API Web Service Specification

Version: **Cocoon 2.13.0**

Date: **October 19th, 2018**

Purpose and target audience of this document

This document is a technical description of the Cocoon API Webservice. With this document, your technical staff member (or an employee at your technical partner) can implement independently. The API works over HTTPS request.

Contents

1 Introduction 3
1.1 Service namespace 3
1.2 WSDL definition 3
2 Basic Reference 4
2.1 Sample Soap request and response 4
2.2 DateTime format 5
3 Authentication 6
3.1 Soap Header 7
3.2 Hash 8
3.3 Code Example 9
4 Operations 10
4.1 getVersion 10
4.2 getThumbtypes 11
4.3 getSets 12
4.4 getSet 13
4.5 getFilesBySet 14
4.6 getFile 15
4.7 getTags 16
4.8 getTag 17
4.9 getFilesByTag 18
5 Parameters 19
5.1 aThumbtypes 19
5.2 aSets 19
5.3 aSet 19
5.4 aFiles 19
5.5 aFile 20
5.6 aTags 21
5.7 aTag 21
6 Embedding files 22
6.1 File url structure 22
6.2 Code Example 23

1 Introduction

The purpose of this document is to specify the operations available in the Cocoon Web Service. This document enables developers to create client software.

1.1 Service namespace

The service default namespace is:

{cocoon url}/webservice/

Example: https://demo.use-cocoon.com/webservice/

1.2 WSDL definition

The service WSDL file can be found at:

{cocoon url}/webservice/wsdl/

Example: https://demo.use-cocoon.com/webservice/wsdl/

2 Basic Reference

The Cocoon web service uses the SOAP protocol version 1.1.

2.1 Sample Soap request and response

A sample SOAP request and response message is added. The sample shows the literal text as it is sent via the HTTP protocol.

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns1="https://demo.use-cocoon.com/webservice/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
    <env:Body>
        <ns1:getVersion env:encodingStyle="http://www.w3.org/2003/05/soap-
encoding"/>
    </env:Body>
</env:Envelope>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns1="https://demo.use-cocoon.nl/webservice/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
    <env:Body xmlns:rpc="http://www.w3.org/2003/05/soap-rpc">
        <ns1:getVersionResponse
env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
            <rpc:result>return</rpc:result>
            <return xsi:type="xsd:string">2.4.1</return>
        </ns1:getVersionResponse>
    </env:Body>
</env:Envelope>
```

2.2 DateTime format

The Cocoon web service returns the dateTime datatype value as:

YYYY-MM-DD hh:mm:ss

Where:

YYYY = four-digit year
MM = two-digit month
DD = two-digit day of month
hh = two digits of hour
mm = two digits of minute
ss = two digits of second

Example: 2015-03-03 09:26:33

Note that this does not fully meet the ISO 8601 standard.

3 Authentication

The following info is needed for authentication:

Name	Example	Description
Cocoon identifier	demo	The identifier of the Cocoon. This is always equal to the subdomain of the cocoon. So if the cocoon domain is “https://demo.use-cocoon.com” the identifier is “demo”.
Username	harrie	The webservice username as provided by Cocoon
Secret key	XEmBhX,r*1YGOjSu?xRqk=Ntrlo)@2b~	The webservice sercret key as provided by Cocoon

3.1 Soap Header

Each request contains a header for authentication

Request parameters:

Name	Datatype	Description
authenticate	Core_Model_Webservice_Type_Auth	The authentication object

Core_Model_Webservice_Type_Auth

Each header contains an authenticate element which is used to authenticate the request and validate the user. All elements in this section are required

Elements:

Name	Datatype	Description
username	string	Username as provided by Cocoon
requestId	string	A unique request identifier
hash	string	Hash for request authentication

3.2 Hash

The request hash is calculated as follows:

- Concatenate a string from Coocn identifier, username, request identifier and secret key consecutively.

-Calculate a SHA1 hash over the obtained string.

An example of the calculation of the hash:

Cocoon identifier	demo
username	harrie
request identifier	1
secret key	XEmBhX,r*1YGOjSu?xRqk=Ntrlo)@2b~
Concatenated string	demoharrie1XEmBhX,r*1YGOjSu?xRqk=Ntrlo)@2b~
SHA1 hash	4f4034a514887737436459005c81fc7df372a4ef

A demonstration in php of how the hash can be calculated:

```
$cocoonId         = 'demo';
$username         = 'harrie';
$requestId        = '1';
$secretKey        = 'XEmBhX,r*1YGOjSu?xRqk=Ntrlo)@2b~';
$concatString     = $cocoonId . $username . $requestId . $secretKey;
$hash             = sha1($concatString);
```

3.3 Code Example

An example in php of how the soap header can be created

```
<?php
//params
$wsdl       = 'https://demo.use-cocoon.com/webservice/wsdl/';
$bb         = 'demo';
$user       = 'harrie'; //Provided by Cocoon
$requestId  = (string)microtime(true); //Something unique
$sk         = 'XEmBhX,r*1YGOjSu?xRqk=Ntrlo)@2b~'; //Provided by Cocoon

//create hash
$hash = sha1("$bb$user$requestId$sk");

//Auth object
$oAuth              = new stdClass;
$oAuth->username    = $user;
$oAuth->requestId   = $requestId;
$oAuth->hash        = $hash;

//Init client
$oSoapClient = new SoapClient($wsdl);
$oSoapClient->__setSoapHeaders(new SoapHeader('auth','authenticate',$oAuth));

try
{
    var_dump($oSoapClient->getVersion());
}
catch(SoapFault $oSoapFault)
{
    var_dump($oSoapFault->faultcode,$oSoapFault->getMessage());
}
```

4 Operations

This chapter lists all operations made available by the web service. A technical definition can be found in the WSDL file. The section titles correspond with the operation names.

4.1 getVersion

Returns the Cocoon version.

Request parameters:

Name	Datatype	Description

Response parameters:

Name	Datatype	Description
version	string	The Cocoon version.

Resultcodes

ResultCode	Datatype	Description
0	integer	OK
901	integer	Authentication Error
902	integer	Technical Failure

4.2 getThumbtypes

Returns all thumbtypes their info.

Request parameters:

Name	Datatype	Description

Response parameters:

Name	Datatype	Description
aThumbtypes	array	All thumbtypes their info. See aThumbtypes

Resultcodes

ResultCode	Datatype	Description
0	integer	OK
901	integer	Authentication Error
902	integer	Technical Failure

4.3 getSets

Returns all public sets.

Request parameters:

Name	Datatype	Description

Response parameters:

Name	Datatype	Description
aSets	array	All public sets. See aSet

Resultcodes

ResultCode	Datatype	Description
0	integer	OK
901	integer	Authentication Error
902	integer	Technical Failure

4.4 getSet

Returns a public set by set id.

Request parameters:

Name	Datatype	Description
setId	int	Set id.

Response parameters:

Name	Datatype	Description
aSet	array	The public set. see aSet

Resultcodes:

ResultCode	Datatype	Description
0	integer	OK
100	integer	Set not found
101	integer	Set not public
901	integer	Authentication Error
902	integer	Technical Failure

4.5 getFilesBySet

Returns files by public set id.

Request parameters:

Name	Datatype	Description
setId	int	Set id.

Response parameters:

Name	Datatype	Description
aFiles	array	All files in the public set. see aFiles

Resultcodes:

ResultCode	Datatype	Description
0	integer	OK
100	integer	Set not found
101	integer	Set not public
901	integer	Authentication Error
902	integer	Technical Failure

4.6 getFile

Returns a file in a public set by file id.

Request parameters:

Name	Datatype	Description
fileId	int	File id.

Response parameters:

Name	Datatype	Description
aFile	array	The file in a public set. See aFile

Resultcodes:

ResultCode	Datatype	Description
0	integer	OK
100	integer	File not found
101	integer	File not in public set
901	integer	Authentication Error
902	integer	Technical Failure

4.7 getTags

Returns all tags.

Request parameters:

Name	Datatype	Description

Response parameters:

Name	Datatype	Description
aTags	array	All tags. See aTags

Resultcodes

ResultCode	Datatype	Description
0	integer	OK
901	integer	Authentication Error
902	integer	Technical Failure

4.8 getTag

Returns a tag by tag id.

Request parameters:

Name	Datatype	Description
tagId	int	Tag id.

Response parameters:

Name	Datatype	Description
aTag	array	The tag. See aTag

Resultcodes:

ResultCode	Datatype	Description
0	integer	OK
100	integer	Tag not found
901	integer	Authentication Error
902	integer	Technical Failure

4.9 getFilesByTag

Returns files (that are in at least one public set) by tag id.

Request parameters:

Name	Datatype	Description
tagId	int	Tag id.

Response parameters:

Name	Datatype	Description
aFiles	array	All files (that are in at least one public set) with the tag. see aFiles

Resultcodes:

ResultCode	Datatype	Description
0	integer	OK
100	integer	Tag not found
901	integer	Authentication Error
902	integer	Technical Failure

5 Parameters

This section specifies some of the parameters available in the web service.

5.1 aThumbtypes

An array containing details about the thumbtypes.

Elements:

Name	Datatype	Description
width	string	The width the thumbnail of this type is scaled to. Note that thumbnails are scaled within their aspect ratio.
height	string	The height the thumbnail of this type is scaled to. Note that thumbnails are scaled within their aspect ratio.
path	string	The path to the thumbtype files.

5.2 aSets

An array containing 0 or more aSet arrays. Note that the array can be empty when - based on the operation's request parameters - no public set is found.

5.3 aSet

An array containing details about the public set. Note that the array can be empty when - based on the operation's request parameters - no public set is found.

Elements:

Name	Datatype	Description
id	int	Set id.
title	string	Set title.
date	dateTime	The date and time the set was inserted or last edited.
file_count	int	The number of files in the set.

5.4 aFiles

An array containing 0 or more aFile arrays. Note that the array can be empty when - based on the operation's request parameters - no file is found.

5.5 aFile

An array containing details about the file in a public set. Note that the array can be empty when - based on the operation's request parameters - no file is found.

Elements:

Name	Datatype	Description
id	int	File id.
title	string	File title.
filename	string	The original filename.
extension	string	The original file extension.
width	int	The original file with in pixels, if available. *
height	int	The original file height in pixels, if available. *
size	int	The original file size in bytes, if available. *
custom_field_1	string	The value of the first custom field.
custom_field_2	string	The value of the second custom field.
custom_field_3	string	The value of the third custom field.
custom_field_4	string	The value of the fourth custom field.
custom_field_5	string	The value of the fifth custom field.
status	string	The status of the file.
upload_date	dateTime	The date and time the file was uploaded.
upload_by	int	The user id of the user who uploaded the file.

* = Only available for files uploaded in the Cocoon version 2.0.2 and onward. The value will be NULL when unavailable.

5.6 aTags

An array containing 0 or more aTag arrays. Note that the array can be empty when - based on the operation's request parameters - no tag is found.

5.7 aTag

An array containing details about the tag. Note that the array can be empty when - based on the operation's request parameters - no tag is found.

Elements:

Name	Datatype	Description
id	int	Tag id.
name	string	Tag title.
date	dateTime	The date and time the tag was inserted or last edited.

6 Embedding files

Trough the embed module it is possible to make sets public. Files in these sets can be accessed publicly.

In order to do so the embed module must first be activate.

6.1 File url structure

The file url structure for accessing files in public sets is as follows:

https://{cocoon_url}{thumbtype_path}/{filename}.{thumbtype_extention}

Url parts:

Name	Example	Description
cocoon_url	demo.use-cocoon.nl	The Cocoon domain
thumbtype_path	/files/web	Path to the desired thumbtype as defined by the aThumbtypes parameter
filename	logo	Filename of the desired file as defined by the aFile parameter
thumbtype_extention	jpg	Extention of the desired thumbtype. For the original thumbtype the extention is as defind by the aFile parameter. For other thumbtypes the extention is gif or png when the original extention is gif or png. Jpg otherwise

Please note that we strongly advise to cache files locally on the first call and serve them from the local cache from there on. Cocoon does not guarantee any performance for embedded files.

6.2 Code Example

An example in php of how the file url can be constructed

```
<?php
$url                    = 'https://demo.use-cocoon.com';
$thumbType              = 'web';
$fileId                 = 1337;

//get thumbtype path
$aThumbTypes            = $oSoapClient->getThumbtypes();
$thumbTypePath          = $aThumbTypes[$thumbType]['path'];

//get filename
$aFile                  = $oSoapClient->getFile($fileId);
$filename               = $aFile['filename'];

//get thumbtype extention
$extention              = $aFile['extention'];

if($thumbType == 'original')
{
      $thumbTypeExtention = $extention;
}
//thumbType != original &&
elseif($thumbType == 'gif' || $thumbType == 'png')
{
      $thumbTypeExtention = $extention;
}
//thumbType != original && ($thumbType != 'gif' && $thumbType != 'png')
else
{
      $thumbTypeExtention = 'jpg';
}

//construct the file url
$fileUrl = $url . $thumbTypePath . '/' . $filename . '.' . $thumbTypeExtention;
```