

**How to consume SAPO SOAP Services from a Windows 8 App**

**v1.0**

Contents

[1 Introduction 3](#_Toc338432218)

[1.1 Why should I read this 3](#_Toc338432219)

[1.2 Before you can use a service 3](#_Toc338432220)

[1.3 Other relevant notes 3](#_Toc338432221)

[2 Authentication 3](#_Toc338432222)

[3 How to guide 4](#_Toc338432223)

[4 Code Samples 5](#_Toc338432224)

# Introduction

The present document describes how to consume a SAPO SOAP Web Service from a Windows Store C# application.

## Why should I read this

If you want to use SAPO SOAP Web Services you should read this because there are some issues with the generated proxy from “add service reference” in Visual Studio 2012.

In this document are presented the issues and the solutions for this issues.

## Before you can use a service

Before you can use a service you must create an account at <https://store.services.sapo.pt/>. After creating the account you can acquire services subscriptions. In the customer area you can manage you services and generate authorization information, i.e. an access key.

## Other relevant notes

If you plan to consume a SAPO SOAP Web Service from a **HTML5+Javascript Windows Store application** you should consider using the HTTP interface rather than the SOAP interface. The SAPO Services Windows 8 SDK includes a set of **NuGet** packages that have the code to interact with the service. For each service you can find working samples in the following URI: <https://github.com/sapo/sapo-services-sdk/tree/master/Code/Windows8/HTML5+Javascript/HTTP/JavascriptHTTP>

It’s not recommend to create a **WinRT** component to consume the SAPO SOAP services in **Windows Store applications**. See following URI to get more information: <http://social.msdn.microsoft.com/Forums/en-US/winappswithcsharp/thread/676f2908-48a7-413e-9340-caa22956363e>

# Authentication

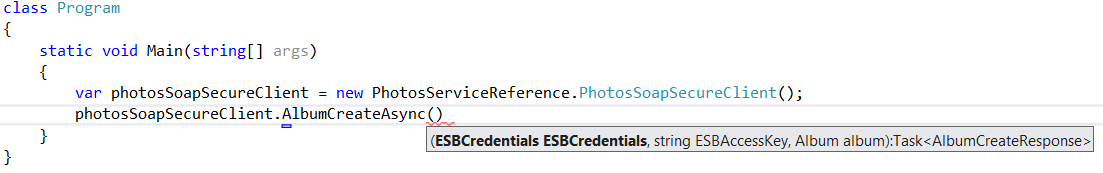
Every request must contain authentication information to establish the identity of whom making the request and an authorization token denominated **ESBAccessKey**. Note that you can generate this authorization tokens that only can be used by a specific user or that can be used by all users (good for a multi-users application).

The authentication information can be provided as a pair of username (**ESBUsername**) and password (**ESBPassword**) or an authentication token (**ESBToken**) obtained from the **SAPO Security Token Service** (STS) <https://store.services.sapo.pt/en/Catalog/development/free-api-sts>.

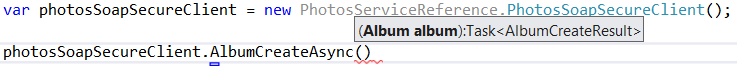
If you don't want to constantly send passwords "over the wire" you should use STS. This way you only send the credentials once.

# How to guide

Using a SAPO SOAP Web Service in **XAML+C# Windows Store applications** shouldn’t be different from using other SOAP service, but for some reason the proxy isn’t generated as expected, *i.e.* every method should receive as parameter an instance of **ESBCredentials** and a string with the **ESBAccessKey** like in the following example, but the generated proxy (with the “Add Service Reference” Visual Studio functionality) don’t include this two parameters in the methods that do the service calls.

The following example shows how the proxy should be generated:

For the same service operation, *i.e.* **AlbumCreate**, the corresponding method just receives an instance of **Album** in the proxy generated with “Add Service Reference” in a **Windows Store App** project (see the following example).



The SAPO services requires the authentication data (the pair **ESBUsername** and **ESBPassword** or the **ESBToken** and the **ESBAccessKey**) in every request. In the SAPO SOAP Web Services this authentication data must be provided in the SOAP message headers.

Because the proxy is not generated as expected you have to ensure yourself that the message headers have your authentication data. You can do this using the **OperationContextScope** type.

In order to abstract the developer of this tasks, it was created a class that implements **IDisposable** that encapsulates this logic named **EnsureCredentialsUseContext**. With this class you cannot use the Visual Studio generated **ESBCredentials** class. The new **EsbCredentials** class and the **EnsureCredentialsUseContext** class are available at: <https://github.com/sapo/sapo-services-sdk/tree/master/Code/Windows8/XAML+CSharp/SOAP/SDK_Code/Utils>

The following example is a sample usage of the EnsureCredentialsUseContext class:

string u = "username";

string pw = "password";

string ak = "accessKey";

var c = new PhotosSoapSecureClient();

string[] tags;

using (var context = new EnsureCredentialsUseContext(u, pw, ak, c))

{

tags = await this.\_client

.UserGetTagsAsync(null, null)

.ContinueWith(t => t.Result.tags);

}

To use the **EnsureCredentialsUseContext** you just have to create an instance of this class before the service call and after call the dispose. You can also use the C# using clause instead of explicitly calling the dispose method, as show in the example above.

Apart from issue, the service will work as any other SOAP Service in C#.

# Code Samples

The SAPO Services Windows 8 SDK includes examples of the use of the SAPO SOAP Web Services. See the following URI for examples of the Photos and Videos services: <https://github.com/sapo/sapo-services-sdk/tree/master/Code/Windows8/XAML+CSharp/SOAP/CSharpSOAP_ProxyInAppProj>