

Введение

Программный интерфейс (*API*) для сервиса *PROMOpark* дает возможность разработчикам создавать приложения, напрямую взаимодействующие с ним. С помощью таких приложений пользователи могут гибко и эффективно управлять сложными и масштабными проектами.

Для работы с сервисом *PROMOpark* через *API*, необходимо написать клиентское приложение, которое должно загружать и осуществлять разбор файлов с *WSDL*-описанием, формировать *XML*-запросы к сервису, и анализировать полученные ответы. Такое приложение может быть реализовано на любом языке программирования с поддержкой *SOAP*, например *Perl*, *PHP*, *.NET*, *C++*, *Java* и т. д.

Для обмена данными между клиентским приложением (клиентом) и веб-сервисом *PROMOpark* используется протокол *SOAP* (простой протокол доступа к объектам) версии 1.1.

Работа с веб-сервисом строится следующим образом: клиент посылает *SOAP* запрос, веб-сервис его обрабатывает и возвращает ответ. И запросы, и ответы представляют собой *XML*-сообщения. Формат этих сообщений, а также доступные для клиента операции с веб-сервисом определены в файлах *WSDL*, хранящихся на сервере *PROMOpark*.

Для того, чтобы использовать *SOAP* в вашем клиенте, понадобится скачать и установить библиотеку для работы с *SOAP* для выбранного языка программирования. Такая библиотека, как правило, уже содержит все функции, необходимые для того, чтобы читать *WSDL* файлы и обрабатывать *XML* запросы/ответы. Использование *SOAP*-библиотек также позволяет получать от веб-сервиса структуры данных и объекты в естественном для вашего языка виде.

Документ *WSDL* (*Web Services Description Language* — язык описания веб-сервиса) — это *XML*-документ, описывающий интерфейс взаимодействия некоторого ресурса (сервера) и клиентского приложения.

WSDL — машиночитаемый язык, и может быть использован для автоматической генерации кода, что помогает скрыть детали отправки и получения сообщений от ресурса. *WSDL* также гарантирует способность взаимодействовать на уровне описания сервиса вне зависимости от языка программирования.

Адрес WSDL документа: <http://promopark.ru/api/Soap/?wsdl>

Обзор методов API *PROMOpark*

Уровень I

p_Login – Авторизация в API

```
boolean p_Login( LoginPar )
```

Входные параметры:

Структура данных *LoginPar*:

```
{  
    Login => string,  
    Password => string,  
    md5 => boolean  
}
```

LoginPar:

Поле	Тип	Описание
Login	string	Логин пользователя в системе PROMOpark.
Password	string	Пароль пользователя в системе PROMOpark.
md5	boolean	В случае установки флага значение Password должно передаваться в зашифрованном алгоритмом md5 виде. По умолчанию - флаг сброшен.

Возвращает:

Параметр	Тип	Значение
Result	boolean	1 – в случае успешной авторизации, 0 – в случае неудачной авторизации

В дальнейшем, при использовании API, так же необходимо передавать cookies, переданные методом *p_Login*.

p_ProjectList – получение списка активных проектов

```
array p_ProjectList()
```

Входные параметры: отсутствуют
Возвращает:

Массив структур данных *Result*:

```
[  
    ...  
    {ProjectInfo}  
    {ProjectInfo}  
    {ProjectInfo}  
    ...  
]
```

ProjectInfo:

<i>Параметр</i>	<i>Тип</i>	<i>Значение</i>
ProjectID	int	ID проекта в системе PROMOpark
ProjectDomain	string	Имя домена
YaPages	int	Кол-во проиндексированных Яндексом страниц
GoPages	int	Кол-во проиндексированных Google страниц
RaPages	int	Кол-во проиндексированных Rambler страниц
PR	int	Page Rank домена
CY	int	ТИЦ домена

p_GetReport – получение отчета по проекту за конкретную дату

```
struct p_GetReport( ReportInfo )
```

Входные параметры:

Структура данных *ReportInfo*:

```
{  
    ProjectID => int,  
    ReportDate => string  
}
```

ReportInfo:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID запрашиваемого проекта.
ReportDate	string	Дата, за которую необходим отчет. Формат уууу-мм-дд. Например, 2009-01-01

Возвращает:

Структуру данных *Result:*

```
{
  CompleteStatus => int
  SEngines => {
    ...
    ExistsEngine => 1
    ...
  },
  Report => {
    ...
    ExistsEngine => {
      ...
      {ReportInfo}
      ...
    }
    ...
  },
  URL => string
}
```

Result:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
CompleteStatus	int	Процент готовности отчета (от 0 до 100)
SEngines	struct	Структура, ключами которой являются идентификаторы поисковых систем, по которым есть данные в отчете (например: 'ya', 'go' и т.д.)
Report	struct	Структура, ключами которой являются те же идентификаторы, что и в SEngines. Каждый элемент структуры является массивом структур ReportInfo.
URL	string	Уникальный адрес отчета в системе PROMOPark

Структура данных *ReportInfo*:

```
{
    QueryID => int,
    GeoID => int,
    Pos => int,
    Dif => int,
    Url => string,
    LocalCopyKey => string
}
```

ReportInfo:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
QueryID	int	ID запроса
GeoID	int	ID региона
Pos	int	Позиция в выдаче поисковой системы ExistsEngine
Dif	int	Изменение позиции в поисковой системе ExistsEngine относительно предыдущего отчета
Url	string	Адрес документа
LocalCopyKey	string	Адрес локальной копии выдачи поисковой системы по запросу

p_DatesList – получение списка дат, по которым доступны отчеты для проекта

```
array p_DatesList( ProjectID )
```

Входные параметры:

Целочисленная переменная *ProjectID*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID проекта.

Возвращает:

Массив данных *Result*:

```
{
    ...
    Date
    ...
}
```

Result:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
Date	string	Дата в формате уууу-мм-дд

p_QueryList – получение списка запросов в проекте

```
array p_QueryList( ReportInfo )
```

Входные параметры:

Структура данных *ReportInfo*:

```
{  
    ProjectID => int  
}
```

ReportInfo:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID запрашиваемого проекта.

Возвращает:

Массив структур данных *Result*:

```
{  
    ...  
    QueryID => int,  
    Query => string  
    ...  
}
```

Result:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
QueryID	int	ID запроса
Query	string	текст запроса

Уровень II

p_AddProject – добавление проекта

boolean p_AddProject(Domain)

Входные параметры:

Строка *Domain*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
Domain	string	Имя нового проекта

Возвращает:

Логическая переменная *Result*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
Result	int	ID добавленного проекта

p_RemoveProject – удаление проекта

boolean p_RemoveProject(ProjectID)

Входные параметры:

Целочисленная переменная *ProjectID*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID проекта, который необходимо удалить

Возвращает:

Логическая переменная *Result*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
Result	boolean	1 – в случае успешного удаления, 0 – в случае неудачного удаления

p_AddProjectQuery – добавление запросов в проект

boolean p_AddProjectQuery(ProjectID, Query)

Входные параметры:

Целочисленная переменная *ProjectID*, массив *Query*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID проекта, который необходимо удалить
Query	array	Массив добавляемых запросов

Возвращает:

Массив структур *Result*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ID	int	ID запроса в системе PROMOpark
Query	string	текст запроса

p_RemoveProjectQuery – удаление запроса из проекта

boolean p_RemoveProject(ProjectID, Query)

Входные параметры:

Целочисленная переменная *ProjectID*, строковая переменная *Query*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID проекта, который необходимо удалить
Query	int	ID запроса

Возвращает:

Логическая переменная *Result*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
-------------	------------	-----------------

Result	boolean	1 – в случае успешного удаления, 0 – в случае неудачного удаления
--------	---------	--

p_SetProjectSe – установка поисковых систем, по которым будут проверяться позиции

boolean p_SetProjectSe(SE, ProjectID)

Входные параметры:

Массив данных SE, целочисленная переменная *ProjectID*

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID проекта, который необходимо удалить
SE	array	список поисковых систем для проекта (доступные поисковики: "ya", "go", "ra", "ug", "uy")

Возвращает:

Логическая переменная *Result*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
Result	boolean	1 – в случае успешного обновления списка, 0 – в случае неудачного обновления списка

p_GetRegionList – получение списка доступных регионов (по Яндексу)

```
array p_GetRegionList()
```

Возвращает:

Массив структур данных *Result*:

```
{
    ...
    ID => int,
    NAME => string
    ...
}
```

Result:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ID	int	ID региона
NAME	string	Символьное значение

p_GetProjectRegions – получение списка Яндекс регионов для проекта

```
boolean p_GetProjectRegions( ProjectID )
```

Входные параметры:

Целочисленная переменная *ProjectID*

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID проекта

Возвращает:

Массив структур данных *Result*:

```
{
    ...
    RegID => int,
    Status => string
    ...
}
```

Result:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
RegID	int	ID региона
Status	int	1 – не основной 2 – основной (по-умолчанию)

p_AddProjectRegion – добавление региона Яндекс в проект

boolean p_AddProjectRegion(ProjectID, RegID)

Входные параметры:

Целочисленная переменная *ProjectID*, целочисленная переменная *RegID*

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID проекта
RegID	int	ID добавляемого региона

Возвращает:

Логическая переменная *Result*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
Result	boolean	1 – в случае успешного добавления, 0 – в случае неудачного добавления

p_DefaultProjectRegion – установка региона Яндекс по умолчанию

boolean p_DefaultProjectRegion(ProjectID, RegID)

Входные параметры:

Целочисленная переменная *ProjectID*, целочисленная переменная *RegID*

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID проекта
RegID	int	ID добавляемого региона

Возвращает:

Логическая переменная *Result*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
Result	boolean	1 – в случае успешного добавления, 0 – в случае неудачного добавления

p_RemoveProjectRegion – удаление региона Яндекс из проекта

boolean p_RemoveProjectRegion(ProjectID, RegID)

Входные параметры:

Целочисленная переменная *ProjectID*, целочисленная переменная *RegID*

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
ProjectID	int	ID проекта
RegID	int	ID удаляемого региона

Возвращает:

Логическая переменная *Result*:

<i>Поле</i>	<i>Тип</i>	<i>Описание</i>
Result	boolean	1 – в случае успешного удаления, 0 – в случае неудачного удаления

В качестве примеров работы с API PROMOPark будут рассмотрены фрагменты программ, написанных на языке PHP с использованием библиотеки [nusoap](#).

1. Авторизация:

```
require_once( "nusoap.php" );  
$client = new nusoap_client( "http://promopark.ru/api/Soap/?wsdl", true );  
$result = $client->call( 'p_Login', array( 'Login' => 'TryThis', "Password" => "TryThis" ) );
```

В случае успешной авторизации метод вернет 1.

Далее при каждом следующем запросе необходимо передавать все вернувшиеся cookies. Для этого используется метод `setCookie`:

```
foreach ( $client->getCookies as $cookie )  
    $client->setCookie( $cookie[ "name" ], $cookie[ "value" ] );
```

2. Получение списка активных проектов:

```
$result = $client->call( 'p_ProjectList' );
```

3. Получение отчета по проекту за определенную дату:

```
$result = $client->call( 'p_GetReport', array( 'ProjectID' => 11, "ReportDate" => "2009-03-20" ) );
```

4. Получение списка дат, по которым доступны отчеты по проекту:

```
$result = $client->call( 'p_DatesList', array( 'ProjectID' => 11 ) );
```

5. Получение списка запросов по проекту:

```
$result = $client->call( 'p_QueryList', array( 'ProjectID' => 11 ) );
```

6. Добавление запроса к проекту:

```
$result = $client->call( 'p_AddProjectQuery', array( 'ProjectID' => 11, "Query" => array( "promopark" ) ) );
```

7. Удаление запроса из проекта:

```
$result = $client->call( 'p_RemoveProjectQuery', array( 'ProjectID' => 11, "QueryID" => 1 ) );
```

8. Добавление проекта:

```
$result = $client->call( 'p_AddProject', array( "Domain" => "example.com" ) );
```

9. Удаление проекта:

```
$result = $client->call( 'p_RemoveProject', array( "ProjectID" => 11 ) );
```

10. Установить список поисковых систем для проекта:

```
$result = $client->call( 'p_SetProjectSe', array( 'SE' => array( 'ya' ), 'ProjectID' => 11 ) );
```

11. Получение списка регионов:

```
$result = $client->call( 'p_GetRegionList' );
```

12. Получение списка регионов для проекта:

```
$result = $client->call( 'p_GetProjectRegions', array( 'ProjectID' => 11 ) );
```

13. Добавление региона к проекту:

```
$result = $client->call( 'p_AddProjectRegion', array( 'ProjectID' => 11, "RegID" => 111 ) );
```

14. Удаление региона из проекта:

```
$result = $client->call( 'p_RemoveProjectRegion', array( 'ProjectID' => 11, "RegID" => 111 ) );
```

15. Установка региона по-умолчанию:

```
$result = $client->call( 'p_DefaultProjectRegion', array( 'ProjectID' => 11, "RegID" => 111 ) );
```

Cookies действительны в течение 30 минут с момента авторизации.