



SAPO Services SDK Documentation

v0.1

This page was intentionally left in blank

Contents

1	Introduction	1
2	How to use a Service	3
3	Authentication	5
3.1	HTTP Clients	5
3.1.1	Using username and password	5
3.1.2	Using authentication token	6
3.2	SOAP Clients	6
4	Photos Service	7
4.1	HTTP	7
4.1.1	ImageCreate	7
4.1.2	ImageGetListBySearch	10
4.1.3	ImageGetListBySearch	11
5	Support and contacts	13
6	Glossary	15
	References	17

This page was intentionally left in blank

List of Figures

This page was intentionally left in blank

List of Tables

This page was intentionally left in blank

Listings

3.1	HTTP GET using ESBUusername and ESBCPassword	5
3.2	HTTP GET using ESBCToken	6
4.1	POST the meta data of the photo	7
4.2	Response of the HTTP POST with the meta data	8
4.3	HTTP GET using ESBCToken	9
4.4	Photo file upload response body	9
4.5	Response body in case of invalid token	10
4.6	ImageGetListBySearch sample request	10
4.7	ImageGetListBySearch sample response body	10

This page was intentionally left in blank

Introduction

The present document describes the SAPO Services Windows 8 SDK.

This page was intentionally left in blank

2

How to use a Service

This page was intentionally left in blank

Authentication

Every request must contain authentication information to establish the identity of whom making the request and a authorization token denominated `ESBAccessKey`.

The authentication information can be provided as a pair of username (`ESBUsername`) and password (`ESBPassword`) or a authentication token (`ESBToken`) obtained from the **SAPO Security Token Service** (STS) [1].

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

3.1 HTTP Clients

As previously mentioned, every request must include both authentication and authorization information. The authentication information must be supplied in the URI. The authorization information must figure in the Authorization HTTP Header.

Note that in the samples below `{token}`, `{username}`, `{pass}`, and `{accessKey}` are place holders. In your code you must replace them with your authorization and authentication data.

3.1.1 Using username and password

In [Listing 3.1](#) is presented the structure of a request using `ESBUsername` and `ESBPassword` to do the authentication.

Listing 3.1: HTTP GET using `ESBUsername` and `ESBPassword`

```
1 GET https://services.sapo.pt/{service_name}/{operation_name}?ESBUsername={
    username}&ESBPassword={pass}&json=true HTTP/1.1
2 Authorization: ESB AccessKey={accessKey}
3 Host: services.sapo.pt
```

3.1.2 Using authentication token

In [Listing 3.2](#) is presented the structure of a request using ESSToken to do the authentication.

Listing 3.2: HTTP GET using ESSToken

```
1 GET https://services.sapo.pt/{service_name}/{operation_name}?ESSToken={token}&
   json=true HTTP/1.1
2 Authorization: ESST AccessKey={accessKey}
3 Host: services.sapo.pt
```

3.2 SOAP Clients

Photos Service

This service allows search and manage of photos that are hosted in **SAPO Photos**.

To interact with the service are available a SOAP interface and a HTTP-JSON interface.

4.1 HTTP

This section describes the HTTP-JSON interface.

In all the samples of this chapter it is used `ESBUsername` and `ESBPassword` to do the authentication. But as mentioned in [section 3.1](#) alternatively you can provided `ESBToken`.

4.1.1 ImageCreate

The operation **ImageCreate** is composed by two steps. First you have to send the meta data associated with the photo through a HTTP POST to `https://services.sapo.pt/Photos/ImageCreate`. Second you have to do a HTTP POST to `http://fotos.sapo.pt/uploadPost.html` with the photo file.

POST the meta data of the photo

In [Listing 4.1](#) is presented a sample request of the POST of the meta data of the photo.

For the sake of simplicity there are only provided the title and the tags of the image. It's recommended that when you submit a new photo you provide at least this attributes.

The complete list of Image attributes can be found at [\[2\]](#). But if you go forward a few pages and look carefully to [Listing 4.2](#) you can see the structure of the Image object.

Listing 4.1: POST the meta data of the photo

```
1 POST https://services.sapo.pt/Photos/ImageCreate?json=true&ESBUsername={username
   }&ESBPassword={password} HTTP/1.1
```

```

2 Content-Type: application/json
3 Authorization: ESB AccessKey={accessKey}
4 Content-Length: 48
5 Host: services.sapo.pt
6
7 {"image":{"title":"windows8","tags":"windows8"}}

```

The service response will have a similar structure to the one in [Listing 4.2](#). In the body of the response figures an object that has a `ImageCreateResponse` attribute. The value attribute is a `ImageCreateResult` object that as two attributes: `image` and `result`.

The image is a instance of `Image` type, which has the meta data supplied and another server generated fields, like `uid` that identifies the photo.

You can see if the request was accepted by the server looking to the `ok` attribute of the `result` object. The `{photo_token}` attribute is the token that you will have to send along with the photo file.

Note that `{photo_token}` is place holder and `{...}` is censoring sensitive data like usernames and photo ids.

Listing 4.2: Response of the HTTP POST with the meta data

```

1 HTTP/1.1 200 OK
2 Content-Type: application/json; charset=utf-8
3 Date: Tue, 18 Sep 2012 10:14:54 GMT
4 Content-Length: 1014
5
6 {
7   "ImageCreateResponse":{
8     "ImageCreateResult":{
9       "image":{
10         "active":"true",
11         "creationDate":"2012-09-18 11:00:43",
12         "innapropriate":"false",
13         "m18":"false",
14         "password":null,
15         "pending":"true",
16         "rating":"0",
17         "subtitle":null,
18         "synopse":null,
19         "tags":"sapo",
20         "title":"sapo",
21         "uid":"{...}",
22         "url":"http://fotos.sapo.pt/{...}/fotos/?uid={...}",
23         "user":{
24           "avatar":"http://imgs.sapo.pt/sapofotos/{...}/imgs/avatar.jpg",
25           "url":"http://fotos.sapo.pt/{...}/perfil",
26           "username":"{...}"

```

```
27         },
28         "visualizations": "0"
29     },
30     "result": {
31         "ok": "true"
32     },
33     "token": "{photo_token}"
34 }
35 }
36 }
```

POST the photo file

The second and final step of the upload of the photo is a multipart POST.

The structure of the request can be seen in [Listing 4.3](#).

Note that {photo_token} and {photo_bytes} are place holders. The first one is the token returned by the server in the ImageCreateResult object. The other is the bytes of the photo file.

Listing 4.3: HTTP GET using ESSToken

```
1 POST http://fotos.sapo.pt/uploadPost.html HTTP/1.1
2 Content-Type: multipart/form-data; boundary="imgboundary"
3 Host: fotos.sapo.pt
4 Content-Length: 75195
5
6 --imgboundary
7 Content-Disposition: form-data; name="token"
8
9 {photo_token}
10 --imgboundary
11 Content-Type: image/png
12 Content-Disposition: form-data; name="photo"; filename="windows8.png"
13
14 {photo_bytes}
```

If all goes as it should in the response you will get a response with **XML** content where you can find a Result tag with "SUCCESS" in the content. Otherwise you will get a response with a smaller **XML** document, that has in the Result tag the error code. You can see the complete error list at [\[3\]](#).

In [Listing 4.4](#) you can see a sample response body. In this case the photo was successfully submitted. So in the **XML** document will be present all the views generated. Each view has a URI (url tag) to the photo file. Along with the URI is also provided the with and the height of the photo.

Listing 4.4: Photo file upload response body

```

1  <?xml version="1.0"?>
2  <uploadPost>
3    <Ok/>
4    <Result>SUCCESS</Result>
5    <views>
6      <view>
7        <size>large</size>
8        <requestWidth>1600</requestWidth>
9        <requestHeight>1200</requestHeight>
10       <url>http://{...}.png</url>
11     </view>
12     (...)
13   </views>
14 </uploadPost>

```

In case of error, for instance if you provide an invalid token the body of the response will look like the one in [Listing 4.5](#).

Important: Note that in case of error the UploadPost tag begin with capital letter.

Listing 4.5: Response body in case of invalid token

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <UploadPost>
3    <Result>INVALID_TOKEN</Result>
4    <Error>Invalid token</Error>
5  </UploadPost>

```

4.1.2 ImageGetListBySearch

Listing 4.6: ImageGetListBySearch sample request

```

1  GET http://services.sapo.pt/Photos/ImageGetListBySearch?string=windows8&page=1&
    datefrom=2012-09-10&dateto=2012-09-12&json=true&ESBUsername={username}&
    ESBPassword={pass} HTTP/1.1
2  Authorization: ESB AccessKey={accessKey}
3  Host: services.sapo.pt

```

Listing 4.7: ImageGetListBySearch sample response body

```

1  {
2    "ImageGetListBySearchResponse":{
3      "ImageGetListBySearchResult":{
4        "images":{
5          "image":[
6            {

```

```
7         "active": "true",
8         "creationDate": "2011-09-14 22:56:00",
9         "innapropriate": "false",
10        "m18": "false",
11        "password": null,
12        "pending": "false",
13        "subtitle": null,
14        "synopse": "Windows 8",
15        "tags": "windows 8",
16        "title": "windows_8",
17        "uid": "{...}",
18        "url": "http://fotos.sapo.pt/{...}/fotos/?uid={...}",
19        "user": {
20            "avatar": "http://imgs.sapo.pt/sapofotos/{...}/imgs/avatar.
                jpg",
21            "url": "http://fotos.sapo.pt/{...}/perfil",
22            "username": "{...}"
23        },
24        "views": {
25            "view": [
26                {
27                    "requestHeight": "405",
28                    "requestWidth": "540",
29                    "size": "original",
30                    "url": "http://{...}.jpeg"
31                },
32                (...)
33            ]
34        }
35    },
36    (...)
37 ]
38 },
39 "result": {
40     "ok": "true",
41     "page": "1",
42     "perPage": "50",
43     "total": "41",
44     "totalPages": "1"
45 }
46 }
47 }
48 }
```

4.1.3 ImageGetListBySearch

This page was intentionally left in blank

Support and contacts

This page was intentionally left in blank

This page was intentionally left in blank

References

- [1] STS. Sapo Services. [Online]. Available: <https://store.services.sapo.pt/en/Catalog/development/free-api-sts>
- [2] Photos image type. Sapo Services. [Online]. Available: <https://store.services.sapo.pt/en/Catalog/social/free-api-photos/technical-description#entity-type-Photos-Image>
- [3] Photos imagecreate. Sapo Services. [Online]. Available: <https://store.services.sapo.pt/en/Catalog/social/free-api-photos/technical-description#service-Photos-operation-ImageCreate>