

API Web Service Specification

**Version: 1.3**

**Date: October 3th, 2017**

**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](#__RefHeading__1281_1939077943)

[1.1 Service namespace 3](#__RefHeading___Toc3035_914277716)

[1.2 WSDL definition 3](#__RefHeading___Toc3037_914277716)

[2 Basic Reference 4](#__RefHeading___Toc3039_914277716)

[2.1 Sample Soap request and response 4](#__RefHeading___Toc3041_914277716)

[2.2 DateTime format 5](#__RefHeading___Toc3043_914277716)

[3 Authentication 5](#__RefHeading___Toc3045_914277716)

[3.1 Soap Header 6](#__RefHeading___Toc3047_914277716)

[3.2 Hash 7](#__RefHeading___Toc3049_914277716)

[3.3 Code Example 8](#__RefHeading___Toc3051_914277716)

[4 Operations 9](#__RefHeading___Toc3053_914277716)

[4.1 getVersion 9](#__RefHeading___Toc3055_914277716)

[4.2 getThumbtypes 10](#__RefHeading___Toc3057_914277716)

[4.3 getSets 11](#__RefHeading___Toc3059_914277716)

[4.4 getSet 12](#__RefHeading___Toc3061_914277716)

[4.5 getFilesBySet 13](#__RefHeading___Toc3063_914277716)

[4.6 getFile 14](#__RefHeading___Toc3065_914277716)

[5 Parameters 15](#__RefHeading___Toc3067_914277716)

[5.1 aThumbtypes 15](#__RefHeading___Toc3069_914277716)

[5.2 aSets 15](#__RefHeading___Toc3071_914277716)

[5.3 aSet 15](#__RefHeading___Toc3073_914277716)

[5.4 aFiles 15](#__RefHeading___Toc3075_914277716)

[5.5 aFile 16](#__RefHeading___Toc3077_914277716)

[6 Embedding files 17](#__RefHeading___Toc1621_1482556115)

[6.1 File url structure 17](#__RefHeading___Toc1623_1482556115)

[6.2 Code Example 18](#__RefHeading___Toc1627_1482556115)

#  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.

##  Service namespace

The service default namespace is:

{cocoon url}/webservice/

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

##  WSDL definition

The service WSDL file can be found at:

{cocoon url}/webservice/wsdl/

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

#  Basic Reference

The Cocoon web service uses the SOAP protocol version 1.1.

##  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>

##  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.

#  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 |

##

##  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 |

##  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);

##  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());

}

#  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.

##  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 |
| 1 | integer | Authentication Error |
| 2 | integer | Technical Failure |

##

##  getThumbtypes

Returns all thumbtypes their info.

**Request parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Datatype | Description |
|  |  |  |

**Response parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Datatype | Description |
| aThumbtypes | array | All thumbtypes their info.See [aThumbtypes](#_aThumbtypes) |

**Resultcodes**

|  |  |  |
| --- | --- | --- |
| ResultCode | Datatype | Description |
| 0 | integer | OK |
| 1 | integer | Authentication Error |
| 2 | integer | Technical Failure |

##

##  getSets

Returns all public sets.

**Request parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Datatype | Description |
|  |  |  |

**Response parameters:**

|  |  |  |
| --- | --- | --- |
| Name | Datatype | Description |
| aSets | array | All public sets.See [aSet](#_aSet) |

**Resultcodes**

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

##  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](#_aSet) |

**Resultcodes:**

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

##  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](#_aFiles) |

**Resultcodes:**

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

##  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](#_aFile) |

**Resultcodes:**

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

#  Parameters

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

##  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. |

##  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.

##  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. |

##  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.

##  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.

#  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.

##  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.

##  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;