SAPO Services SDK Documentation

v0.9

Contents

1 Introduction

The present document describes how to use the SAPO Services.

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.

2 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.

2.1 HTTP Clients

As previously said, 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 some services the **ESBAccessKey** may figure in the query string (**...&ESBAccessKey={accessKey}&...**).

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.

2.1.1 Using username and password

The following example presents the structure of a request using **ESBUsername** and **ESBPassword** to do the authentication. Authorization and Host are HTTP headers.

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

2.1.2 Using authentication token

The following example presents the structure of a request using **ESBToken** to do the authentication. Authorization and Host are HTTP headers.

```
GET
https://services.sapo.pt/{service_name}/{operation_name}?ESBToken={token}&j
son=true HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

3 Photos Service

The **Photos** service allows to search and to manage of photos that are hosted in **SAPO Photos** (http://fotos.sapo.pt/).

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

3.1 HTTP

This section describes the HTTP-JSON interface of the **Photos** service

In all the samples of this chapter it is used **ESBUsername** and **ESBPassword** to do the authentication. But as mentioned in the previous chapter, alternatively you can provide **ESBToken**.

3.1.1 Submit a photo to the service

To submit a photo to the service you can use the **ImageCreate** operation.

The **ImageCreate** operation is composed by two steps. First you have to send the metadata 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.

3.1.1.1 POST the metadata of the photo

The following example is a sample HTTP POST request of the metadata of the photo.

```
POST
https://services.sapo.pt/Photos/ImageCreate?json=true&ESBUsername={username
}&ESBPassword={password} HTTP/1.1
Content-Type: application/json
Authorization: ESB AccessKey={accessKey}
Content-Length: 48
Host: services.sapo.pt

{"image":{"title":"windows8","tags":"windows8"}}
```

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 https://store.services.sapo.pt/en/Catalog/social/free-api-photos/technical-description#entity-type-Photos-Image. But if you go forward a couple of pages you will find a example with all the Image object attributes.

The service response will have a similar structure to the one in example below. In the body of the response figures an object that has an **ImageCreateResponse** attribute. The value attribute is an **ImageCreateResult** object that as three attributes: image, result and token.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Date: Tue, 18 Sep 2012 10:14:54 GMT
Content-Length: 1014
{
   "ImageCreateResponse":{
      "ImageCreateResult":{
         "image":{
            "active":"true",
            "creationDate":"2012-09-18 11:00:43",
            "innapropriate":"false",
            "m18":"false",
            "password":null,
            "pending":"true",
            "rating":"0",
            "subtitle":null,
            "synopse":null,
            "tags":"sapo",
            "title":"sapo",
            "uid":"{...}",
            "url":"http://fotos.sapo.pt/{...}/fotos/?uid={...}",
            "user":{

"avatar":"http://imgs.sapo.pt/sapofotos/{...}/imgs/avatar.jpg",
               "url":"http://fotos.sapo.pt/{...}/perfil",
               "username":"{...}"
            },
            "visualizations":"0"
         },
         "result":{
            "ok":"true"
         },
         "token":"{photo_token}"
      }
   }
}
```

The image is an instance of **Image** type, which has the metadata 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 censuring sensitive data like usernames and photo ids.

3.1.1.2 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 following example.

```
POST http://fotos.sapo.pt/uploadPost.html HTTP/1.1
Content-Type: multipart/form-data; boundary="imgboundary"
Host: fotos.sapo.pt
Content-Length: 75195

--imgboundary
Content-Disposition: form-data; name="token"

{photo_token}
--imgboundary
Content-Type: image/png
Content-Disposition: form-data; name="photo"; filename="windows8.png"
{photo_bytes}
--imgboundary--
```

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.

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 https://store.services.sapo.pt/en/Catalog/social/free-api-photos/technical-description#service-Photos-operation-ImageCreate.

In the following example 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 **width** and the **height** of the photo.

```
<?xml version="1.0"?>
<uploadPost>
   <Ok/>
   <Result>SUCCESS</Result>
   <views>
      <view>
      <size>large</size>
      <requestWidth>1600</requestWidth>
      <requestHeight>1200</requestHeight>
      <url>http://{...}.png</url>
      </view>
      (...)
   </views>
</uploadPost>
```

In case of error, for instance if you provide an invalid token the body of the response will look like the one in following example.

```
<?xml version="1.0" encoding="utf-8" ?>
<UploadPost>
   <Result>INVALID_TOKEN</Result>
   <Error>Invalid token</Error>
</UploadPost>
```

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

3.1.2 Search images by terms

To search images by terms you can use the **ImageGetListBySearch** operation.

To do this search you have to do a HTTP GET to the following URI: https://services.sapo.pt/Photos/ImageGetListBySearch?json=true.

The search terms must be specified in the **string** query string parameter. But it's recommended that you can provide additional search information like dates. The complete list of query string parameters that you can provide to this service are:

- **string** – the search terms separated by commas and with no spaces (e.g. "windows,microsoft");
- **datefrom** – the date from the search results begin in the format "yyyy-mm-dd" (e.g. 2012-10-15);
- **dateto** – the date to the search results end in the format "yyyy-mm-dd" (e.g. 2012-10-17);
- **page** – the results supports paging, in this parameter you can specify the page. (e.g. "1").

Important: Note that it's mandatory to include **json** parameter and it must be set with **true**. And like all other requests the authentication date must be provided in the query string.

The following example is presented a sample request:

```
GET
https://services.sapo.pt/Photos/ImageGetListBySearch?string=windows,microso
ft&page=1&datefrom=2012-09-10&dateto=2012-09-
12&json=true&ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The service response body will have a similar structure to the one in example below. In the body of the response figures an object that has an **ImageGetListBySearchResponse** attribute. The value attribute is an **ImageGetListBySearchResult** object that as two attributes: **images** and **result**. The **images** attribute is an array of **image** objects.

```
{
   "ImageGetListBySearchResponse":{
      "ImageGetListBySearchResult":{
         "images":{
            "image":[
               {
                  "active":"true",
                  "creationDate":"2011-09-14 22:56:00",
                  "innapropriate":"false",
                  "m18":"false",
                  "password":null,
                  "pending":"false",
                  "subtitle":null,
                  "synopse":"Windows 8",
                  "tags":"windows 8",
                  "title":"windows_8",
                  "uid":"{...}",
                  "url":"http://fotos.sapo.pt/{...}/fotos/?uid={...}",
                  "user":{
"avatar":"http://imgs.sapo.pt/sapofotos/{...}/imgs/avatar.jpg",
                     "url":"http://fotos.sapo.pt/{...}/perfil",
                     "username":"{...}"
                  },
                  "views":{
                     "view":[
                        {
                           "requestHeight":"405",
                           "requestWidth":"540",
                           "size":"original",
                           "url":"http://{...}.jpeg"
                        },
                        (...)
                     ]
                  }
               },
               (...)
            ]
         },
         "result":{
            "ok":"true",
            "page":"1",
            "perPage":"50",
            "total":"41",
            "totalPages":"1"
         }
      }
   }
}
```

If you pretend to download the photo, in the **image** object's **views** attribute is provided and array of **view** objects. Each view is a version of the photo with different dimensions (with and height). The **url** attribute has the URI from you can GET the photo file.

3.1.3 Delete a photo

To delete a photo you can use the **ImageDelete** operation.

To do a delete of a photo you have to do a HTTP GET to the following URI: https://services.sapo.pt/Photos/ImageDelete?json=true.

The **uid** of the photo must be included in the query string parameters.

The following example is a sample request to delete the photo with the **uid** {id}.

```
GET
https://services.sapo.pt/Photos/ImageDelete?uid={id}json=true&ESBUsername={
username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

If the photo is successfully deleted, the server will send in the response body a JSON object similar to the one below:

```
{
   "ImageDeleteResponse": {
      "ImageDeleteResult":{"ok":"true"}
   }
}
```

3.1.4 Get photo details

To get the details of a photo you can use the **ImageDetails** operation.

In order to get the details of the photo you have to do a HTTP GET to the following URI: https://services.sapo.pt/Photos/ImageDetails?json=true.

The **uid** of the photo must be included in the query string parameters.

The following example is a sample request to get the details of the photo with the **uid** {id}.

```
GET
https://services.sapo.pt/Photos/ImageDetails?uid={id}json=true&ESBUsername=
{username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body will be similar to the one in the following example:

```
{
   "ImageDetailsResponse":{
      "ImageDetailsResult":{
         "image":{
            "active":"true",
            "creationDate":"2012-09-05 11:16:03",
            "innapropriate":"false",
            "m18":"false",
            "password":null,
            "pending":"false",
            "subtitle":null,
            "synopse":null,
            "tags":"stack,windows",
            "title":"Windows 8 stack",
            "uid":"{...}",
            "url":"http://fotos.sapo.pt/{...}/fotos/?uid={...}",
            "user":{
               "avatar":"http://imgs.sapo.pt/sapofotos/{...}/imgs/avatar.jpg",
               "url":"http://fotos.sapo.pt/{...}/perfil",
               "username":"{...}"
            },
            "views":{
               "view":[
                  {
                     "requestHeight":"581",
                     "requestWidth":"1024",
                     "size":"original",
                     "url":"http://{...}.jpeg"
                  },
                  (....)
               ]
            }
         },
         "result":{
            "ok":"true"
         }
      }
   }
}
```

3.1.5 Get user's photos

To get the user's photos you can use the **ImageGetListByUser** operation.

In order to get the list of photos of the user you have to do a HTTP GET to the following URI: https://services.sapo.pt/Photos/ImageGetListByUser?json=true.

The **username** of the user should be included in the query string parameters. If you don't specify the username the service will assume that the user is the authenticated user.

The following example is a sample request to get the photos list of the user with the **username** {username}.

```
GET
https://services.sapo.pt/Photos/ImageGetListByUser?username={username}json=
true&ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body will be similar to the one in the following example:

```
{
   "ImageGetListByUserResponse":{
      "ImageGetListByUserResult":{
         "images":{
            "image":[
               {
                  "active":"true",
                  "creationDate":"2012-09-18 11:00:42",
                  "innapropriate":"false",
                  "m18":"false",
                  "password":null,
                  "pending":"false",
                  "subtitle":null,
                  "synopse":null,
                  "tags":null,
                  "title":"sapo",
                  "uid":"{...}",
                  "url":"http://fotos.sapo.pt/{...}/fotos/?uid={...}",
                  "views":{
                     "view":[
                        {
                           "requestHeight":"450",
                           "requestWidth":"450",
                           "size":"original",
                           "url":"http://{...}.png"
                        },
                        (...)
                     ]
                  }
               },
               (...)
            ]
         },
         "result":{
            "ok":"true",
            "page":"1",
            "perPage":"50",
            "total":"11",
            "totalPages":"1"
         }
      }
   }
}
```

Note that if the photos are in an album the **image** object will have an attribute **albums** with the albums that the photo belongs to.

3.1.6 Create an Album

To create an album you can use the **AlbumCreate** operation.

The creation is done doing a HTTP POST to the following URI: https://services.sapo.pt/Photos/AlbumCreate?json=true. You must provide a JSON object with the album representation.

The following example is a sample request to create an album:

```
POST
https://services.sapo.pt/Photos/AlbumCreate?json=true&ESBUsername={username
}&ESBPassword={pass} HTTP/1.1
Host: services.sapo.pt
Authorization: ESB AccessKey={accessKey}
Content-Length: 30
Content-Type: application/json

{"album":{"title":"Windows8"}}
```

The complete list of the **album** attributes can be found at https://store.services.sapo.pt/en/Catalog/social/free-api-photos/technical-description#entity-type-Photos-Album.

If the album is successfully created the response body of the request will be similar to the one in the following example:

```
{
   "AlbumCreateResponse":{
      "AlbumCreateResult":{
         "album":{
            "active":"true",
            "cover":"http://fotos.sapo.pt/{...}/capa-de-album/4",
            "creationDate":"2012-09-19 12:51:03",
            "description":null,
            "id":"4",
            "ownerId":"181291",
            "password":null,
            "title":"Windows8",
            "url":"http://fotos.sapo.pt/{...}/albuns/?aid=4",
            "user":{
               "avatar":"http://imgs.sapo.pt/sapofotos/{...}/imgs/avatar.jpg",
               "url":"http://fotos.sapo.pt/{...}/perfil",
               "username":"{...}"
            }
         },
         "result":{
            "ok":"true"
         }
      }
   }
}
```

3.1.7 Add an photo to albums

To add a photo to albums you can use the **AlbumCreate** operation.

To do this you have to do a GET request to the following URI: https://services.sapo.pt/Photos/ImageAddToAlbum?json=true.

In the query string you have to specify the **image uid** and the **albums ids**. This is done with the following query string parameters:

- **imageuid – image uid**;
- **albumid – albums ids** separated by commas (e.g. “2,3”);

The following example is a sample request that adds the photo with the **uid {id}** to **albums 2** and **3**.

```
GET
https://services.sapo.pt/Photos/ImageAddToAlbum?ESBUsername={user}&ESBPassw
ord={pass}&imageuid={id}&albumid=2,3&json=true HTTP/1.1
Host: services.sapo.pt
Authorization: ESB AccessKey={accessKey}
```

If the photos are successfully added to the albums, the response body will be similar to the one in the following example:

```
{
   "ImageAddToAlbumResponse":{
      "ImageAddToAlbumResult":{
         "result":{
            "ok":"true"
         }
      }
   }
}
```

3.1.8 Get album images

To get a user's album photos list you can use the **ImageGetListByUserAlbum** operation.

You have to do a HTTP GET request to the following URI, specifying the **SAPO Photos username** and the **album id**: https://services.sapo.pt/Photos/ImageGetListByUserAlbum?json=true. The **SAPO Photos username** should be passed as **username** query string parameter and the **album id** should be passed as the **id** query string parameter.

The following example is a sample request to get the photos list of the **album** with **id 2** of the **user** with **SAPO Photos username {sapophotos_username}**:

```
GET
https://services.sapo.pt/Photos/ImageGetListByUserAlbum?ESBUsername={userna
me}&ESBPassword={pass}&username={sapofotos_username}&id=2&page=1&json=true
HTTP/1.1
Host: services.sapo.pt
Authorization: ESB AccessKey={accessKey}
Content-Type: application/json
```

The response body is similar to the one in the following example:

```
{
   "ImageGetListByUserAlbumResponse":{
      "ImageGetListByUserAlbumResult":{
         "images":{
            "image":[
               {
                  "active":"true",
                  "creationDate":"2012-09-04 12:58:45",
                  "innapropriate":"false",
                  "m18":"false",
                  "password":null,
                  "pending":"false",
                  "subtitle":null,
                  "synopse":null,
                  "tags":null,
                  "title":"microsoft",
                  "uid":"{...}",
                  "url":"http://fotos.sapo.pt/{...}/fotos/?uid={...}",
                  "views":{
                     "view":[
                        {
                           "requestHeight":"465",
                           "requestWidth":"620",
                           "size":"original",
                           "url":"http://{...}.jpeg"
                        },
                        (...)
                     ]
                  }
               },
               (...)
            ]
         },
         "result":{
            "ok":"true",
            "page":"1",
            "perPage":"50",
            "total":"2",
            "totalPages":"1"
         }
      }
   }
}
```

3.1.9 Get user's albums

To get an user' albums list you can use the **AlbumGetListByUser** operation.

You have to do a HTTP GET request to the following URI, specifying the **SAPO Photos username**: https://services.sapo.pt/Photos/AlbumGetListByUser?json=true. The SAPO Photos username should be passed as username query string parameter

The following example is a sample request to get the albums list of the user which SAPO Photos username is **{sapophotos_username}**:

```
GET
https://services.sapo.pt/Photos/AlbumGetListByUser?username={sapophotos_use
rname}&json=true&ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "AlbumGetListByUserResponse":{
      "AlbumGetListByUserResult":{
         "albums":{
            "album":[
               {
                  "active":"true",
                  "cover":"http://fotos.sapo.pt/{...}/capa-de-album/5",
                  "creationDate":"2012-09-24 16:47:14",
                  "description":null,
                  "id":"5",
                  "password":null,
                  "title":"Codebits",
                  "url":"http://fotos.sapo.pt/{...}/albuns/?aid=5"
               },
               (...)
            ]
         },
         "result":{
            "ok":"true",
            "page":"1",
            "perPage":"50",
            "total":"6",
            "totalPages":"1"
         }
      }
   }
}
```

3.1.10 Get user's account details

To get the account details of a user you can use the **UserDetails** operation.

You have to do a HTTP GET request to the following URI: https://services.sapo.pt/Photos/UserDetails?json=true. This request will return the account details of the authenticated user.

The following example is a sample request to get the user's account details:

```
GET
https://services.sapo.pt/Photos/UserDetails?json=true&ESBUsername={username
}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "UserDetailsResponse":{
      "UserDetailsResult":{
         "result":{
            "ok":"true"
         },
         "user":{
            "active":"true",
            "avatar":"http://imgs.sapo.pt/sapofotos/{...}/imgs/avatar.jpg",
            "banned":"false",
            "creationDate":"2012-09-04 12:46:11",
            "lastLogin":"2012-10-19 11:28:40",
            "url":"http://fotos.sapo.pt/{...}/perfil",
            "username":"{...}"
         }
      }
   }
}
```

4 Videos Service

The **Videos** service allows to search and to manage videos that are hosted in **SAPO Videos** (http://videos.sapo.pt/);

To interact with the service is available a HTTP interface and a SOAP interface.

4.1 HTTP

This section describes the HTTP-JSON interface of the **Videos** service.

4.1.1 Submit a video to the service

To submit a video to the service you can use the **AddVideoPost** operation.

The **AddVideoPost** operation is composed by two steps. First, you have to send the metadata associated with the video through a HTTP POST request to the following URI: https://services.sapo.pt/Videos/AddVideoPost?json=true. Second, you have to do a HTTP POST request to the following URI with the video file: http://upload.videos.sapo.pt/upload_token.html.

4.1.1.1 POST the video's metadata

The following example is a sample request of the HTTP POST request that sends the video metadata:

```
POST
https://services.sapo.pt/Videos/AddVideoPost?json=true&ESBUsername={usernam
e}&ESBPassword={password} HTTP/1.1
Content-Type: application/json
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
Content-Length: 105

{"Video":{"Title":"Microsoft meets Sapo","Tags":"Microsoft meets
Sapo","Synopse":"Microsoft meets Sapo"}}
```

In the request body must be sent a JSON representation of the Video object. You can find in the following URI the complete list of attributes of this object: https://store.services.sapo.pt/en/Catalog/social/free-api-videos/technical-description#entity-type-Videos-VideoSubmition.

Note that for this operation the **title**, **synopse** and **tags** attributs are mandatory.

The service response will have a similar structure to the one in the following example:

```
{
   "AddVideoPostResponse":{
      "AddVideoPostResult":{
         "Active":"0",
         "CanPublish":"1",
         "CanShare":"1",
         "Comments":"0",
         "CreationDate":"2012-10-17 17:26:07",
         "Embed":null,
         "HD":"0",
         "Id":"{...}",
         "Length":"0",
         "Pending":"1",
         "Randname":"{...}",
         "Rating":"0",
         "ReferedExternal":null,
         "Subtitle":null,
         "Synopse":"Microsoft meets Sapo",
         "Tags":"microsoft, meets, sapo",
         "Thumbnail":null,
         "Title":"Microsoft meets Sapo",
         "Token":"{video_token}",
         "URL":null,
         "Username":"{...}",
         "Views":"0"
      }
   }
}
```

The AddVideoPost object's **token** attribute has the video token that must be sent when submitting the video file.

Note that **{video_token}** is a place holder and **{...}** is censuring sensitive data like usernames and video ids.

4.1.1.2 POST the video's file.

The second and final step of the upload of the video is a **multipart** POST. The structure of the request can be seen in the following example:

```
POST http://upload.videos.sapo.pt/upload_token.html HTTP/1.1
Content-Type: multipart/form-data; boundary="videoboundary"
Host: upload.videos.sapo.pt
Content-Length: 189497

--videoboundary
Content-Disposition: form-data; name="token"

{video_token}
--videoboundary
Content-Type: video/mp4
Content-Disposition: form-data; name="content_file"; filename="Microsoft
meets Sapo.mp4"
{video_bytes}
--videoboundary--
```

Note that **{video_token}** and **{video_bytes}** are place holders. The first one is the token returned by the server in **AddVideoPostResult** object. The other is the bytes of the video file.

If all goes as it should in the response you will get a response with **XML** content where you can find an **Ok** tag.

In the following example you can see a sample response body. In this case the video was successfully submitted.

```
HTTP/1.1 200 OK
Date: Mon, 22 Oct 2012 13:41:52 GMT
Server: Apache/2.2.15 (Unix) PHP/5.2.13
X-Powered-By: Videos@Sapo PTC
Access-Control-Allow-Origin: *
Vary: Accept-Encoding
Content-Length: 123
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8" ?>
<UploadToken xmlns="http://services.sapo.pt/definitions">
 <Ok></Ok>
</UploadToken>
```

4.1.2 Delete a video

To delete a video you can use the **DeleteVideo** operation.

To do a video deletion you have to do a HTTP GET request to the following URI: https://services.sapo.pt/Videos/DeleteVideo?json=true.

The **Randname** of the video must be included in the query string parameters.

The following example is a sample request to delete the video with the **Randname {randname}**.

```
GET
https://services.sapo.pt/Videos/DeleteVideo?Randname={randname}&json=true&E
SBUsername={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

If the video is successfully deleted, the server will send a response similar to the one in the following example

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: application/json; charset=utf-8
Content-Encoding: gzip
Server: SAPO SDB 3.1
X-Server: LEA
Date: Mon, 22 Oct 2012 13:53:37 GMT
Content-Length: 0
```

4.1.3 Get video details

To get a video details you can use the **CheckVideo** operation. To use this operation you have to do a HTTP GET request to the following URI: https://services.sapo.pt/Videos/CheckVideo?json=true.

The video's **Randname** must be included in the **VideoRandname** query string parameter.

The following example is a sample request to get the video details of the video with **Randname {randname}**.

```
GET
https://services.sapo.pt/Videos/CheckVideo?VideoRandname={randname}&json=tr
ue&ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "CheckVideoResponse":{
      "CheckVideoResult":{
         "Active":"0",
         "CanPublish":"1",
         "CanShare":"1",
         "Comments":"0",
         "CreationDate":"2012-10-22 14:41:50",
         "Embed":"<embed
src="http://rd3.videos.sapo.pt/play?file=http://rd3.videos.sapo.pt/{...}/mov/1"
type="application/x-shockwave-flash" allowFullScreen="true" width="400"
height="350"></embed>",
         "HD":"0",
         "Id":"{...}",
         "Length":"4",
         "Pending":"0",
         "Randname":"{...}",
         "Rating":"0",
         "ReferedExternal":null,
         "Subtitle":null,
         "Synopse":"Microsoft meets Sapo",
         "Tags":"meets, microsoft, sapo",
         "Thumbnail":"http://{...}.jpg",
         "Title":"Microsoft meets Sapo",
         "URL":"http://videos.sapo.pt/{...}",
         "Username":"{...}",
         "VideoSourceHD":null,
         "VideoSourceSD":"http://{...}.mp4",
         "Views":"0"
      }
   }
}
```

4.1.4 Get user details

To get an user's account details you can use the **GetUserInfo** operation. To use this operation you have to do a HTTP GET request to the following URI: https://services.sapo.pt/Videos/GetUserInfo?json=true.

The user's email must be included in the **Email** query string parameter. Note that the provided email must be the authenticated user email.

The following example is a sample request to get the account details of the authenticated user.

```
GET
https://services.sapo.pt/Videos/GetUserInfo?Email={email}&json=true&ESBUser
name={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "GetUserInfoResponse":{
      "GetUserInfoResult":{
         "Active":"1",
         "Banned":"0",
         "CommentCaptcha":"1",
         "CommentModerate":"0",
         "CommentNotify":"0",
         "CreationDate":"2012-09-21 10:14:58",
         "Email":"{...}",
         "Id":"{...}",
         "LastLogin":"2012-10-16 14:37:39",
         "Partner":"0",
         "UploadMail":"{...}",
         "Username":"{...}",
         "ViewAll":"0",
         "Website":null
      }
   }
}
```

4.1.5 Get user videos

To get an user's video list you can use the **ListUserVideos** operation. To use this operation you have to do a HTTP GET request to the following URI: https://services.sapo.pt/Videos/ListUserVideos?json=true.

The following example is a sample request to get the video's list of the authenticated user.

```
GET
https://services.sapo.pt/Videos/ListUserVideos?json=true&ESBUsername={usern
ame}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "ListUserVideosResponse":{
      "ListUserVideosResult":{
         "item":[
            {
               "Active":"0",
               "Comments":"0",
               "CreationDate":"2012-10-16 14:58:51",
               "Embed":"<embed
src="http://rd3.videos.sapo.pt/play?file=http://rd3.videos.sapo.pt/{...}/mo
v/1" type="application/x-shockwave-flash" allowFullScreen="true"
width="400" height="350"></embed>",
               "Id":"2035494",
               "Length":"4",
               "Pending":"0",
               "Randname":"{...}",
               "Rating":"0",
               "ReferedExternal":null,
               "ResultTotalVideos":"15",
               "Subtitle":null,
               "Synopse":"Microsoft meets Sapo",
               "Thumbnail":"http://{...}.jpg",
               "Title":"Microsoft meets Sapo",
               "URL":"http://videos.sapo.pt/{...}",
               "Username":null,
               "Views":"0"
            },
            (...)
         ]
      }
   }
}
```

4.1.6 Get highlighted videos

To get the current highlighted videos you should use the **JSON2/Highlights** operation. To use this operations you do a HTTP GET request to the following URI: https://services.sapo.pt/Videos/JSON2/Highlights?json=true.

The following request is a sample request to get the current video highlights.

```
GET
https://services.sapo.pt/Videos/JSON2/Highlights?json=true&ESBUsername={use
rname}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "rss":{
      "channel":{
         "atom:link":{
            "href":"http://videos.sapo.pt/destaques_xml2.html",
            "rel":"self",
            "type":"application\/rss+xml"
         },
         "description":"Lista de Destaques",
         "item":[
            {
               "description":"<h3>Autor: meo</h3>\u000a<h4>Blasted Mechanism no
MEO LIKE MUSIC</h4>\u000a<a
href="http://videos.sapo.pt/vKWa1eBAvyorhzR9OSoY">\u000a<img
src="http://cache15.stormap.sapo.pt/vidstore07/thumbnais/66/a4/5d/6349415_BCc0n.
jpg" width="120" height="90" border="0"/>\u000a</a>\u000a<p>\u000aVisite os
vídeos de <a href="http://videos.sapo.pt/meo">meo</a>\u000a</p>",
               "guid":"http://videos.sapo.pt/vKWa1eBAvyorhzR9OSoY",
               "link":"http://videos.sapo.pt/vKWa1eBAvyorhzR9OSoY",
               "media:content":{
                  "height":"90",
                  "type":"image/jpeg",

"url":"http://cache15.stormap.sapo.pt/vidstore07/thumbnais/66/a4/5d/6349415_BCc0
n.jpg",
                  "width":"120"
               },
               "pubDate":"Fri, 12 Oct 2012 15:09:13 +0100",
               "sapo:author":"meo",
               "sapo:comment_count":"0",
               "sapo:default_playlist":null,
               "sapo:m18":"false",
               "sapo:rate":"3",
               "sapo:synopse":"Não percas o concerto dos Blasted Mechanism, dia
24/OUT às 22h00, no MEO Like Music, a tua sala de concertos
interativa!\u000d\u000a\u000d\u000aSabe mais em http://meo.pt/likemusic",
               "sapo:time":"00:01:21",
               "sapo:videoURL":"http://videos.sapo.pt/vKWa1eBAvyorhzR9OSoY",
               "sapo:views":"509",
               "title":"Blasted Mechanism no MEO LIKE MUSIC"
            },
            (...)
         ],
         "lastBuildDate":"Wed, 17 Oct 2012 17:51:03 +0100",
         "link":"http://videos.sapo.pt",
         "title":"SAPO Vídeos RSS: Destaques"
      },
      "version":"2.0",
      "xmlns:atom":"http://www.w3.org/2005/Atom",
      "xmlns:media":"http://search.yahoo.com/mrss/",
      "xmlns:sapo":"http://videos.sapo.pt/mrss/"
   }
}
```

4.1.7 Search videos

To search for videos you should use the **JSON2/Query** operation. To use this operation you do a HTTP GET request to the following URI: https://services.sapo.pt/Videos/JSON2/Query?json=true.

The following parameters are accepted in the query string:

- **search** – the search terms;
- **user** – you can provide the **SAPO Videos** username if you want to search videos of a specific user;
- **page** – the page number;

- **limit** – number of videos per page;
- **order** – order of the returned videos;

The following example is a sample request to get the videos from the videos service that match the provided search terms (microsoft):

```
GET
https://services.sapo.pt/Videos/JSON2/Query?search=microsoft&page=5&limit=1
&order=views&json=true&ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "rss":{
      "channel":{
         "atom:link":{
            "href":"http://videos.sapo.pt/query_rss2.html",
            "rel":"self",
            "type":"application/rss+xml"
         },
         "description":"Lista de resultados dos vídeos",
         "item":{
            "description":"<h3>Autor: gameover</h3>\u000a<h4>Kinect e o monitor do
&quot;futuro&quot;</h4>\u000a<a
href="http://videos.sapo.pt/Go4ZqYKFItopZCZMbuqe">\u000a  <img
src="http://cache15.stormap.sapo.pt/vidstore11/thumbnais/ee/76/89/5264850_1MYG0.jpg"
width="120" height="90" border="0"/>\u000a </a>\u000a<p>\u000aVisite os vídeos de <a
href="http://videos.sapo.pt/gameover">gameover</a>\u000a</p>",
            "guid":"http://videos.sapo.pt/Go4ZqYKFItopZCZMbuqe",
            "link":"http://videos.sapo.pt/Go4ZqYKFItopZCZMbuqe",
            "media:content":{
               "height":"90",
               "type":"image/jpeg",

"url":"http://cache15.stormap.sapo.pt/vidstore11/thumbnais/ee/76/89/5264850_1MYG0.jp
g",
               "width":"120"
            },
            "pubDate":"Tue, 13 Mar 2012 10:39:54 +0000",
            "sapo:author":"gameover",
            "sapo:comment_count":"0",
            "sapo:default_playlist":null,
            "sapo:m18":"false",
            "sapo:rate":"5",
            "sapo:synopse":"A Microsoft uniu a tecnologia de monitorização de
movimentos a um monitor OLED transparante da Samsung para demonstrar um possível
monitor do "futuro".",
            "sapo:time":"00:01:48",
            "sapo:videoURL":"http://videos.sapo.pt/Go4ZqYKFItopZCZMbuqe",
            "sapo:views":"6578",
            "title":"Kinect e o monitor do "futuro""
         },
         "lastBuildDate":"Mon, 22 Oct 2012 15:48:10 +0100",
         "link":"http://videos.sapo.pt/search.html?word=microsoft",
         "opensearch:itemsPerPage":"1",
         "opensearch:startIndex":"4",
         "opensearch:totalResults":"1287",
         "title":"SAPO Vídeos RSS: Query"
      },
      "version":"2.0",
      "xmlns:atom":"http://www.w3.org/2005/Atom",
      "xmlns:media":"http://search.yahoo.com/mrss/",
      "xmlns:opensearch":"http://a9.com/-/spec/opensearch/1.1/",
      "xmlns:sapo":"http://videos.sapo.pt/mrss/"
   }
}
```

5 Verbetes Service

Verbetes is a service that answers to requests of type "WhoIs" for public personalities. All the information from **Verbetes** is collected automatically from news sources, and its information is updated on an hour basis, as new news are collected. **Verbetes** also answers to requests of type "WhoIs" for jobs of public personalities mentioned on news.

To interact with the service is available a HTTP-JSON interface.

5.1 HTTP

This section describes the HTTP-JSON interface of the **Verbetes** service.

In all the samples of this chapter it is used **ESBUsername** and **ESBPassword** to do the authentication. But as previously mentioned, alternatively you can provide **ESBToken**.

5.1.1 Who is

To find "verbetes" about public personalities or jobs you can use the **WhoIs** operation.

To do this searches you have to do a GET request to the following URI: https://services.sapo.pt/InformationRetrieval/Verbetes/WhoIs. It's mandatory to specify one of the following query string parameters:

- **name** – personality's name;
- **name_like** – a part of a personality's name;
- **job** – profession name;
- **job_like** – a part of a profession's name;

There are additional query string parameters that can be included:

- **date** – occured date;
- **margin** – margin of occurrence (in days);
- **min** – retrieve only entries that appear at least this number of times;

For more information about these additional parameters see the following URI: https://store.services.sapo.pt/en/Catalog/other/free-api-information-retrieval-verbetes/technical-description#service-InformationRetrieval/Verbetes-operation-WhoIs

The following example is a sample request that searches for "verbetes" of personalities for name.

```
GET
https://services.sapo.pt/InformationRetrieval/Verbetes/WhoIs?name={name}&da
te=2012-11-
15&margin=60&min=5&format=json&ESBUsername={username}&ESBPassword={pass}
HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "verbetes":[
      {
         "jobs":[
            {
               "lastSeen":"2012-01-14",
               "num":"6",
               "ergo":"chefe do Governo",
               "active":"no",
               "firstSeen":"2010-10-21"
            },
            (...)
         ],
         "officialName":"José Sócrates",
         "alternativeNames":[
            "Sócrates"
         ]
      }
   ]
}
```

Note that exists a light version of the **WhoIs** operation that's designed to be faster. It is similiar to the **WhoIs**, so the considerations of this subsection also apply to the light version (**WhoIsLight**).

5.1.2 Get personalities

To get all personalities that have at least **n** occurrences you can use the **GetPersonalities** operation.

Through the **min** query string parameter you can specify the minimum number of occurrences of the personality "verbete" in order to filter results.

The following example is a sample request to get all the personalities with verbetes, where each verbete has at least 10 occurrences.

```
GET
https://services.sapo.pt/InformationRetrieval/Verbetes/GetPersonalities?min
=10&format=json&ESBUsername={username}&ESBPassword={pass}} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example (list of pairs **<name>:<number of occurences>**):

```
{
   "listPersonalities":{
      "Jorge Moreira da Silva":98,
      "José Eduardo Matos":40,
      "Domingos de Azevedo":71,
      "João Semedo":266,
      "José Mota":178,
      (...)
   }
}
```

5.1.3 Get jobs

To get all jobs that have at least **n** occurrences you can use the **GetErgos** operation.

Through the **min** query string parameter you can specify the minimum number of occurrences of the job “verbete” in order to filter results.

The following example is a sample request to get all the personalities with verbetes, where each verbete has at least 300 occurrences.

```
GET
https://services.sapo.pt/InformationRetrieval/Verbetes/GetErgos?min=300&for
mat=json&ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example (list of pairs **<job>:<number of occurences>**):

```
{
   "listErgos":{
      "presidente":9754,
      "ministro":4880,
      "secretário-geral":3770,
      "treinador":3445,
      "secretário":2400,
      "primeiro-ministro":1640,
   }
}
```

5.1.4 Get egocentric network of personality

To get the egocentric network of some personality you can use the **GetEgoNet** operation.

In addition to the name of the personality that must be specified in the **name** there are other query string parameters that you can specify:

- **depth** – depth of the search in the network;
- **minFrequencyEdges** – minimum number of frequency edges;
- **beginDate** – start date of the dates interval to search;
- **endDate** – end date of the dates interval to search;

The following example is a sample request to get the egocentric network of some personality.

```
GET
https://services.sapo.pt/InformationRetrieval/Verbetes/GetEgoNet?depth=1.5&
minFrequencyEdges=20&name={name}&beginDate=2012-09-01&endDate=2012-09-17&
ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Host: services.sapo.pt
Authorization: ESB AccessKey={accessKey}
```

The response body is similar to the one in the following example:

```
{
   "edges":[
      {
         "source":"André Almeida",
         "target":"Carlos Martins",
         "edge_frequency":41
      },
      {
         "source":"André Almeida",
         "target":"Maxi Pereira",
         "edge_frequency":40
      },
      (...)
   ],
   "endDate":"2012-09-17",
   "beginDate":"2012-09-01",
   "minFrequencyEdges":"20",
   "nodes":{
      "André Almeida":107,
      "Carlos Martins":240,
      "Pablo Aimar":45,
      (...)
   },
   "depth":"1.5",
   "ego":"Jorge Jesus"
}
```

5.1.5 Get personalities co-occurrences

To get the personalities who co-occur with a given personality you can use the **GetCoOccurences** operation.

In order to do this search you have to do a GET request to the following URI: https://services.sapo.pt/InformationRetrieval/Verbetes/GetCoOccurrences. You should specify in the query string a personality's name and the date interval of the search (**begin_date** and **end_date**).

The following example is a sample request to get the personalities who co-occur with the given personality:

```
GET
https://services.sapo.pt/InformationRetrieval/Verbetes/GetCoOccurrences?nam
e={name}&begin_date=2012-08-01&end_date=2012-08-
30&format=json&ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Host: services.sapo.pt
Authorization: ESB AccessKey={accessKey}
```

The response body is similar to the one in the following example:

```
[
   {
      "Carlos Martins":98
   },
   (...)
   {
      "Rui Costa":3
   },
   {
      "Lionel Messi":1
   }
]
```

5.1.6 Get number of times that two personalities co-occur

To get the number of co-occurrences of two given personalities you can use the **GetCoOccurrencesTrends** operation.

In order to do this search you have to do a GET request to the following URI: https://services.sapo.pt/InformationRetrieval/Verbetes/GetCoOccurrencesTrends. You should specify in the query string the two personality's names (**name1** and **name2**) and the date interval of the search (**begin_date** and **end_date**).

The following example is a sample request to get the number of co-ocurrences of the two given personalities:

```
GET
https://services.sapo.pt/InformationRetrieval/Verbetes/GetCoOccurrencesTren
ds?name1={name1}&name2={name2}&begin_date=2012-09-10&end_date=2012-09-
17&format=json&ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "edges":[
      {
         "source":"André Almeida",
         "target":"Javi García",
         "edge_frequency":21
      },
      {
         "source":"Carlos Martins",
         "target":"Jorge Jesus",
         "edge_frequency":88
      },
      (...)
   ],
   "endDate":"2012-09-17",
   "beginDate":"2012-09-01",
   "minFrequencyEdges":"20",
   "nodes":{
      "André Almeida":107,
      "Carlos Martins":240,
      (...)
   },
   "depth":"1.5",
   "ego":"Jorge Jesus"
}
```

6 Auto Service

The **Auto** service allows search of auto vehicles in the contents of **SAPO Auto** (http://auto.sapo.pt/).

To interact with the service is available a HTTP interface with both JSON and XML formats.

6.1 HTTP

This section describes the HTTP-JSON interface of the **Auto** service.

6.1.1 Search by terms

To search for auto vehicles by given terms you can use **JSON** operation freely specifying the terms in the **q** query string parameter.

Furthermore you can specify other query string parameters:

- **rows** – number of results per page;
- **start** – start result’s index;
- **sort** – type of sort;

The following example is a sample request to get the auto vehicles whose name starts with “brav”. The results should be ordered by price in descending order, starting in the index 1 and showing ten results per page.

```
GET
https://services.sapo.pt/Auto/JSON?q=brav*&start=1&rows=10&sort=Price+desc&
ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "rss":{
      "channel":{
         "copyright":"@2012 PT.COM",
         "description":"Auto Sapo Vehicles RSS Feed",
         "item":[
            {
               "category":[
                  "Carro",
                  "Peças/Acessórios"
               ],
               "enclosure":{
                  "length":"1",
                  "type":"image/jpeg",
                  "url":"http://ftbs.sl.pt/auto/media/{...}.jpg"
               },
               "guid":{
                  "isPermaLink":"true",
                  "value":"http://auto.sapo.pt/{...}.aspx"
               },
               "link":"http://auto.sapo.pt/{...}.aspx",
               "pubDate":"Wed, 22 Aug 2012 22:26:00 GMT",
               "title":"Fiat Brava 1.4 SX (80cv) (5 lug) (3p) 1000000.0"
            },
             (...)
            {
               "category":[
                  "Carro",
                  "Utilitário"
               ],
               "enclosure":{
                  "length":"1",
                  "type":"image/jpeg",
                  "url":"http://ftbs.sl.pt/auto/media/{...}.jpg"
               },
               "guid":{
                  "isPermaLink":"true",
                  "value":"http://auto.sapo.pt/{...}.aspx"
               },
               "link":"http://auto.sapo.pt/{...}.aspx",
               "pubDate":"Wed, 26 Oct 2011 16:15:00 GMT",
               "title":"Fiat Bravo 1.6 Mjet 16v Sport (120cv) (5 lug) (5p) 16500.0"
            }
         ],
         "language":"pt-pt",
         "link":"http://services.sapo.pt/Auto/RSS",
         "openSearch:itemsPerPage":"10",
         "openSearch:startIndex":"1",
         "openSearch:totalResults":"95",
         "title":"Auto Sapo",
         "ttl":"60"
      },
      "version":"2.0",
      "xmlns:openSearch":"http://a9.com/-/spec/opensearchrss/1.0/"
   }
}
```

6.1.2 Search by brand, model, price

To search for auto vehicles by brand, model or price you can use the **JSON** operation.

To do this search you can specify in the **q** query string parameter the brand, model or price with the following syntax: Brand:{brand}+Model:{model}+Price:{price}.

Furthermore you can specify other query string parameters:

- **rows** – number of results per page;
- **start** – start result's index;
- **sort** – type of sort;

The following example is a sample request to get the auto vehicles whose **brand** is fiat, **model** is bravo and **price** is 2500.

```
GET
https://services.sapo.pt/Auto/JSON?q=Brand:fiat+Model:bravo+Price:2500&star
t=0&rows=10&sort=Price+desc&ESBUsername={username}&ESBPassword={pass}
HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "rss":{
      "channel":{
         "copyright":"@2012 PT.COM",
         "description":"Auto Sapo Vehicles RSS Feed",
         "item":{
            "category":[
               "Comercial",
               "Ligeiro de Mercadorias"
            ],
            "enclosure":{
               "length":"1",
               "type":"image/jpeg",
               "url":"http://ftbs.sl.pt/auto/media/{...}.jpg"
            },
            "guid":{
               "isPermaLink":"true",
               "value":"http://auto.sapo.pt/{...}.aspx"
            },
            "link":"http://auto.sapo.pt/{...}.aspx",
            "pubDate":"Sun, 30 Sep 2012 22:56:00 GMT",
            "title":"Fiat Bravo TD100 Bravo TD100 2500.0"
         },
         "language":"pt-pt",
         "link":"http://services.sapo.pt/Auto/RSS",
         "openSearch:itemsPerPage":"10",
         "openSearch:startIndex":"0",
         "openSearch:totalResults":"1",
         "title":"Auto Sapo",
         "ttl":"60"
      },
      "version":"2.0",
      "xmlns:openSearch":"http://a9.com/-/spec/opensearchrss/1.0/"
   }
}
```

7 PunyURL Service

The **PunyURL** service allows to, given an URI, obtain two compressed URI's: One in Unicode based on RFC3492 of PunyCode and other with alphanumeric characters in lowercase.

To interact with the service is available a HTTP interface with both JSON and XML formats.

7.1 HTTP

This section describes the HTTP-JSON interface of the **PunyURL** service.

7.1.1 Get compressed URI

To get a compressed version of a given URI you can use the **GetCompressedURLByURLJSON** operation.

To get the compressed version of a given URI you do a GET request to the following URI specifying the URI to be compressed in the **url** query string parameter: https://services.sapo.pt/PunyURL/GetCompressedURLByURLJSON.

The following example is a sample request to get the compressed version of a given URI:

```
GET
https://services.sapo.pt/PunyURL/GetCompressedURLByURLJSON?url=http://www.g
oogle.pt&ESBUsername={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "punyURL":{
      "ascii":"http://b.lb.sl.pt",
      "preview":"http://b.lb.sl.pt /-",
      "puny":"http://溯.sl.pt",
      "url":{
         "cdata":"http://www.google.pt"
      },
      "xmlns":"http://services.sapo.pt/Metadata/PunyURL"
   }
}
```

7.1.2 Get URI from compressed URI

To get the URI that corresponds to a given compressed URI you can use the **GetURLByCompressedURL** operation.

To get the URI that corresponds to the given compressed URI you should do a GET request to the following URI specifying the compressed version of the URI in the **url** query string parameter: https://services.sapo.pt/PunyURL/GetURLByCompressedURL?json=true.

The following example is a sample request to get the URI from a compressed version of the URI:

```
GET
https://services.sapo.pt/PunyURL/GetURLByCompressedURL?url=http://b.lb.sl.p
t&ESBUsername={username}&ESBPassword={pass}&json=true HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "punyURL":{
      "ascii":"http://b.lb.sl.pt",
      "preview":"http://b.lb.sl.pt/-",
      "puny":"http://溯.sl.pt",
      "url":"http://www.google.pt/"
   }
}
```

8 Captcha Service

The **Captcha** service allows the generation and presentation of a customizable captcha.

To interact with the service is available a HTTP-XML interface.

8.1 HTTP

This section describes the HTTP-XML interface of the **Captcha** service.

You can get captchas in audio or image formats. And note that for the same captcha you can present it as an image and as an audio.

8.1.1 Get captcha image

The process of obtaining an image captcha requires two different steps. First you have to generate the captcha using the **Get** operation. Then you can obtain the captcha image using the **Show** operation and specifying the **id** of the generated captcha.

8.1.1.1 Generate captcha

To generate a captcha you do a GET request to the following URI: https://services.sapo.pt/Captcha/Get. Optionally you can specify the captcha's length, mode (e.g. numeric, mixed) and ttl, through **length**, **mode** and **ttl** query string parameters.

The following request is a sample request to generate a captcha in numeric mode with a ttl of 60 seconds and a length of 7 chars:

```
GET
https://services.sapo.pt/Captcha/Get?ttl=60&length=7&mode=numeric&ESBUserna
me={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Captcha xmlns="http://services.sapo.pt/Metadata/Captcha">
   <id>5f62(...)88de</id>
   <code>42933845</code>
   <msg>ok</msg>
</Captcha>
```

8.1.1.2 Get captcha image

To get an image representation for a previously generated captcha do a GET request to the following URI specifying the captcha **id**: http://services.sapo.pt/Captcha/Show.

The following example is a sample request to get an image representation for the captcha. Note that in this example the **ESBAccessKey** is specified in the URI. This way you can use for instance an **img** HTML tag to show the captcha.

```
GET http://services.sapo.pt/Captcha/Show?id=5f62(...)88de
&ESBUsername={username}&ESBPassword={pass}&ESBAccessKey={accessKey}
HTTP/1.1
Host: services.sapo.pt
```

Optionally you can specify other parameters in the query string:

- **font** – font type (e.g. Courier_New.ttf);
- **textcolor** – text color (e.g. ff0000);
- **size** – font size in pixels;
- **background** – background color (e.g. 0000ff).

The request's response has the captcha's image file.

8.1.2 Get captcha audio

The process of obtaining an audio captcha requires two diferent steps. First you have to generate the captcha using the **Get** operation. Then you can obtain the captcha audio using the **Play** operation and specifying the **id** of the generated captcha.

8.1.2.1 Generate captcha

To generate a captcha you do a GET request to the following URI: https://services.sapo.pt/Captcha/Get. Optionally you can specify the captcha's length, mode (e.g. numeric, mixed) and ttl, through **length**, **mode** and **ttl** query string parameters.

The following request is a sample request to generate a captcha in numeric mode with a ttl of 60 seconds and a length of 7 chars:

```
GET
https://services.sapo.pt/Captcha/Get?ttl=60&length=7&mode=numeric&ESBUserna
me={username}&ESBPassword={pass} HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Captcha xmlns="http://services.sapo.pt/Metadata/Captcha">
   <id>5f62(...)88de</id>
   <code>42933845</code>
   <msg>ok</msg>
</Captcha>
```

8.1.2.2 Get captcha audio

To get an audio representation for a previously generated captcha do a GET request to the following URI specifying the captcha **id**: http://services.sapo.pt/Captcha/Play.

The following example is a sample request to get an audio representation for the captcha. Note that in this example the **ESBAccessKey** is specified in the URI. This way you can use for instance an **audio** HTML tag to show the captcha.

```
GET http://services.sapo.pt/Captcha/Play?id=5f62(...)88de
&ESBUsername={username}&ESBPassword={pass}&ESBAccessKey={accessKey}
HTTP/1.1
Host: services.sapo.pt
```

The request's response has the captcha's audio file.

9 Holiday Service

The **Holiday** service returns a list of national, regional and municipal holidays for a given year. Supports calculation of holidays for years between 1582 and 2299.

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

9.1 HTTP

This section describes the HTTP-JSON interface of the **Holiday** service.

9.1.1 Get all holidays

To get all holidays of a given year you have to do a GET request specifying the year like the one in the following example:

```
GET
http://services.sapo.pt/Holiday/GetAllHolidays?year=2012&ESBUsername={usern
ame}&ESBPassword=jsonText=false HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "GetAllHolidaysResponse":{
      "GetAllHolidaysResult":{
         "Holiday":[
            {
               "Date":"2012-12-01T00:00:00",
               "Description":"A Restauração da Independência é a
instauração da casa de Bragança face à dinastia filipina em 1 de dezembro
de 1640, que durante 60 anos regeu o país. É comemorada anualmente em
Portugal por um feriado no dia 1 de Dezembro.",
               "Name":"Restauração da Independência",
               "Type":"National"
            },
            {
               "Date":"2012-12-11T00:00:00",
               "Description":null,
               "Municipality":{
                  "Id":"0811",
                  "Name":"Portimão"
               },
               "Name":"Feriado Municipal",
               "Type":"Municipal"
            },
            (...)
         ]
      }
   }
}
```

9.1.2 Get Carnival

To get the **Carnival** holiday of a given year you have to do a GET request specifying the year like the one in the following example:

```
GET
http://services.sapo.pt/Holiday/GetCarnival?year=2012&ESBUsername={username
}&ESBPassword=jsonText=false HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "GetCarnivalResponse":{
      "GetCarnivalResult":{
         "Date":"2012-02-21T00:00:00",
         "Description":"Feriado facultativo, sendo rara a sua não
utilização na prática. A data tem origem na tradição pagã de celebrar o
final do inverno e foi depois adaptada pela Igreja Católica marcando agora
o período de 40 dias antes da Semana Santa (Quaresma), ou 47 dias antes da
Páscoa, sendo conhecido também por Entrudo",
         "Name":"Carnaval",
         "Type":"Optional"
      }
   }
}
```

9.1.3 Get Corpus Christi

To get the **Corpus Christi** holiday of a given year you have to do a GET request specifying the year like the one in the following example:

```
GET
http://services.sapo.pt/Holiday/GetCorpusChristi?year=2012&ESBUsername={use
rname}&ESBPassword=jsonText=false HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "GetCorpusChristiResponse":{
      "GetCorpusChristiResult":{
         "Date":"2012-06-07T00:00:00",
         "Description":"Corpus Christi (latim para Corpo de Cristo) é uma
festa móvel da Igreja Católica que celebra a presença de Cristo na
Eucaristia. É realizada na quinta-feira seguinte ao domingo da Santíssima
Trindade. É uma festa de \u0027preceito\u0027, isto é, para os católicos é
de comparecimento obrigatório assistir à Missa neste dia, na forma
estabelecida pela Conferência Episcopal do país respectivo",
         "Name":"Corpo de Deus",
         "Type":"National"
      }
   }
}
```

9.1.4 Get Easter

To get the **Easter** holiday of a given year you have to do a GET request specifying the year like the one in the following example:

```
GET
http://services.sapo.pt/Holiday/GetEaster?year=2012&ESBUsername={username}&
ESBPassword=jsonText=false HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "GetEasterResponse":{
      "GetEasterResult":{
         "Date":"2012-04-08T00:00:00",
         "Description":"Primeiro Domingo após a primeira lua cheia que se
verificar a partir de 21 de Março. Sendo celebrado a um Domingo não é
classificado como feriado oficial. A Páscoa (do hebraico Pessach,
significando passagem) é um evento religioso cristão, normalmente
considerado pelas igrejas ligadas a esta corrente religiosa como a maior e
a mais importante festa da cristandade. Na Páscoa os cristãos celebram a
Ressurreição de Jesus Cristo (Vitória sobre a morte) depois da sua morte
por crucificação (ver Sexta-Feira Santa) que teria ocorrido nesta altura do
ano em 30 ou 33 d.C.",
         "Name":"Páscoa",
         "Type":"Religious"
      }
   }
}
```

9.1.5 Get Good Friday

To get the **Good Friday** holiday of a given year you have to do a GET request specifying the year like the one in the following example:

```
GET
http://services.sapo.pt/Holiday/GetGoodFriday?year=2012&ESBUsername={userna
me}&ESBPassword=jsonText=false HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "GetGoodFridayResponse":{
      "GetGoodFridayResult":{
         "Date":"2012-04-06T00:00:00",
         "Description":"A Sexta-feira Santa é a Sexta-feira antes do
Domingo de Páscoa. Em algumas localidades este feriado pode ser celebrado
noutra data na época da Páscoa de acordo com a tradição local. É a data em
que os cristãos lembram o julgamento, paixão, crucificação, morte e
sepultura de Jesus Cristo, através de diversos ritos religiosos",
         "Name":"Sexta-Feira Santa",
         "Type":"National"
      }
   }
}
```

9.1.6 Get local holidays

To get the local holidays of a given year you have to do a GET request specifying the year like the one in the following example:

```
GET
http://services.sapo.pt/Holiday/GetLocalHolidays?year=2012&ESBUsername={use
rname}&ESBPassword=jsonText=false HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "GetLocalHolidaysResponse":{
      "GetLocalHolidaysResult":{
         "Holiday":[
            {
               "Date":"2012-01-11T00:00:00",
               "Description":null,
               "Municipality":{
                  "Id":"1012",
                  "Name":"Óbidos"
               },
               "Name":"Feriado Municipal",
               "Type":"Municipal"
            },
            (...)
         ]
      }
   }
}
```

9.1.7 Get national holidays

To get the national holidays of a given year you have to do a GET request specifying the year like the one in the following example:

```
GET
http://services.sapo.pt/Holiday/GetNationalHolidays?year=2012&ESBUsername={
username}&ESBPassword=jsonText=false HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "GetNationalHolidaysResponse":{
      "GetNationalHolidaysResult":{
         "Holiday":[
            {
               "Date":"2012-06-10T00:00:00",
               "Description":"O Dia de Portugal; Dia de Camões, de Portugal
e das Comunidades Portuguesas, é o dia em que se assinala a morte de Luís
Vaz de Camões a 10 de Junho de 1580, e é também o Dia Nacional de
Portugal.",
               "Name":"Dia de Camões, de Portugal e das Comunidades
Portuguesas",
               "Type":"National"
            },
            (...)
         ]
      }
   }
}
```

9.1.8 Get regional holidays

To get the regional holidays of a given year you have to do a GET request specifying the year like the one in the following example:

```
GET
http://services.sapo.pt/Holiday/GetRegionalHolidays?year=2012&ESBUsername={
username}&ESBPassword=jsonText=false HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response body is similar to the one in the following example:

```
{
   "GetRegionalHolidaysResponse":{
      "GetRegionalHolidaysResult":{
         "Holiday":[
            {
               "Date":"2012-05-28T00:00:00",
               "Description":"Dia da Região Autónoma dos Açores.",
               "Name":"Segunda-Feira do Espírito Santo",
               "Type":"Regional"
            },
            {
               "Date":"2012-12-26T00:00:00",
               "Description":null,
               "Name":"Feriado Regional da Região Autónoma da Madeira",
               "Type":"Regional"
            }
         ]
      }
   }
}
```

10 SMS Service

The SMS Service allows you to send SMS to multiple recipients.

To interact with the service is available a RESTful API that implements the **OneAPI** specification to SMS services RESTful APIs. It's also available a SOAP interface.

10.1 HTTP

This section describes the RESTful API of the SMS service. Because this API implements the **OneAPI** specification the documentation presented in the following URI applys to this API: http://oneapi.gsma.com/sms-restful-api/. But note that in the SAPO SMS Service the **senderName** is mandatory parameter unlike the **OneAPI** specification.

10.1.1 Send SMS

To send a SMS to multiple recipients you do a request like the one in the following example (for more information see the **OneAPI** specification http://oneapi.gsma.com/sms-restful-api/):

```
POST
https://services.sapo.pt/OneAPI/SMS/SMSMessaging/outbound/tel%3A00351900000
000/requests HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 169
Host: services.sapo.pt
Authorization: ESB AccessKey={accessKey}

senderAddress=tel%3A%2B351900000000&senderName=Test&message=hello%20world&a
ddress=tel%3A%2B351900000001,tel%3A%2B351900000002&ESBUsername={username}&E
SBPassword={pass}
```

Note: senderName cannot contain whitespace characters.

The response is similar to the one in the following example:

```
HTTP/1.1 201 Created
Cache-Control: private
Content-Type: application/json; charset=utf-8
Location:
https://services.sapo.pt/OneAPI/SMS/SMSMessaging/outbound/tel:+351900000000
/requests/924
Server: SAPO SDB 3.1
X-Server: MINA
Date: Mon, 15 Oct 2012 13:49:58 GMT
Content-Length: 129

{"resourceReference":
{"resourceURL":"https://services.sapo.pt/OneAPI/SMS/SMSMessaging/outbound/t
el:+351900000000/requests/924"}}
```

10.1.2 Get SMS delivery status

To get SMS delivery status you do a request like the one in the following example (for more information see the **OneAPI** specification http://oneapi.gsma.com/sms-restful-api/):

```
GET
http://services.sapo.pt/OneAPI/SMS/SMSMessaging/outbound/tel%3A%2B351900000
000/requests/924/deliveryInfos?ESBUsername={username}&ESBPassword={pass}
HTTP/1.1
Authorization: ESB AccessKey={accessKey}
Host: services.sapo.pt
```

The response is similar to the one in the following example:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: application/json; charset=utf-8
Server: SAPO SDB 3.1
X-Server: MINA
Date: Mon, 15 Oct 2012 14:30:00 GMT
Content-Length: 301

{"deliveryInfoList":{"deliveryInfo":[{"address":"tel:+351900000001","delive
ryStatus":"DeliveredToTerminal"},{"address":"tel:+351900000002","deliverySt
atus":"DeliveredToTerminal"}],"resourceURL":"http://services.sapo.pt/OneAPI
/SMS/SMSMessaging/outbound/tel:+351900000000/requests/924/deliveryInfos"}}
```

11 MMS Service

The MMS Service allows you to send MMS to multiple recipients.

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

11.1 HTTP

This section describes the HTTP-JSON interface of the **MMS** service.

11.1.1 Send MMS with inline attachments

In order to send a MMS with inline attachments you do a request like the one in the following example:

```
POST
http://services.sapo.pt/OneAPI/MMS/SendMessageInlineAttachments/sendMessage
WithInlineAttachments?ESBUsername={username}&ESBPassword={pass}&json=true
HTTP/1.1
Content-Type: application/json
Content-Length: 448
Host: services.sapo.pt

{"sendMessageWithInlineAttachments":{"subject":"Hello
World","senderAddress":"Test","priority":"Default","format":"MMS","addresse
s":["tel:00351900000001","tel:00351900000002"],"attachments":{"attachment":
[{"type":"text/plain;charset=utf-
8","body":"VGhlIGF0dGFjaG1lbnQgY29udGVudHMgKGJhc2U2NCBlbmNvZGVkKQ==","id":"
<a0@local>"},{"type":"text/plain;charset=utf-
8","body":"VGhlIGF0dGFjaG1lbnQgY29udGVudHMgKGJhc2U2NCBlbmNvZGVkKQ==","id":"
<a1@local>"}]}}}
```

The response is similar to the one in the following example:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: application/json; charset=utf-8
Server: SAPO SDB 3.1
X-Server: MINA
Date: Mon, 15 Oct 2012 16:15:22 GMT
Content-Length: 61

{"sendMessageWithInlineAttachmentsResponse":{"result":"163"}}
```

11.1.2 Get message delivery status

To get the message delivery status you do a request like the one in the following example:

```
GET
http://services.sapo.pt/OneAPI/MMS/SendMessage/getMessageDeliveryStatus?req
uestIdentifier=163&ESBUsername={username}&ESBPassword={pass}&json=true
HTTP/1.1
Host: services.sapo.pt
```

The response is similar to the one in the following example:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: application/json; charset=utf-8
Server: SAPO SDB 3.1
X-Server: LEA
Date: Mon, 15 Oct 2012 16:16:09 GMT
Content-Length: 259

{"getMessageDeliveryStatusResponse":{"result":[{"address":"tel:+35190000000
1","deliveryStatus":"DeliveredToNetwork","description":"DeliveredToNetwork"
},{"address":"tel:+351900000002","deliveryStatus":"DeliveredToNetwork","des
cription":"DeliveredToNetwork"}]}}
```