TMS IntraWeb Cloud Pack
DEVELOPERS GUIDE

April 2017

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

Index

Availability

TMS IntraWeb Cloud Pack is a set of IntraWeb components for web application development and is available for Embarcadero Delphi & C++Builder XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo.

The IntraWeb framework 12.x, 14.0.x is required.

The TMS IntraWeb Cloud Pack contains components that offer integration with several cloud services.
The TMS IntraWeb Cloud Pack also contains a TTIWPayPalClient component that offers an easy way to integrate a Paypal based payments service in IntraWeb applications.

In this document you will find an overview of the components and their features, code snippets to quickly start using the component and an overview of properties, methods and events.

TMS IntraWeb Cloud Pack has been designed for and tested with: Widows 2008 server, Windows 7, Windows 8 and Windows 10.
Current version of TMS IntraWeb Cloud Pack has been designed for and tested with Microsoft Internet Explorer 9, Google Chrome 19, Firefox 5, Apple Safari for Mac OS-X, Apple Safari for iOS, Opera 11, WebKit

Online references

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

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

Purchase a license

The TMS Cloud Pack is separately available and also as part of the TMS IntraWeb Component Studio:

- TMS IntraWeb Cloud Pack: http://www.tmssoftware.com/site/tmsiwcloud.asp
- TMS IntraWeb Component Studio: http://www.tmssoftware.com/site/iwstudio.asp
- TMS Cloud Studio: http://www.tmssoftware.com/site/tmscloudstudio.asp

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

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

- TMS Cloud Pack for FireMonkey : http://www.tmssoftware.com/site/tmsfmxcloudpack.asp
- TMS LCL Cloud Pack : http://www.tmssoftware.com/site/tmslclcloudpack.asp
- TMS Cloud Pack for .NET : http://www.tmssoftware.com/site/tmscloudnet.asp

Terms of use

With the purchase of TMS 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, Microsoft Live Calendar, Microsoft Live Contacts, Facebook, Twitter, LinkedIn, PushOver, FourSquare, BitCasa, YouTube, Pryv service in Delphi applications and with this consulting and support comes the full source code needed to do this integration. As TMS Cloud Pack uses the Amazon Cloud Drive, Google GDrive, Microsoft SkyDrive, DropBox, Box, Flickr, Google Calendar, Google Contacts, Google Picasa, Microsoft Live Calendar, Microsoft Live Contacts, Facebook, Twitter, LinkedIn, FourSquare, BitCasa, YouTube, Pryv 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
https://m.box.com/static/html/terms.html
http://developer.linkedin.com/documents/linkedin-apis-terms-use
https://foursquare.com/legal/terms
https://www.bitcasa.com/legal
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

TMS software is not responsible for the use of TMS Cloud Pack components. The purchase of TMS Cloud Pack does not include any license fee that you might possibly be required to pay to Amazon, Google, Microsoft, DropBox, Box, Flickr, Facebook, Twitter, LinkedIn, FourSquare, BitCasa, YouTube, Pryv. 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, LinkedIn, FourSquare, BitCasa, YouTube, Pryv service to respect the Google, Microsoft, DropBox, Facebook, Twitter, LinkedIn, FourSquare, BitCasa, YouTube, Pryv terms and conditions. TMS software does not take any responsibility nor indemnifies any party violating the Google, Microsoft, DropBox, Box, Flickr, Facebook, Twitter, LinkedIn, FourSquare, BitCasa, YouTube, Pryv service terms & conditions.

Limited warranty

TMS software cannot guarantee the current or future operation & uptime of the Amazon, Google Drive, Microsoft SkyDrive, Microsoft Live Calendar, Microsoft Live Contacts, DropBox, Box, Flickr, Facebook, Twitter, Google Contacts, Google Calendar, Google Picasa, LinkedIn, FourSquare, BitCasa, YouTube, Pryv services.
TMS software offers the consulting and support for TMS Cloud Pack in good faith that the Amazon, Google Drive, Microsoft SkyDrive, Microsoft Live Calendar, Microsoft Live Contacts, DropBox, Box, Flickr, Facebook, Twitter, Google Contacts, Google Calendar, Google Picasa, LinkedIn, FourSquare, BitCasa, YouTube, Pryv 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 SkyDrive, Microsoft Live Calendar, Microsoft Live Contacts, DropBox, Box, Flickr, Facebook, Twitter, Google Contacts, Google Calendar, Google Picasa, LinkedIn, FourSquare, BitCasa, YouTube, Pryv service terms/operation changes or stops.

Main features

- Set of VCL components to offer easy access from IntraWeb 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 Google Drive storage
- Component to get access to Microsoft SkyDrive storage
- Component to get access to BitCasa 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 Tasks API
- Component to get access to YouTube 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 LinkedIn API
- Component to get access to FourSquare API
- Component to get access to DropBox DataStore API
- Component to get access to Pryv API
- Built-in support for OAuth 1.0 & 2.0 handling
- Built-in support for use of refresh tokens for use with one time authentication

Registering your application

A first step will be to register your application so you can obtain an application key and secret at the different cloud services. 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 TIWAdvOneDrive, TIWAdvDropBox, TIWAdvAmazonCloudDrive, TIWAdvBoxNetDrive, TIWAdvBitCasa, and TIWAdvGDrive components to access your cloud storage. All storage components work in a similar way:

1) Drop the component on the main form.

2) Add a TIWCloudServerController to the ServerController form. (See TTIWCloudServerController section for further information)

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

4) Assign the WebApplication and ServerController properties

5) Call the .DoAuth method.

Code:

```
  TIWAdvGDrive1.App.Key := 'xxxxxxxxx.apps.googleusercontent.com';

  TIWAdvGDrive1.App.Secret := 'yyyyyyyyyyyyyyy';

  TIWAdvGDrive1.WebApplication := WebApplication;

  TIWAdvGDrive1.ServerController :=
IWServerController.TIWCloudServerController1;

  TIWAdvGDrive1.DoAuth;
```

And this will show the Google login screen. When the user has provided the correct credentials, the browser will be returned to the IntraWeb application and the event OnReceivedAccessToken will be triggered. 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 TIWAdvXXXDrive.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.

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:

TIWAdvXXXDrive.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.

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

Public shared files

TIWAdvDropBox, TIWAdvOneDrive, TIWAdvBoxNetDrive, TIWAdvBoxNet 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.

Note that for TIWAdvGDrive, 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.

CloudStorage specific settings

TIWAdvDropBox 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 AdvDropBox.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.

TTIWCloudServerController

Description
The TTIWCloudServerController component has been designed as a non-visual helper component that enables the client controls to process the information retrieved from the external service websites.

Use
It is required that this control is present on the ServerController form of any IntraWeb application that will be using TMS IntraWeb Cloud controls.
From the Delphi repository select a new IntraWeb application template.

This is done by choosing: File > New > Other > IntraWeb > IntraWeb Application Wizard.

- Open the ServerController form and drop a TTIWCloudServerController on it.
  Assign the OnBeforeDispatch event and add a call to ProcessRequest.

```
TIWCloudServerController1.ProcessRequest(Request.QueryFields);
```

Methods
- **procedure ProcessRequest(QueryFields: TStrings);**

  This method passes the Request parameters received from the external service website to the appropriate client control.

TIWAdvTwitter

Usage

TIWAdvTwitter 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:

```
TIWAdvTwitter1.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:

```
TIWAdvTwitter1.GetFollowers;
TIWAdvTwitter1.GetProfileListInfo(TIWAdvTwitter1.Followers);
```

TIWAdvFacebook

Usage

TIWAdvFacebook 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 or av21 for v2.1).

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.

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.

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.

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 := AdvFacebook1.PostImage('Description', OpenDialog1.FileName);
ImageURL := AdvFacebook1.GetImageURL(ImageID);
TIWAdvFacebook1.Post('Message', 'http://www.mywebsite.com', ImageURL);
```

TIWAdvFlickr

Usage

TIWAdvFlickr 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:
IWAdvFlickr1.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:
IWAdvFlickr1.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:
IWAdvFlickr1.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 := AdvFlickr1.UploadPhotoToStream('C:\folder\image1.jpg', 'new
image', 'image description');
  if Assigned(APhoto) then
    IWAdvFlickr1.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.

```
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 TIWAdvFlickr.

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.

TIWAdvFourSquare

Usage

TIWAdvFourSquare 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.

TIWAdvGCalendar

Usage

TIWAdvGCalendar 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.
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';
TIWAdvGCalendar1.Update(ci);
```

Delete(Item: TGCalendarItem);
Delete an item.

TIWAdvGContacts

Usage

TIWAdvGContacts 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;
TIWAdvGContacts1.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.

TIWAdvGPlaces

Usage

TIWAdvGPlaces 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

TIWAdvGTasks

Usage

TIWAdvGTasks 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

TIWAdvPicasa

Usage

TIWAdvPicasa 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 AdvPicasa.Albums

Album[].GetPhotos: TPicasaPhotos;
Fill the list of Picasa photos assigned to the album, accessible via AdvPicasa.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 := AdvPicasa1.Albums.Add;
  Album.Title := edAlbumTitle.Text;
```

```
Album.Summary := edAlbumDescription.Text;
TIWAdvPicasa1.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
  TIWAdvPicasa1.UploadPhoto(AdvPicasa1.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 AdvPicasa.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.

TIWAdvYouTube

Usage

TIWAdvYouTube 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 AdvYouTube.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 AdvYouTube.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 AdvYouTube.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.

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

Example:

```
  if OpenDialog1.Execute then
  begin
     TIWAdvYouTube1.UploadVideo(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.

TIWAdvLinkedIn

Usage

TIWAdvLinkedIn 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 := TIWAdvLinkedIn1.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:

```
  TIWAdvLinkedIn1.Share(edTitle.Text, edDescription.Text, edHyperlink.Text,
edImageLink.Text);
```

TIWAdvLiveCalendar

Usage

TIWAdvLiveCalendar 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';
TIWAdvLiveCalendar1.Update(ci);
```

Delete(Item: TLiveCalendarItem);
Delete an event.

TIWAdvLiveContacts

Usage

TIWAdvLiveContacts 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;
TIWAdvLiveContacts1.Add(ci);
```

TIWAdvDropBoxDataStore

Usage

TIWAdvDropBoxDataStore is a component that provides access to the DropBox DataStore service. The component supports creating and deleting datastores as well as reading, inserting, updating and deleting records. Note that the DropBox datastore is a loosely typed not relational recordset. This means that you can insert in a record any set of name/value pairs.

Properties

The **TIWAdvDropBoxDataStore** has following public properties:

DataStoreList: TDataStoreList; The collection of datastores
LastError: string; The description of the last error that occurred

The **TDataStore** has following properties:

ID: string; The datastore id
DataStoreName: string; The name of the datastore
DateTime: TDateTime; The timestamp of the datastore
Revision: integer; The revision number of the datastore
Rows: TDataStoreRows; The collection of rows that belong to the datastore
Title: string; The title of the datastore

The **TDataStoreRow** has following properties:

ID: string; The row id
TableName: string; The name of the table this row belongs to
Fields: TDataStoreFields; The collection of fields that belong to the row

The **TDataStoreField** has following properties:

FieldName: string; The name of the field
Value: TValue; The value of the field
Tag: TTagInt; The tag value of the field

Methods

CreateDataStore(DataStoreName: string): TDataStore;
Creates a new datastore with the name defined by the DataStoreName parameter.

DeleteDataStore(DataStore: TDataStore);
Deletes an existing datastore.

GetDataStores
Fills the DataStoreList collection with the existing datastores of the currently authenticated user.

GetDataStoreByName(DataStoreName: string): TDataStore;
Returns the datastore with the matching DataStoreName value.

DataStoreList[].SetMetaData(Title: string; DateTime: TDateTime);
Set the meta data for the datastore.

DatatStoreList[].GetRecords: integer;
Fills the Rows collection with records that belong to the datastore. Returns the number of rows in the collection.

DataStoreList[].GetNewID: string;
Returns a unique ID for creating a new TDataStoreRow.

DataStoreList[].InsertRecord(Row: TDataStoreRow): Boolean;
Inserts a new row in a datastore. Returns true if the operation succeeded, false otherwise. When an error occurred, it can be retrieved via public read-only property LastError.

DataStoreList[].DeleteRecord(Row: TDataStoreRow): Boolean;
Deletes an existing row from a datastore. Returns true if the operation succeeded, false otherwise. When an error occurred, it can be retrieved via public read-only property LastError.

DataStoreList[].UpdateRecord(Row: TDataStoreRow): Boolean;
Updates an existing row in a datastore. Returns true if the operation succeeded, false otherwise. When an error occurred, it can be retrieved via public read-only property LastError.

DataStoreList[].DeleteFields(Row: TDataStoreRow; FieldNames: TStringList): Boolean;
Deletes one or more fields (defined in the FieldNames parameter) from a row. Returns true if the operation succeeded, false otherwise.

DataStoreList[].Rows.Find(ID: string): TDataStoreRow;
Returns the row with a matching ID value.

DataStoreList[].Rows[].AddData(AName: string; AValue: string);
Adds a new field to a datastore row.

DataStoreList[].Rows[].Fields.AddPair(AFieldName: string; AValue: string): TDataStoreFIeld;
DataStoreList[].Rows[].Fields.AddPair(AFieldName: string; AValue: integer): TDataStoreFIeld;
DataStoreList[].Rows[].Fields.AddPair(AFieldName: string; AValue: double): TDataStoreFIeld;
DataStoreList[].Rows[].Fields.AddPair(AFieldName: string; AValue: boolean): TDataStoreFIeld;
Adds a new field to a datastore row based on the AFieldName and AValue parameter values. The AValue can be a string, integer, double or boolean.

Sample

This code snippet shows how a datastore with a 2 row recordset can be programmatically created on the DropBox datastore:

```
var
  ds: TDataStore;
  dr: TDataStoreRow;


begin
  ds := TIWAdvDropBoxDataStore1.CreateDataStore('demo');

  dr := ds.Rows.Add;
  dr.ID := ds.GetNewID;
  dr.TableName := 'Capitals';

  dr.Fields.AddPair('Name','Paris');
  dr.Fields.AddPair('Country','France');
  dr.Fields.AddPair('Citizens',10000000);
  dr.Fields.AddPair('Europe',true);
  ds.InsertRecord(dr);

  dr := ds.Rows.Add;
  dr.ID := ds.GetNewID;
  dr.TableName := 'Capitals';

  dr.Fields.AddPair('Name','Brussels');
  dr.Fields.AddPair('Country','Belgium');
  dr.Fields.AddPair('Citizens',3000000);
  dr.Fields.AddPair('Europe',true);

  ds.InsertRecord(dr);
end;
```

TTIWPayPalClient

Description
The TTIWPayPalClient component has been designed as a non-visual component that enables making payments through PayPal.

Features

- Make payments using an existing PayPal account

- Make payments via credit card using the PayPal service

Use
From the Delphi repository select a new IntraWeb application template.

This is done by choosing: File -> New ->Other-> VCL for the Web Application Wizard.

- Open the ServerController form and drop a TTIWCloudServerController on it.
  Assign the OnBeforeDispatch event and add a call to ProcessRequest.

```
TIWCloudServerController1.ProcessRequest(Request.QueryFields);
```

- From the component palette, select a TTIWPayPalClient component and drop it on a form.
  Then assign the following properties:
  - ServerController
  - Seller.UserName
  - Seller.Password
  - Seller.Signature
  (obtained after requestion API credentials on the PayPal website)
  When using the PayDirect method you also need to set the Buyer and CreditCard properties.

```
TIWPayPalClient1.ServerController :=
IWServerController.TIWCloudServerController1;
TIWPayPalClient1.Seller.UserName := APIusername;
TIWPayPalClient1.Seller.Password := APIpassword;
TIWPayPalClient1.Seller.Signature := APIsignature;
with TIWPayPalClient1.Transactions.Add do
begin
  Description := 'Description';
  Name := 'Name';
  Number := 1;
  Price := 5.00;
  Quantity := '1';
end;
```

- Now the control is configured and ready to initiate a payment transaction.

```
TIWPayPalClient1.PayOnline(WebApplication);
```
or
```
TIWPayPalClient1.PayDirect(WebApplication);
```

Properties
- **Buyer:** Set the buyer details (only required when using the PayDirect method)

- **CreditCard:** Set the buyer's credit card details (only required when using the PayDirect method)
- **Seller:** Set the password, signature and username of the seller app that is used to authenticate the buyer on the PayPal website
- **ServerType:** Switch between **stSandbox** (use the PayPal sandbox server for testing purposes) or **stLive** (use the Live PayPal server for production use)
- **Transaction.Currency:** Set the currency of the transaction
- **Transaction.Insurance:** Set the insurance costs of the transaction (optional)
- **Transaction.Shipping:** Set the shipping costs of the transaction (optional)
- **Transaction.Tax:** Set the tax costs of the transaction (optional)
- **Transactions:** A collection of the items in the transaction

Methods

- **procedure PayOnline(WebApplication: IWApplication);**

  Initiates the payment transaction based on the settings from the Seller properties. The login screen from the external service is displayed. After logging in and confirming the payment, the browser returns to the IntraWeb application.

- **procedure PayDirect(WebApplication: IWApplication);**

  Initiates the payment transaction based on the settings from the Seller, Buyer and CreditCard properties. There is no PayPal login required.

Events

- **OnPayPalCancel:** Event triggered when the payment transaction was cancelled
- **OnPayPalFailed:** Event triggered when the payment transaction was not successful
- **OnPayPalPaymentOk:** Event triggered when the payment transaction was successful

TIWAdvPryv

Usage

TIWAdvPryv 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 **TIWAdvPryv** 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;
   TIWAdvPryv1.Events.Add(val);
   val.Content := '10';
   val.StreamID := TIWAdvPryv1.Streams[0].ID;
   val.UnitValue := 'mymeasurement/myunit';
   val.DateTime := Now;
   val.Tags.CommaText := 'custom, value';
   val.Description := 'my custom value';
   TIWAdvPryv1.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 := TIWAdvPryv1.Streams.Add;
  it.Summary := edStreamName.Text;
  TIWAdvPryv1.AddStream(it);

  pos := TPryvPosition.Create;
  TIWAdvPryv1.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';
  TIWAdvPryv1.AddEvent(pos);
```

Authentication persistence

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

Test if the access token is still accepted

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

Load the tokens from an INI file or registry key

The component has methods:

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

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

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

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

and the property:

TokensAsString: string; encrypted string holding all token values

A typical flow is:

```
var
  acc: boolean;

if Storage.App.Key <> '' then
begin
  // load tokens from registry
  Storage.LoadTokens;
  // test if the access token is still accepted
  acc := Storage.TestTokens;
  // when not, try to get a new access token with the refresh token
  if not acc then
    acc := Storage.RefreshAccess;
  // when no new access token can be obtained, do new authentication
  if not acc then
    Storage.DoAuth
end
```

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

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

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

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

Tips and FAQ

Logging

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

Scopes

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

For Microsoft OneDrive, 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 OneDrive files
wl.skydrive_update : write access to user OneDrive 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 OneDrive or Google Drive, remove the specifiers wl.skydrive_update from the TIWAdvSkyDrive scopes or remove from https://www.googleapis.com/auth/drive.file from the TIWAdvGDrive scopes.