TMS IntraWeb Cloud Pack
DEVELOPERS GUIDE

October 2012

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

Table of contents

Introduction

The TMS IntraWeb Cloud Pack contains components that offer integration with several cloud services.
A first set of components makes the integration of authentication for IntraWeb apps via services (OAuth) of Google, Facebook, Twitter and Windows Live easy.
Also included in the client components is basic support for posting a status update on Facebook or posting a tweet on Twitter.

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.

Availability

TMS IntraWeb Cloud Pack is available as VCL components for Delphi.

TMS IntraWeb Cloud Pack is available for Delphi XE, XE2, XE3.

The IntraWeb framework 11.0, 12.0, 12.1, 12.2, 14.0 is required.

TMS IntraWeb Cloud Pack has been designed for and tested with: Widows 2008 server & Windows 7. 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.

List of included components

TTIWAuthServerController: servercontroller component

TTIWFacebookAuthClient: client component for Facebook authentication

TTIWGAuthClient: client component for Google authentication

TTIWPayPalClient: client component for PayPal payment transactions

TTIWTwitterAuthClient: client component for Twitter authentication

TTIWWindowsLiveAuthClient: client component for WindowsLive authentication

Online references

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

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

TMS IntraWeb products page:
http://www.tmssoftware.com/site/products.asp?t=iw

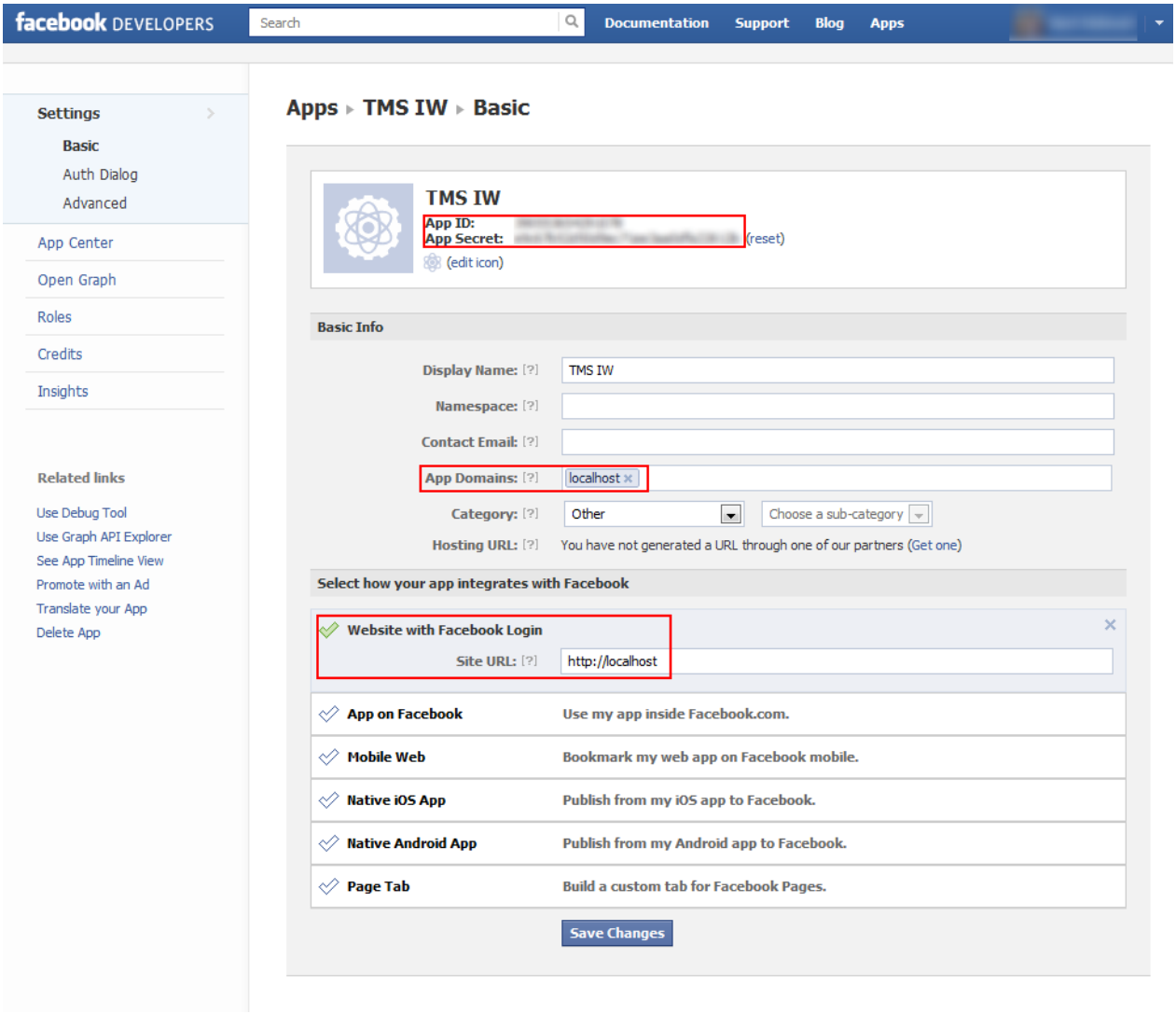
The oAuth protocol website:
http://oauth.net/

Creating API keys

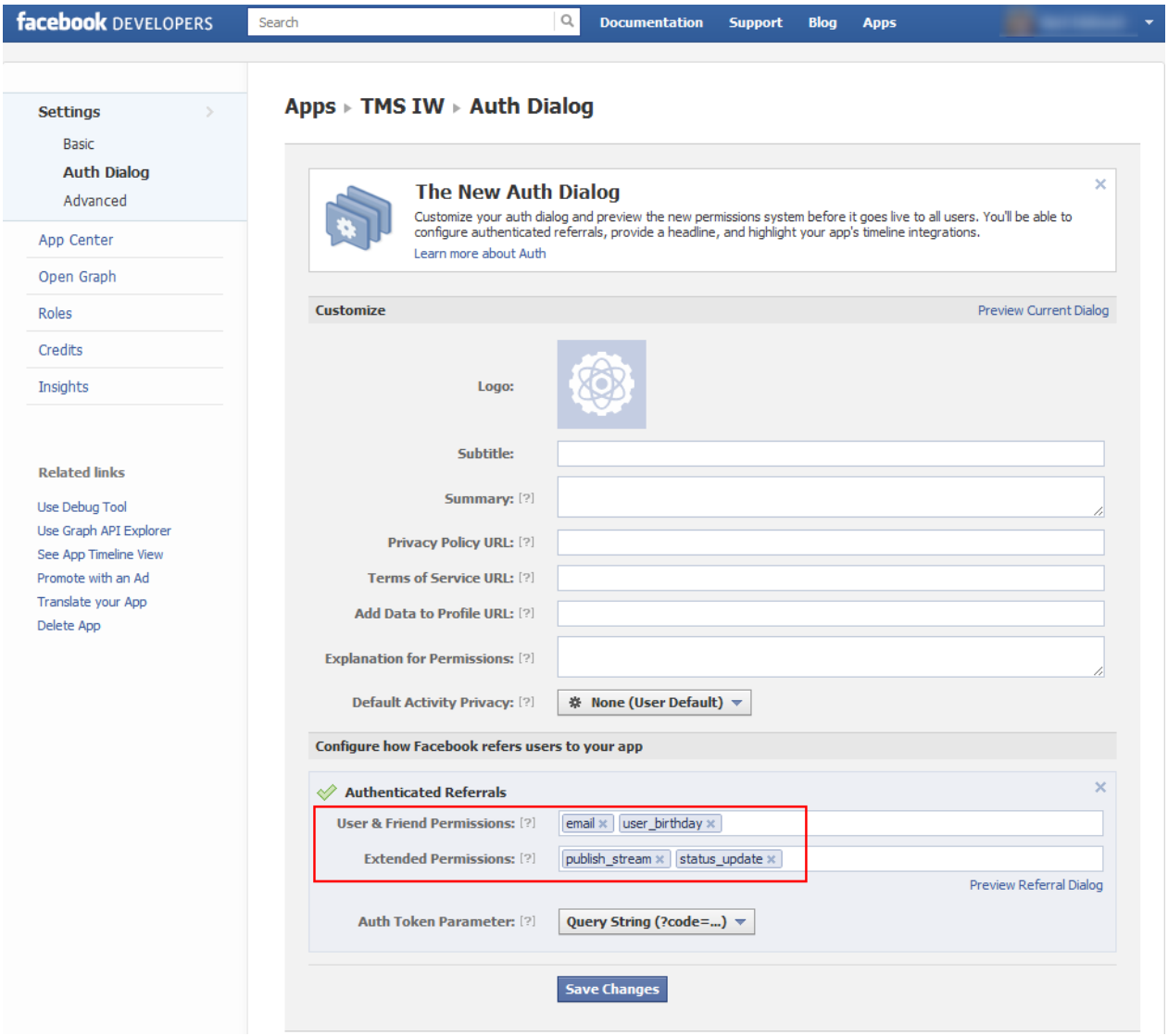
In order to enable authentication through an external website, the web application must first be registered with the service.
Below is a basic explanation of how to register an application and create the necessary API keys for each service:

Facebook

- Go to http://developers.facebook.com
- Login with your Facebook account
- Go to “Apps”
- Click “+ Create New App”
- Enter a name for your App and hit Continue
- Now the Basic settings page is displayed with the App ID and App Secret
- You still need to enter the appropriate values in the “App Domains” and “Site URL” fields

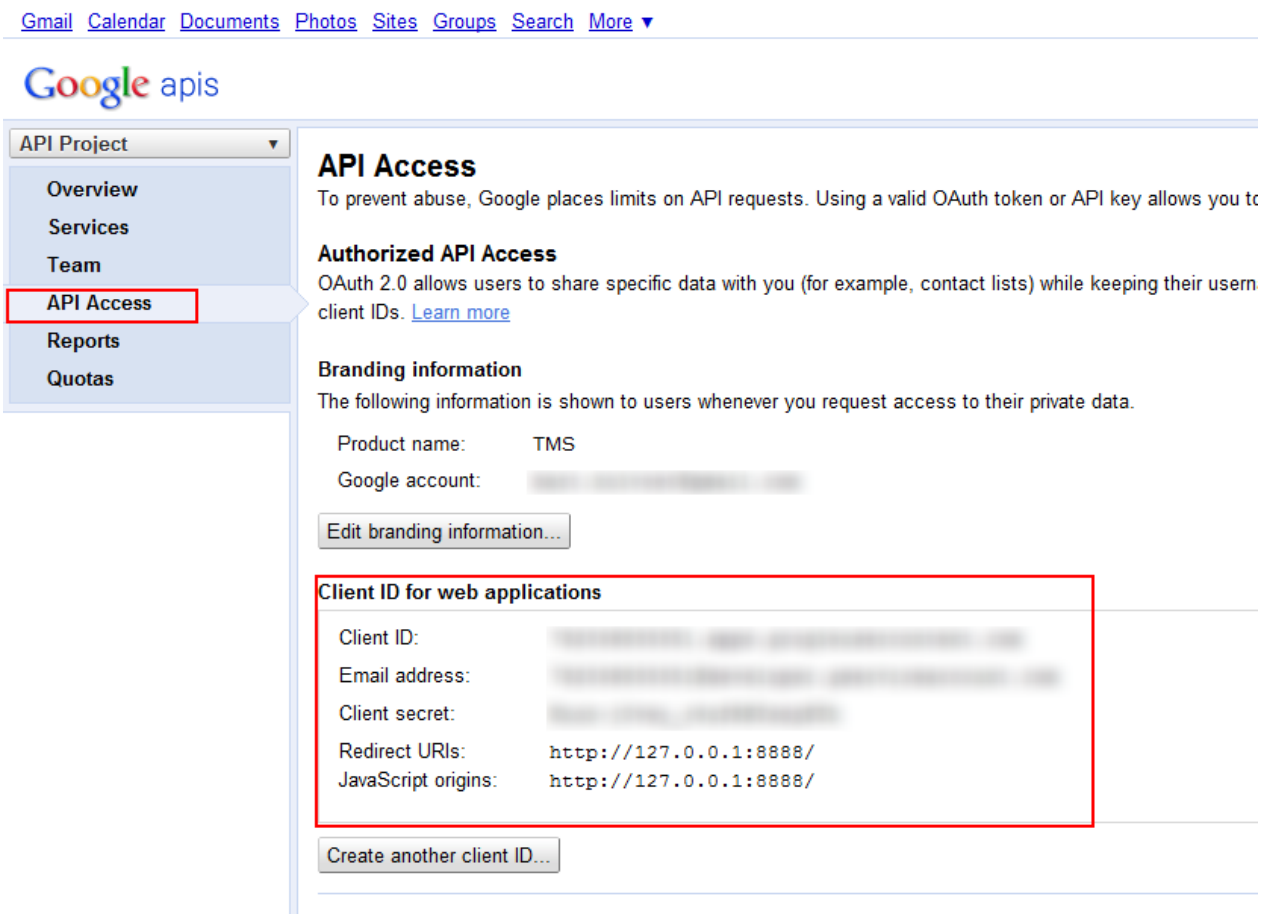


- Now go to “Auth Dialog” settings.
- Enable retrieval of the user’s email and birthday (based on the Scopes settings of your TIWFacebookAuthClient control) by entering the appropriate values (email, user_birthday) in the “User & Friend Permissions” field
- Enable posting a status update on the user’s Facebook page by entering the appropriate values (publish_stream, status_update) in the “Extended Permissions” field



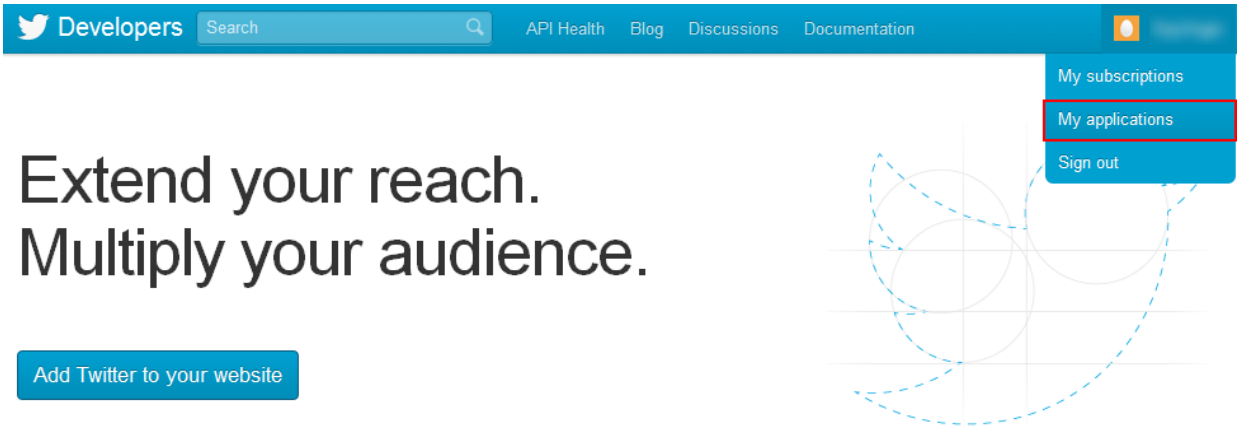
Google

- Go to http://code.google.com/apis/console
- Login with your Google account
- Create a new API Project
- Select the "API Access" tab and fill in the necessary information
- The App ID and App Secret are displayed on the following page

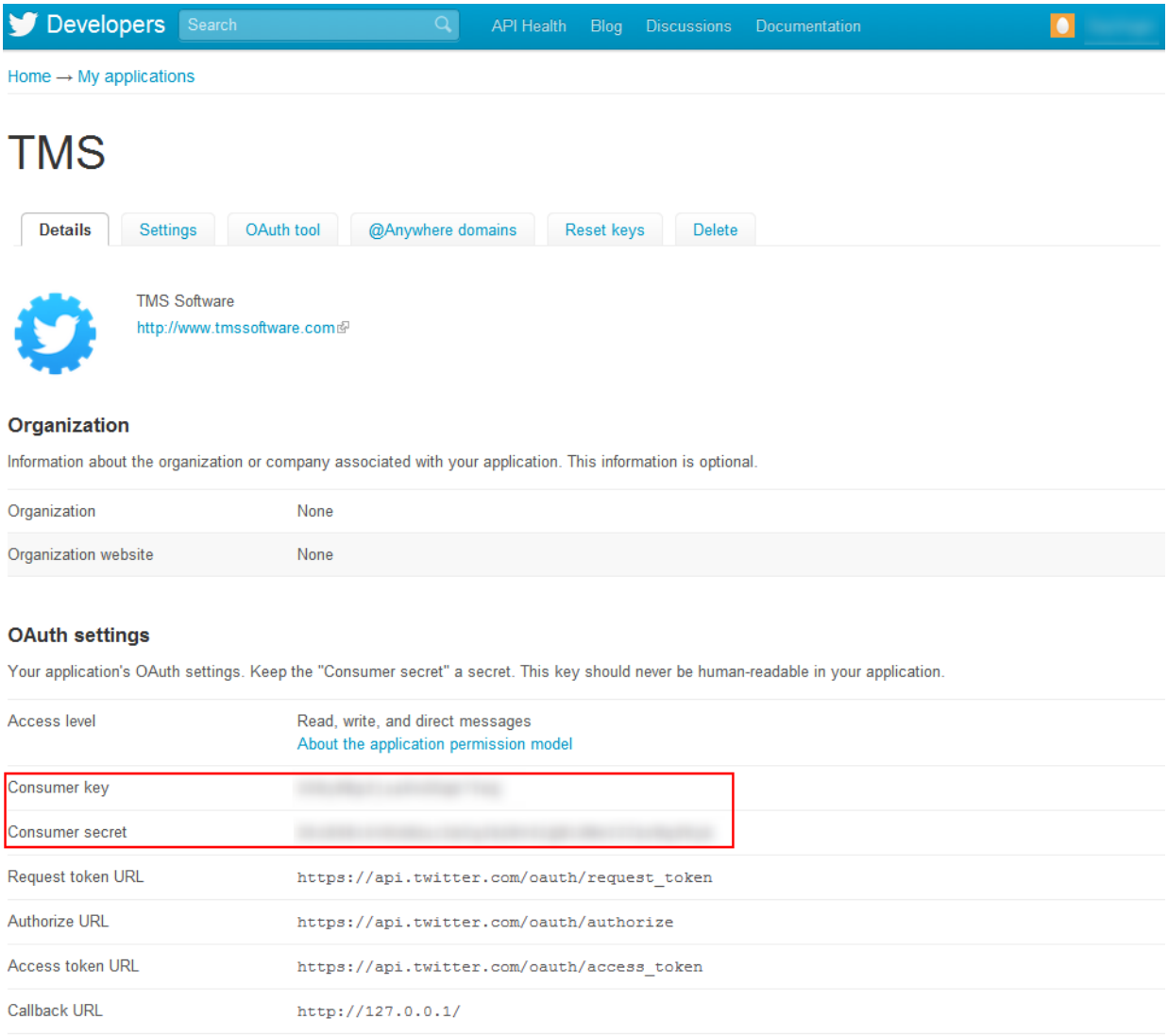


Twitter

- Go to http://dev.twitter.com
- Login with your Twitter account
- Click on your Twitter ID in the top right corner and select “My Applications”

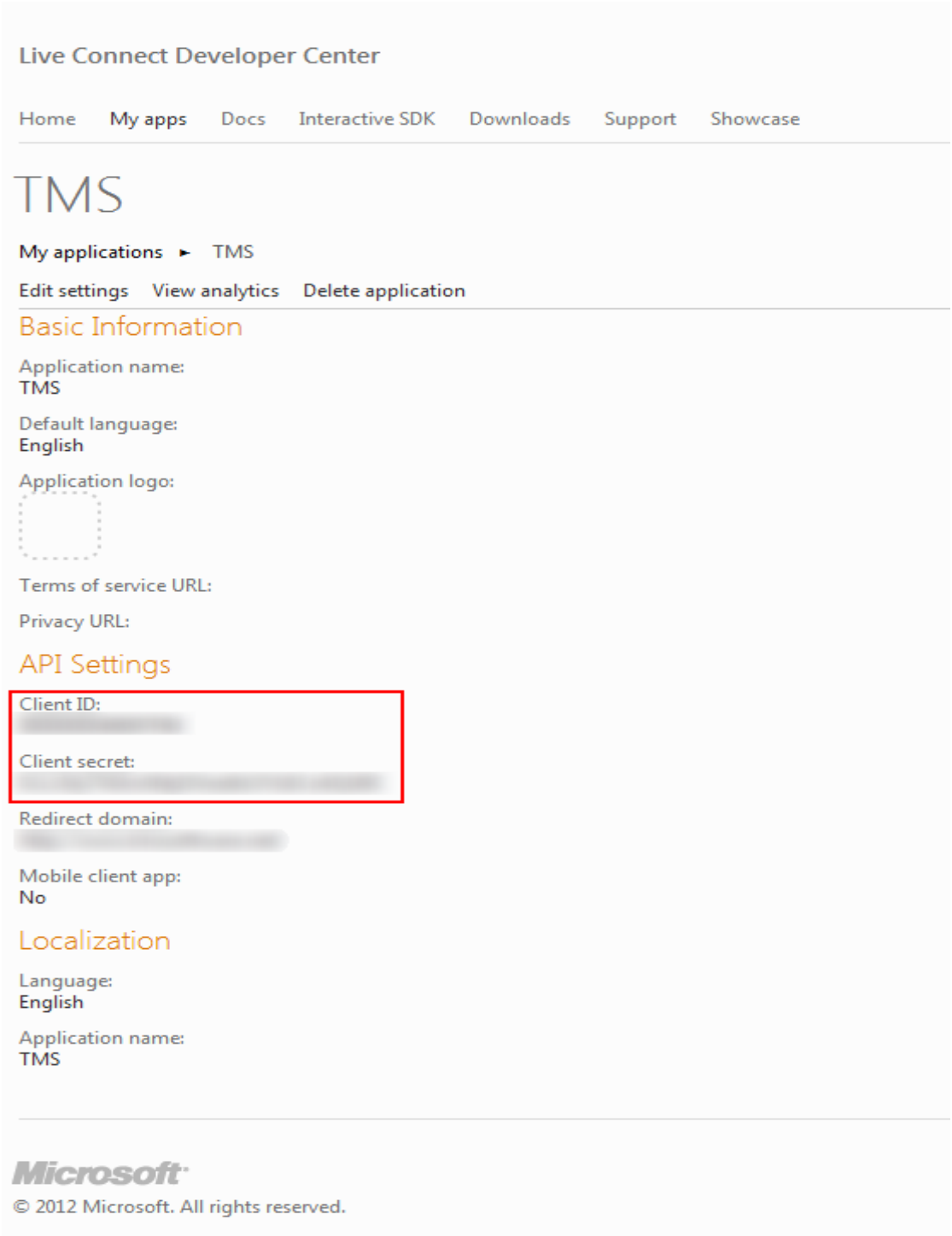


- On the “My Applications” screen click “Create a new application” and enter the required information. The API credentials are displayed on the following page


Developers
Search
API Health
Blog
Discussions
Documentation
Home → My applications
TMS
Details
Settings
OAuth tool
@Anywhere domains
Reset keys
Delete
TMS Software
http://www.tmssoftware.com
Organization
Information about the organization or company associated with your application. This information is optional.
Organization
None
Organization website
None
OAuth settings
Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should never be human-readable in your application.
Access level
Read, write, and direct messages
About the application permission model
Consumer key
Consumer secret
Request token URL
https://api.twitter.com/oauth/request_token
Authorize URL
https://api.twitter.com/oauth/authorize
Access token URL
https://api.twitter.com/oauth/access_token
Callback URL
http://127.0.0.1/

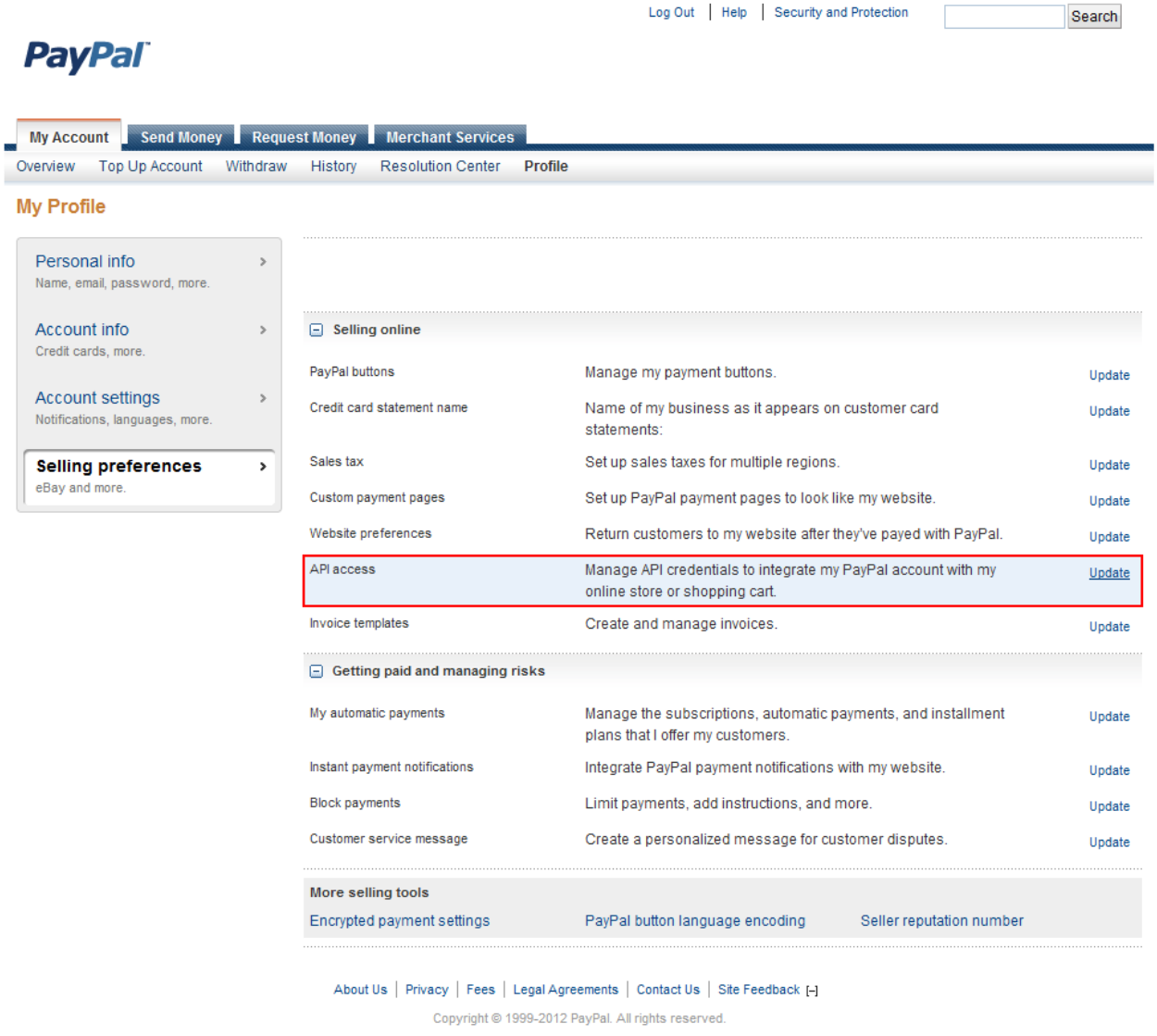
WindowsLive

- Go to http://dev.live.com
- Click on “My Apps” and login with your WindowsLive account
- Click on “Create application” and enter the required information
- The API credentials are displayed on the following page



PayPal

- Go to http://www.paypal.com
- Login with your PayPal account
- Go to "Profile" under the "My Account" tab
- Go to "Selling preferences"
- Click on "update" next to "API access"
- The API credentials are displayed the following page



TTIWFacebookAuthClient, TTIWGAuthClient, TTIWTWitterAuthClient, TTIWWindowsLiveAuthClient

Description

The TTIW*AuthClient components have been designed as non-visual components that allow the use of the OAuth protocol through a trusted external service.
The TTIWFacebookAuthClient and TTIWTwitterAuthClient controls also have basic support for authorization to post a status update on Facebook or post a tweet on Twitter.

Features

- Site login authentication through an external service
- Obtain basic user profile information
- Post a status message on the external service *

Architecture

The internal architecture of the TTIW*AuthClient controls is based on the OAuth protocol.
OAuth is an open protocol to allow secure API authorization in a simple and standard method from web applications.

More information about the OAuth protocol can be found at http://oauth.net/

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 TTIWAuthServerController on it.
  Assign the OnBeforeDispatch event and add a call to ProcessRequest.

```
TIWAuthServerController1.ProcessRequest(Request);
```

- From the component palette, select a TTIW*AuthClient component and drop it on a form.
  Then assign the following properties:
  - App.ID (obtained after creating an app on the external service website)

- App.Secret (obtained after creating an app on the external service website)
- ServerController (assign the TTIWAuthServerController control that was placed on the ServerController form)

```
TIWGAuthClient1.ServerController :=
IWServerController.TIWAuthServerController1;
TIWGAuthClient1.App.ID := 'xyz123';
TIWGAuthClient1.App.Secret := 'xyz123';
```

- Now the control is configured and ready to call the Login method.

```
TIWGAuthClient1.Login(WebApplication);
```

- If the login was successful the OnLoginOK event gets fired and the UserName and ID can be obtained from the LoginUserName and LoginID properties.

```
procedure TIWForm9.TIWGAuthClient1LoginOK(Sender: TObject);
begin
  IWLabel1.Text := 'The ID of ' + TIWGAuthClient1.LoginUserName + '
is ' + TIWGAuthClient1.LoginID;
end;
```

- If the login was not successful the OnLoginFailed event gets fired. If an error message is available it will be stored in the LoginError message.

```
procedure TIWForm9.TIWGAuthClient1LoginFailed(Sender: TObject);
begin
  IWLabel1.text := 'Login failed: ' + TIWGAuthClient1.LoginError;
end;
```

Properties

- **App.ID:** Set the ID key of the app that is used to authenticate on the external service
- **App.Secret:** Set the Secret key of the app that is used to authenticate on the external service
- **Scopes:** Optionally select any required items from the set of available scopes. The number of available scopes is relative to the IW*AuthClient control. See the documentation of the service for the list of available scopes.
- **ServerController:** Assign the TTIWAuthServerController control that was placed on the ServerController form
- **StandaloneRedirectURL:** Set the custom redirect URL if the external service does not accept localhost as redirect URL. (only for TTIWWindowsLiveAuthClient used in a Standalone IW app)

- **User:** Any additional user information (based on the set of selected scopes) that was obtained while logging in is stored in the User sub-properties

Methods

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

  Initiates the login sequence based on the settings from the App and Scopes properties. The login screen from the external service is displayed and after logging in the browser returns to the IntraWeb application.

- **procedure Post(S: String); ***

  Performs a post on the external service based on the settings retrieved from logging in to the service.

Events

- **OnLoginFailed:** Event triggered when the login action has failed after calling the Login procedure
- **OnLoginOK:** Event triggered when the login action has succeeded after calling the Login procedure
- **OnPostFailed(ErrorMessage: String) *:** Event triggered when the post action has failed after calling the Post procedure
- **OnPostOK(PostID: String) *:** Event triggered when the post action has succeeded after calling the Post procedure

*Only supported for TTIWFacebookAuthClient and TTIWTwitterAuthClient. The Windows Live service has no stream to post on and the Google+ stream is not accessible by an API for posting.

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 TTIWAuthServerController on it.
  Assign the OnBeforeDispatch event and add a call to ProcessRequest.

```
TIWAuthServerController1.ProcessRequest(Request);
```

- From the component palette, select a TTIWPayPalClient component and drop it on a form.
  Then assign the following properties:
  - 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.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.

```
TIWGAuthClient1.PayOnline(WebApplication);
```
or
```
TIWGAuthClient1.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

TTIWAuthServerController

Description

The TTIWAuthServerController 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

- See the “Use” topic under the “TTIWFacebookAuthClient, TTIWGAuthClient, TTIWTwitterAuthClient, TTIWWindowsLiveAuthClient” or “TTIWPayPalClient” chapters.

Methods

- **procedure ProcessRequest(Request: TWebRequest);**

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