloudDataEntities

This is the collection of entities retrieved via Table.Query.
Typical operations on entities are as such:

1. Create a new entity in the cloud storage:

```
var
  ent: TMyCloudDataEntity;
begin
  ent := Table.Entities.Add;
  ent.Value['NAME'] := 'Elon Musk';
  ent.Value['STATE'] := 'California';
  ent.Value['COMPANY'] := 'Tesla';
  ent.Insert;
end;
```

2. Update an existing entity in the cloud storage:

```
var
  ent: TMyCloudDataEntity;
begin
  ent := Table.Entities[x];
  ent.Value['COMPANY'] := 'SpaceX';
  ent.Update;
end;
```

3. Delete an entity permanently from the cloud storage:

```
var
  ent: TMyCloudDataEntity;
begin
  ent := Table.Entities[y];
  ent.Delete;
end;
```

TMyCloudDataBlob

The TMyCloudDataBlob class is a wrapper class for binary data stored in a blob in the cloud service. Note that the blob storage capability is not available for a free account in myCloudData.net but requires a subscription.
The TMyCloudDataBlob class has following properties and methods:

Properties

property Table: TMyCloudDataTable
Table to which the blob belongs

property Entity: TMyCloudDataEntity
Entity to which the blob belongs

property Field: string
Name of the field holding the blob

Methods

procedure LoadFromFile(AFileName: string);
Load data from AFileName into the blob field

procedure SaveToFile(AFileName: string);
Get data from the blob field and save it to a file

procedure LoadFromStream(AStream: TStream);
Load data from the stream into the blob field

procedure SaveToStream(AStream: TStream);
Get data from the blob field and save it to a stream

A typical operation to store some binary data into a blob field in a new entity would be:

Example:

```
var
  ent: TMyCloudDataEntity;
begin
  blob: TMyCloudDataBlob;

  ent := Table.Entities.Add;
  blob := ent.Blob['BIN'];
  blob.LoadFromFile('mybinfile.bin');
end;
```

Note that for performance reasons, blobs are returned via the entity only and retrieved from the cloud storage at the time SaveToStream() or SaveToFile() is executed.

TMyCloudDataTable

This class represents the table in the cloud storage and is part of the set of tables in the collection TTMSLCLCloudmyCloudData.Tables retrieved with TTMSLCLCloudmyCloudData.GetTables.

A TTMSLCLCloudmyCloudData class has following properties and methods:

Properties

property ID: int64;
Read-only property returning the unique identifier of the table

property OwnerID: int64;
Read-only property returning the unique owner identifier of the table

property IsOwner: boolean;
Read-only property returns true when the logged in user owns the table

property Name: string;
Gets or sets the name of the table

property MetaData: TMyCloudDataMetaData;
Access to the metadata of the table via a collection after calling GetMetaData

property Entities: TMyCloudDataEntities;
Access to the entities of the table via a collection after calling Query

property Filters: TMyCloudDataFilters;
Access to the filter conditions for a query via a collection

property SortOrder: TMyCloudDataSortOrderList;
Access to the sort order settings for a query via a collection

Methods

function GetMetaData: boolean;
Retrieves the metadata from a table

function SetMetaData: boolean;
Updates the metadata of the table on the cloud storage with Table.MetaData

function Query: boolean; overload;
Simply query for all entities of the table

function Query(Fields: TStringArray): boolean; overload;
Query with specifier of selection of fields to return

function Query(AFields: TStringList): boolean; overload;
Query with specifier of selection of fields to return

procedure Share(Email: string; Permissions: TMyCloudDataPermissions);
Share a table with another user defined by email

procedure RemoveShare(Email: string);
Remove an existing share with another user via email

procedure Delete;
Delete the table from the cloud storage

function GetShares: TMyCloudDataShares;
Retrieve the list of email addresses & permissions with who the table was shared

property Permissions: TMyCloudDataPermissions read FPermissions;
Permissions the user has on the table. Permissions are: CRUD, i.e. create/read/update/delete

To perform a simple query, use:

```
Table.Query;
```

This will fill the Entities collection with the entities retrieved from the cloud storage.

Table.Query(['NAME','COMPANY','BIRTHDATE'];

This will fill the Entities collection but the entities will only hold the fields NAME, COMPANY, BIRTHDATE

To filter data, following code can be used:

```
var
  filter: TMyCloudDataFilter;
begin
  Table.Filters.Clear; // removes all filter conditions
  filter := Table.Filters.Add('NAME', coLike, 'Musk', loNone);
  Table.Query;
end;
```

Or alternatively:

```
var
  filter: TMyCloudDataFilter;
begin
  Table.Filters.Clear; // removes all filter conditions
  filter := Table.Filters.Add;
  filter.FieldName := 'NAME';
```

```
  filter.ComparisonOperator := coLike;
  filter.Value := 'Musk';
  filter.LogicalOperator := loNone;
  Table.Query;
end;
```

Note that the ComparisonOperator can be any of the following values:
coEqual, coNotEqual, coLike, coGreater, coGreaterOrEqual, coLess, coLessOrEqual, coStartsWith, coEndsWith, coNull, coNotNull;

The LogicalOperator that sets the logical operation between two sequential filter conditions can be: loAND, loOR, loNone

To specify the sort order for a query, the Table.SortOrder collection can be used:

```
  Table.SortOrder.Clear;
  Table.SortOrder.Add('NAME', soAscending);
  Table.SortOrder.Add('COMPANY', soDescending);
  Table.Query;
```

Or alternatively:

```
var
  sortorder: TMyCloudDataSortOrderItem;
begin
  Table.SortOrder.Clear;
  sortorder := Table.SortOrder.Add;
  sortorder.FieldName := 'NAME';
  sortorder.SortOrder := soAscending;
  sortorder := Table.SortOrder.Add;
  sortorder.FieldName := 'COMPANY';
  sortorder.SortOrder := soDescending;
  Table.Query;
end;
```

TMyCloudDataTables

This is the collection of all the tables a user has access to, either because the user owns the table or the table was shared with the user.

TMyCloudDataMetaDataItem

This class holds the information about a single meta data item in the meta data collection of a table. The meta data item class has following properties:

Properties

property PropertyName: string;
Gets or sets the field name

property DataType: TFieldType
Gets or sets the field type. The field type can be any of following value: ftString, ftWideString, ftInt, ftBigInt, ftFloat, ftBlob, ftSmallInt, ftWord, ftBoolean, ftDate, ftDateTime, ftTime

property Data: Boolean
Returns true when the meta data item pertains actual data

property Size: integer
Optionally gets or sets the size of a field (only available for fields of type ftString, ftWideString)

The meta data item is part of the meta data collection TMyCloudDataMetaData accessible via Table.MetaData.

Typical operations on the meta data are:

1. Retrieval of meta data & list all fields in a listbox

```
var
   i: integer;
begin
   Table.GetMetaData;

   for i := 0 to Table.MetaData.Count - 1 do
   begin
     listbox.Items.Add(Table.MetaData[i].PropertyName);
   end;
end;
```

2. Creating meta data for a new table

```
   Table.MetaData.Clear;
   Table.MetaData.Add('NAME', ftWideString, 50);
   Table.MetaData.Add('COMPANY', ftWideString, 50);
   Table.MetaData.Add('BIN', ftBlob);
   Table.SetMetaData;
```

Or alternatively

```
var
  metadata: TMyCloudDataMetaDataItem;
begin
   Table.MetaData.Clear;
   metadata := Table.MetaData.Add;
   metadata.PropertyName := 'NAME';
```

```
    metadata.DataType := ftWideString;
    metadata.Size := 50;
    metadata := Table.MetaData.Add;
    metadata.PropertyName := 'COMPANY';
    metadata.DataType := ftWideString;
    metadata.Size := 50;
    metadata := Table.MetaData.Add;
    metadata.PropertyName := 'BIN';
    metadata.DataType := ftBlob;
    Table.SetMetaData;
end;
```

TMyCloudDataShares

To share a table with another myCloudData.net user, call:

```
Table.Share('myfriend@company.com', [pCreate, pRead, pUpdate]);
```

This adds a share with user myfriend@company.com. Note that it is required that myfriend@company.com is recognized as a valid myCloudData.net user. The myCloudData.net will not send a notification of the share itself. It is the responsibility of the user to do so. Here a share is created with all permissions except the permission to delete entities in the table.

To remove the share at a later time, call:

```
Table.RemoveShare('myfriend@company.com');
```

When the share existed for the user it will be removed.

To see with who a table is shared, use:

```
var
  shares: TMyCloudDataShares;
begin
  shares := Table.GetShares;

  for i := 0 to shares.Count - 1 do
  begin
    listbox.Items.Add(shares[i].Email);
  end;
end;
```

TTMSLCLCloudmyCloudData

TTMSLCLCloudmyCloudData is the class that wraps the entire access to the myCloudData.net service. Its interface is compatible with the Apple CloudKit and Google DataStore components, so

usage of myCloudData.net, CloudKit or Google DataStore can be fully abstracted. Note that in addition to the common interface, myCloudData.net introduces many unique capabilities only available in myCloudData.net.

Methods & properties available in TTMSLCLCloudmyCloudData:

Properties

property TableId: int64
Gets or sets the unique ID of the table TTMSLCLCloudmyCloudData can work on

Methods

Table related methods:

function GetTables: boolean;
Retrieves the list of tables and makes these accessible via TTMSLCLCloudmyCloudData.Tables

function TableByName(AName: string): TMyCloudDataTable;
Retrieves on table based on its name

function TableList: TStrings;
Retrieves the list of available tables as stringlist

function AddTable(ATable: TMyCloudDataTable): boolean;
Creates a new table from an existing TMyCloudDataTable class

function CreateTable(ATableName: string): TMyCloudDataTable;
Creates a new table with name ATableName and returns an instance to the table

function DeleteTable(AID: int64): boolean; overload;
Deletes a table based on its unique ID

function DeleteTable(ATable: TMyCloudDataTable): boolean; overload;
Deletes a table based on an existing TMyCloudDataTable class

function UpdateTable(ATable: TMyCloudDataTable): boolean;
Updates table info, such as name, permissions based on an existing TMyCloudDataTable class

function ShareTable(ATable: TMyCloudDataTable; AEmail: string; APermissions: string): boolean;
Share a table with specific permissions with another myCloudData.net user

function GetTableShares(ATable: TMyCloudDataTable): boolean;
Fills the Table.Shares collection with shares found

Metadata related methods:

function GetMetaData: boolean;
Retrieves the metadata for the table specified by TTMSLCLCloudmyCloudData.TableId

function AddMetaData(AMetaData: TMyCloudDataMetaDataItem): boolean;
Sets the metadata for table specified by TTMSLCLCloudmyCloudData.TableId

function UpdateMetaData(AOldFieldName, ANewFieldName: string; ADataType: TFieldType = ftUnknown; ASize: integer = -1): boolean;
Modifies the meta data for a table specified by TTMSLCLCloudmyCloudData.TableId

function DeleteMetaData(APropertyName: string): boolean;
Delete a field from the meta data for a table specified by TTMSLCLCloudmyCloudData.TableId

Entity related methods:

function Insert(AValues: TStringList): TDatastoreEntity; override;
Inserts entity values via a stringlist

function Query: boolean; overload; override;
Retrieves entities for a table specified by TTMSLCLCloudmyCloudData.TableId

function Query(AFields: TStringList): boolean;
Retrieves entities with fields limitied to the specified list for a table specified by TTMSLCLCloudmyCloudData.TableId

function Query(AFields: TStringList; AFilters: TMyCloudDataFilters): boolean;
Retrieves entities with filter conditions

function Query(AFields: TStringList; ASortOrder: TMyCloudDataSortOrderList): boolean;
Retrieves entities with sort order specified

function Query(AFields: TStringList; AFilters: TMyCloudDataFilters; ASortOrder: TMyCloudDataSortOrderList): boolean; overload;
Retrieves entities with filter conditions and sort order specified

function Delete(AID: string): boolean; override;
Delete an entity with ID from a table specified by TTMSLCLCloudmyCloudData.TableId

function Update(AEntity: TDataStoreEntity): boolean; override;
Update the entity in a table specified by TTMSLCLCloudmyCloudData.TableId

function Download(ATableID, AEntityId: Int64; AFieldName: string; const TargetFile: string): boolean;
Download a blob to a file value from a specific entity in a specific table and fieldname

function Download(ATableID, AEntityId: Int64; AFieldName: string; AStream: TStream): boolean;
Download a blob value in a stream from a specific entity in a specific table and fieldname

function Upload(ATableID, AEntityId: Int64; AFieldName: string; FileName: string): boolean;
Upload a file to a blob field from a specific entity in a specific table and fieldname

function Upload(ATableID, AEntityId: Int64; AFieldName: string; AStream: TStream): boolean;
Upload a stream to a blob field from a specific entity in a specific table and fieldname

Users related methods:

function GetUsers: boolean;
Get list of users, for non-admin users, this retrieves the logged in user

function AddUser(AUser: TMyCloudDataUser): boolean;
Available for admin level users only

function DeleteUser(AID: int64): boolean;
Available for admin level users only

Additional properties:

property Users: TMyCloudDataUsers;
Collection of users, filled by the GetUsers method

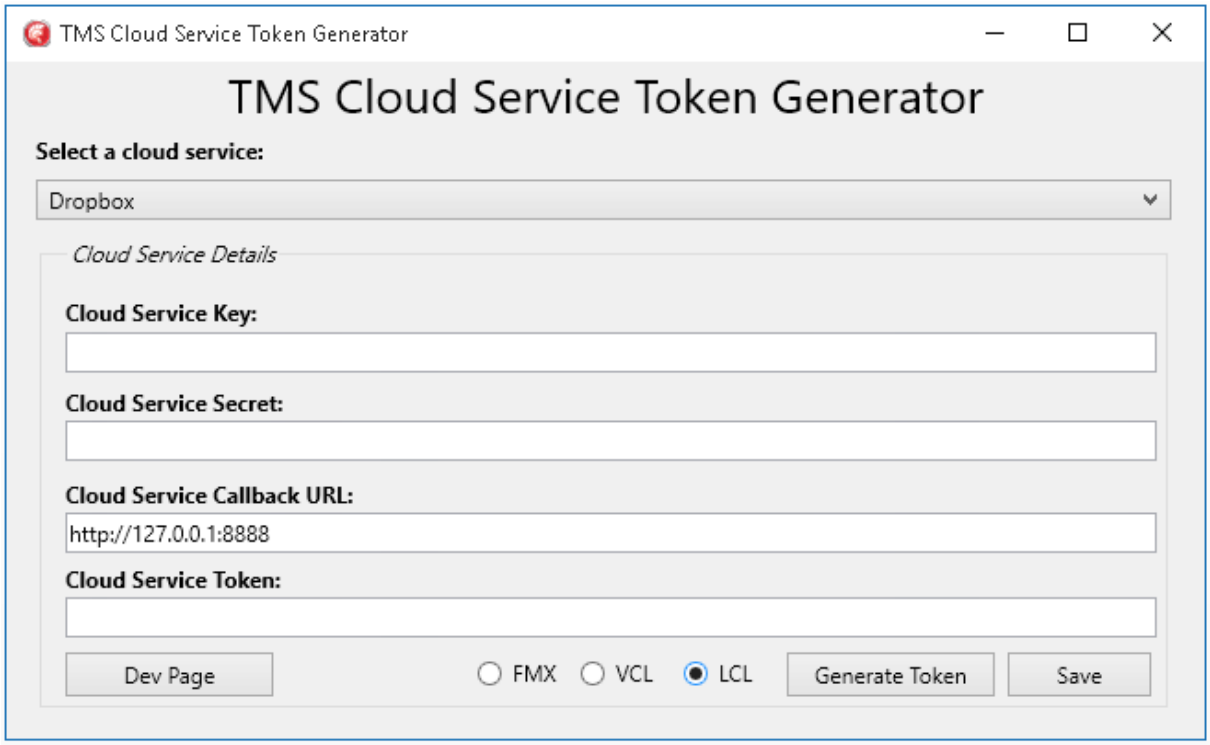
property Tables: TMyCloudDataTables;
Collection of users, filled by the GetTables method

property PageIndex: integer;
When > 0, this specifies the page of entities to retrieve for the Query() methods

property PageSize: integer;
When > 0, specifies the maximum number of entities to return for the Query() methods.
The last page is retrieved when number of entities is smaller than PageSize.

Authentication persistence

Token retrieval goes through the included token generator.



1) Choose your service
2) Insert the proper key, secret and callback url
3) Choose your platform (in this case LCL)
4) Press the “Generate Token” button
5) Complete the authorization steps
6) Press the “Save” button to save your token to the correct .ini file.

Internally, after authentication with the cloud service, the component has obtained an access token. This access 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

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
    // when no new access token can be obtained, do new authentication
  if not acc then
    // Regenerate a token with the token generator
end;
```

Tips and FAQ

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 access 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 TTMSLCLCloudSkyDrive scopes or remove from https://www.googleapis.com/auth/drive.file from the TTMSLCLCloudGDrive scopes.