API Description

Working with the Uniplace system API

Contents

General provisions

The protocol is designed to automate part of the client's actions in the Uniplace system.

This document describes the overall structure of the Software interface (API) and the requirements for the functionality it provides.

An integral part of this description is the specification of parameters, formats, signatures that describe the exact implementation of the protocol with the client.

The exchange format

The exchange is based on the client/server model under the Web Service (SOAP) protocol. The Client (requesting) Party is the service of the client, the Server (responding) is the Uniplace service.

Thus, any operation is a "request"-"response" pair.

Restrictions on connecting to the Uniplace API service. The authentication process

Note: All tokens are submitted and accepted in lowercase. Email and password should also be lowercase before hashing.

For the service authentication, you should

1. Call the *Login* method by passing the *loginToken* as a parameter: ***Md5 hash*** from a string composed of your login (*email*), password and "*salt*" - the string **"yourqlinkapi"**. When you call this method, api gives you a *random Token* (a string of 32 characters).
2. Use the same operation for the resulting *random token* — the string <*random_token*>+ "**Yourqlinkapi**" is made and **MD5 hash** is taken from it and as a result of this operation we receive a *token to access the API methods* (hereinafter *MethodToken*).
3. MethodToken is required by api methods that have a corresponding parameter, by which api identifies a particular user.

The limitations imposed on the account are no more than 500 requests per minute to the API.

The limitations imposed on the *methodToken* are the token's lifetime = 4 hours. After that the token is considered invalid.

An outline of an authentication/method call code:

1. Get the authentication token:
```
var loginToken = ("IWebmasterEmail" +"IWebmasterPassword" +
"yourqlinkapi").GetMd5();
```
2. Create an instance of the service proxy class
```
var_api = new WebmasterApi ();
```

3. Get a random token
   ```
   varrandom_token = _api.Login(loginToken);
   ```
4. Modify the resulting random token by obtaining a token to access the methods of
   ```
   var methodToken = (random_token + "yourqlinkapi").GetMd5();
   ```
5. Call the api method by passing the methodToken parameter.
   ```
   var siteDonors = _api. GetListSiteDonor(methodToken);
   ```

If you use the term "composition", the token to access the api methods is

```
var methodToken =
(_api.Login(("your_login " + "yout_password" + "yourqlinkapi").GetMd5()) +
"yourqlinkapi")
.GetMd5();5 ();
```

PHP 5.3. Example.

```php
$pass = "";//Your password
$salt = "yourqlinkapi";//Public salt
//That's strange, but hashing just login and password without the salt is wrong.
$client = new \SoapClient('http://www.uniplace.ru/api/WebmasterApi.svc?singleWsdl', array ('soap_version' => SOAP_1_1));
//Logging in
try {
    $s = $client->Login(array('loginToken' => strtolower(md5(strtolower($login.$pass.$salt)))));
} catch (SoapFault $e) {
    var_dump($e->getMessage());
}
var_dump($s);
//Get a token to access the methods. You can save it. The token expires after 4 hours (240 minutes).
$methodToken = strtolower(md5(strtolower($s->LoginResult."yourqlinkapi")));
var_dump($methodToken);
//Check when the token is refreshed.
$price = $client->CountRenewLeftSeconds(array('methodToken' => $methodToken));
var_dump($price);
// Or check how long the token will remain valid (in minutes).
$lifeTime = $client->TokenLeftTimeMinutes(array('methodToken' => $methodToken));
var_dump($lifeTime);
//Here is the list of sites.
$siteList = $client->GetListSiteDonor(array('methodToken' => $methodToken))
var_dump($siteList);
//Chose site ID (for example, 50472) and got the page list for it.
//Here is the difference – there are many pages, that's why the method returns pages in batches. Batch number - pageNumber
//Numeration starts from 0, as in all data arrays
$pageList = $client->GetListPageDonor(
    array(
        'methodToken' => $methodToken,
        'siteId' => 50472,
        'pageNumber' => 0
    )
);
var_dump($pageList);
//Take the second data page.
$pageList = $client->GetListPageDonor(
    array(
        'methodToken' => $methodToken,
        'siteId' => 50472,
        'pageNumber' => 1
    )
);
var_dump($pageList);
```

Protocol specification

The API service is available at http://www.uniplace.ir/api/. The Web service protocol is SOAP. Possible error codes:

- 401 HttpStatusCode.Unauthorize - the token used to request the API method is unknown. The problem is in incorrectly generated methodToken.
- 403 HttpStatusCode.Forbidden – the token has expired or the limit of requests for the current period has been reached.
- 500 HttpStatusCode.InternalServerError - internal service error.

Agreement on the data types to be used:

1. Standard data types:

Data type	Range
byte	0 .. 255
sbyte	-128 .. 127
short	-32,768 .. 32,767
ushort	0 .. 65,535
int	-2,147,483,648 .. 2,147,483,647
uint	0 .. 4,294,967,295
long	-9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807
ulong	0 .. 18,446,744,073,709,551,615
float	-3,402823e38 .. -3,402823e38 ..
double	-1,79769313486232e308 .. 1,79769313486232e308
decimal	-79228162514264337593543950335 .. 79228162514264337593543950335
char	Unicode symbol
string	Unicode symbol string
bool	true or false

DateTime is a standard type under the ISO standard http://www.w3.org/TR/NOTE-datetime. Example - `YYYY-MM-DD (e.g. 1997-07-16).`

Note: if you use a standard type name as name?, for example, bool? the result may not be an explicit type, but a *null* value.

2. The application data types.

- Enumerations and wrappers:
  - ActSiteDonorResult, it is wrapped in the WebmasterActResult container with the `ActionResult` field

```
/// Result of an operation in api methods for a webmaster.
ActSiteDonorResult {
    /// Operation is successfully completed.
    Success = 0,

    /// Operation failed for an unknown reason.
    Fail = 1,

    /// Operation failed. The site not found.
    FailSiteNotExists = 2,

    /// Operation failed. The page not found.
    FailPageNotExists = 3,

    /// Operation failed. The site already exists.
    FailSiteExists = 4,

    /// Operation failed. The page already exists.
    FailPageExists = 5,

    /// Banned site/Stop word limit is exceeded.
    ToManyItems = 6,
}
```

- WebmasterActResultIdentified is the wrapper on the above mentioned WebmasterActResult, the fields – inherited from WebmasterActResult `ActionResult`, `type` ActSiteDonorResult, and long ID, through which the ID of the entities on which the action was effected was passed.

- Types-Entities:
  - Siteinfo

```
/// Site info
SiteInfo {

    /// Site ID
    long ID;

    /// Site domain
    string Domain;

    /// Confirmed
    bool? Confirmed;
}
```

- PageInfo

```
/// Page info
PageInfo {

    /// Page ID
    long Id;

    /// Page URL
    string URL;

    ///Placement availability status
    bool PlacementStatus;
}
```

- LinkInfo

```
/// Link info
LinkInfo {

    /// ID
    public long ID;

    /// Address
    public string Href;

    /// Link text
    public string Text;
}
```

General note - If you use methods that provide data page by page and receive a page number parameter such as int pagenumber, the numbering starts at 0 as in all data arrays.

The standard number of data per page is 1000 objects, the maximum word length in a blacklist is 350 characters, the maximum number of pages is 1000, the maximum number of items in a blacklist/list of banned sites is 1000.

Interface provided

1. Authentication/Authorization methods and other services:

- Authorization in API:

LoginToken is the authentication token received as an md5 hash from the string <your_login>+ <your_password>+ "yourqlinkapi"

Returns a random token - a string of 32 characters to access methods that require the authorized state. For more information, see "Restrictions on connecting to the Uniplace API service. The Authentication process"

string Login(string loginToken);

- Ping request:

Returns the "UniplaceApi" string. Does not require authorization, makes it possible to check whether the service is running.

string WhoAmI();

- **Echo request:**

Input: an arbitrary string you have passed. Returns the input string. Checking the data communication protocol.

string Echo(string input);

- **The time remaining before the request limit is updated:**

The time remaining before the limit of the number of requests is updated

double CountRenewLeftSeconds(string methodToken);

- **The remaining lifetime of the token:**

The remaining lifetime of the token in minutes.

int TokenLeftTimeMinutes(string methodToken);

2. Finance methods:

- **Accruals for the date on the site:**

Payment for the links on the site with ID siteId for the date

decimal GetPayment(long siteid, DateTime date, string methodToken);

Another option for this method is used if you do not want to work with datetime -

decimal GetPaymentByDate(long siteid, int year, int month, int day, string methodToken)

Note:

- Prior to February 2, 2015, the method provided the value of accrued funds before accounting for the 10% of the exchange fee
- The method provides the real date of the accrual and the accrued funds data are shifted by + 1 day in the interface.

I.e., if you ask GetPaymentByDate(my_site_id 2015,2, 9, my_token), the number must match the one shown in the interface as of 2015-2-10.

3. Site management methods:

- **Get a site hash code by domain**

The unique hash code of your site in our system. This code is used in scripts on the client side.

string GetHashCode(string domain, string methodToken);

- **Update indexing**

Reset of the indexing information for your site causes the site to be re-indexed during a short period.

WebmasterActResult RenewIndexation(string domain, string methodToken);

- **Add a site**

Add a new site "siteUrl". Note: similar to the exchange interface.

siteUrl

WebmasterActResultIdentified AddSiteDonor(string siteUrl, string methodToken);

Add a new site "siteUrl". Note: add sites with synchronization through the dump of links.

WebmasterActResultIdentified AddSiteDonorRawDump (string siteUrl, string methodToken);

- **Delete a site**

Delete a site with the siteid identifier

siteId site identifier

WebmasterActResult RemoveSiteDonor(long siteId, string methodToken);

- **Get list of my sites**

Gets a list of sites that have been added to your account

SiteInfo[] GetListSiteDonor(string methodToken);

4. Manage page indexing:

- **Lists indexed pages for a site**

Paginal receipt of the site pages added to the exchange

siteId site identifier

PageNumber Active page number

PageInfo[] GetListPageDonor(long siteId, int pageNumber, string methodToken);

- **Add pages to the exchange**

Adding pages pageUrls to the exchange of the site siteId

siteId site identifier

pageUrls

WebmasterActResultIdentified[] AddPageDonor(long siteId, string[] pageUrls, string methodToken);

5. Working with purchased links:

- **Get the information on links purchased on the site**

A page-by-page list of the links purchased on the siteId site.

siteId — Donor Site ID

pageNumber — Number of the page

LinkInfo [] GetListLinks (long siteId, int pageNumber, string methodToken);

- **Manual removal of links**

Remove links with IDs in the linkIds list

linkIds IDs of links to be removed

WebmasterActResultIdentified[] RemoveLinks(long[] linkIds, string methodToken);

6. Blacklists:

- **Add a site to the blacklist**

Adding accepting sites banUrls to the list of banned accepting sites of the site with the ID (siteId)

siteId Donor Site ID

banUrls List of sites to be added to the blacklist

WebmasterActResult[] BanSiteAcceptor(long siteId, string[] banUrls, string methodToken);

- **Remove sites from the blacklist**

Removing accepting sites unbanUrls from the list of banned accepting sites of the site with the ID (siteId)

siteId Donor Site ID

unBanUrls URLs of the sites to be removed from the list

WebmasterActResult[] UnBanSiteAcceptor(long siteId, string[] unBanUrls, string methodToken);

- **Get the list of banned sites**

Get the list of accepting sites

siteId Donor Site ID

string[] GetListBanSiteAcceptor(long siteId, string methodToken);

7. Stop words:

- **Add stop words for a site**

Add stop words stopWords for the site with ID siteId

siteId Donor Site ID

stopWords Stop words

WebmasterActResult[] AddStopWords(long siteId, string[] stopWords, string methodToken);

- **Delete stop words for the site**

Remove stop-words stopWords for the site with siteId

siteId Donor Site ID

stopWords List of stop words

WebmasterActResult[] RemoveStopWords(long siteId, string[] stopWords, string methodToken);

- **Get a list of the stop words for the site**

Get a list of stop words for the site with the siteId

siteId Donor Site ID

string[] GetListStopWords(long siteId, string methodToken);

8. Managing the purchase of links on pages:

- **Mark a page as unavailable for purchase**

Prohibit the purchase of links on site pages pageIds with the siteId

siteId Donor Site ID

pageIds Pages

WebmasterActResultIdentified[] DisablePlacementPageDonor(long siteId, long[] pageIds, string methodToken);

- **Mark a page as available for purchase**

Allow the purchase of links on pageIds pages of the site with siteId

siteId Donor Site ID

pageIds Pages

WebmasterActResultIdentified[] EnablePlacementPageDonor(long siteId, long[] pageIds, string methodToken);

- **Calculate the statistical domain price**

Statistical (based on purchasing other sites with similar parameters) calculation of the expected profits from the domain.

decimal PriceCalc(string domain, string methodToken);

Annex 1. A brief description of the API operations

The API involves the authentication by a calculated token. To call any API method that has a methodToken parameter, you should obtain this token by manipulating the API and save it for further queries.

If you are using .Net, it is desirable to generate a Service Reference for the URL http://uniplace.ir/api/and use the provided proxy class

Example. Authentication, retrieving all sites within the account and pages for one of the sites.

```
var api = new QLinkWebmasterApiClient();
var loginToken = ("test@test.ru".ToLower() + "testWebmaster".ToLower() +
ApiLoginData.ApiLoginSalt).GetMD5().ToLower();
var randomToken = api.Login(loginToken);
var methodToken = (randomToken + "yourqlinkapi").GetMD5().ToLower();
var siteDonors = api.GetListSiteDonor(methodToken);
var pagedonors = api.GetListPageDonor(siteDonors[0].ID, 0, methodToken);
Console.WriteLine(pagedonors.First().Url);
```