RANDOMPICKER
by VeroMotion

RandomPicker.com API documentation

Document version: May 21th, 2023

RandomPicker's SOAP API

RandomPicker's SOAP API uses the SOAP 1.2 (communication protocol) a WSDL 1.1 (web service definition language) standards. All requests must be made over SSL.

How to create a project and conduct drawing

1. Call the function **LoginInsert** to log in**.** The obtained token is used for authentification and authorization of your requests.

2. Create a new project by calling **ProjectInsert**. Use the returned ID to identify which project to use.

3. Upload all entries using **ParticipantInsertProject** (single entry) or **ParticipantInsertBatch** (multiple entries).

4. Conduct the drawing by running **ProjectUpdateDraw.**

5. You can edit your project with **ProjectUpdate.**

6. If you need to edit entries delete them all first with **ParticipantDeleteProject** and upload them again.

7. All changes can be made only before the final drawing. The project will be locked after the final drawing.

User Administration

Provided by the service https://app.randompicker.com/Webservice/User.asmx

We recommend creating a separate API user on your RandomPicker account. If you provide your own username and password for the API (not recommended), all actions through the API will appear as being committed by you. This technique also helps protect your personal authentication parameters, as you can provide a different username and password.

LoginInsert

Signs in the user to the system. Your account must have permission to use API by RandomPicker.com system. Contact our helpdesk to get the permission.

Input – name and password

Output – access token; if the login process is not successful the system returns null value. The token is valid for 30 minutes.

LoginDetail – returns details about the current user

Inputs:

Access token	ID_Login

Outputs:

Validation errors	ValidationMessages
ID User	ID_User
Token validity from	ValidFrom
Token validity to	ValidTo
User name	UserName
User email	Email

Projects

Provided by the service https://app.randompicker.com/Webservice/Project.asmx

ProjectAll – read projects

Inputs:

		Values	**Description**
Acess token*	**ID_Login**		
Project ID	**ID**		All projects are listed if blank.
Project type	**ID_ProjectType**	PrizeWinner GroupsDivided	draw winners sport draws All projects are listed if blank.
Project name	**DisplayName**		All projects are listed if blank.
Type of drawing	**ID_Method**	guid hwThermalNoise	GUID numbers themal noise RNG All projects are listed if blank.
Project state	**ID_ProjectState**	Open Closed Archived Deleted UserDeleted ArchivedFinished ReadyToDraw ReadyToFinalDraw	All projects are listed if blank.
Type of Seal	**ID_Seal**	round squareDark squareLight	All projects are listed if blank.

Outputs – an array of projects filtered as requested:

ID – project ID
ID_ProjectType
DisplayName
ID_Method
Conditions
CreatedDate
ClosedDate
TimeZone
UtcOffset
ID_ProjectState
ProjectStateResource
PublicResults
Key
ID_Seal
WWW
HumanAuditor
IsCompany

ProjectInsert – create a new project

Inputs:

		Values	Description
Acess token*	**ID_Login**		
Project name*	**DisplayName**		
Method*	**ID_Method**	**guid** hwThermalNoise	GUID numbers (quicker) HW generator (real random - slower)
Conditions*	**Conditions**		
Public results*	**PublicResults**	**true** false	
Website	**WWW**		
Project type*	**ID_ProjectType**	**PrizeWinner** GroupsDivided	draw winners sport draws
Type of drawing	**ID_DrawType**	Commercial Nonprofit	paid free, limited
Prizes	arrays of pairs sorted in ascending order (first prize on the first place etc.) **Count** **PrizeName** (You can also add prizes using the method PrizeInsertProject.)		 integer short text
Equal Weights	EqualWeights	true **false**	Weight will be cropped to 1 for all participants (same change to win).

Outputs:

ID of the created project	
Error message	Error
Validation errors	ValidationMessages

ProjectUpdate – edit a project

Inputs: Almost the same as the function *ProjectInsert*. With the *ID* parameter (identifies which project to edit) and without EqualWeights. Equal weights cannot be changed after project is created.

Outputs:

Error message	Error
Validation errors	ValidationMessages

ProjectDetail – get project details

Inputs:

Access token	ID_Login
Project ID	ID

Outputs:

Error message	Error
Validation errors	ValidationMessages
ID project	ID
Project name	DisplayName
Project www	Www
Project key	Key
Created date	CreatedDate
Closed date	ClosedDate
Project conditions	Conditions

ProjectUpdateRecalculateCache – reload project cache

For example participants count, max weight and other cached values

Inputs:

Access token	ID_Login
Project ID	ID_Project

Outputs:

Error message	Error
Validation errors	ValidationMessages

Prizes

PrizeInsertProject – add a new prize item

Inputs:

Access token	ID_Login
Project ID	ID_Project
Prizes array	Items

Outputs:

ID prize	ID
Error message	Error
Validation errors	ValidationMessages

PrizeDeleteProject – delete the prize item from the project

Inputs:

Access token	ID_Login
Project ID	ID_Project
ID prize (use PrizeAllProject to get ID)	ID

Outputs:

Error message	Error
Validation errors	ValidationMessages

PrizeAllProject – delete all prizes from the project

Inputs:

Access token	ID_Login
Project ID	ID_Project
Prize ID (if you wish to user method as detail method)	ID
Prizes order	IsReverseOrder

Outputs:

Error message	Error
Validation errors	ValidationMessages
Project ID number	ID_Project
Prize ID	ID
Prize order	Order
Prize number	Number
Prize number	DisplayName

Entries

ParticipantInsertProject – insert one entry to the project

Use the method only for a single entry. If you have more than one entry, we strongly recommend using the ParticipantInsertBatch (see below) to avoid possible slowdowns.

Inputs:

Access token	ID_Login
Project ID	ID_Project
Public information	PublicInfo
Private information	PrivateInfo
Entry weight	Weight

Outputs:

ID of participant	ID	int
if the weight was reduced	IsCut	true false
Error message	Error	
Validation errors	ValidationMessages	

ParticipantInsertBatch – insert multiple entries

Inputs:

Access token	ID_Login
Project ID	ID_Project
Participants (array of a group of tree): Public information (required) Private information (optional) Entry weight (optional)	 PublicInfo PrivateInfo Weight
Project cache recalculate. Cache will be reloaded (participants count, …). We recommend to use it when your last batch is uploaded. Cache reloading can take a while. Uses ProjectUpdateRecalculateCache api method	IsCacheRecalculate

Outputs:

Number of entries with a reduced weight	CutCount
Error message	Error
Validation errors	ValidationMessages

ParticipantDeleteProject – remove entries in the project

Inputs:

Access token	ID_Login
Project ID	ID_Project
ID Participant (use ParticipantAllProject to get ID)	ID

Outputs:

Error message	Error
Validation errors	ValidationMessages

ParticipantAllProject – read project entries or winners

Inputs:

Access token	ID_Login
Participant ID	ID
Project ID number	ID_Project
Prize ID number	ID_Prize
Search string	Search
Boolean to show only winners or all participants (winners also have ID_Prize not null)	ShowOnlyWinners
Search in private info	PrivateInfo
Search in public info	PublicInfo

Outputs:

Error message	Error
Validation errors	ValidationMessages
Participant ID	ID
Order	RowNumber
Project ID number	ID_Project
Public info	PublicInfo
Internal note	PrivateInfo
Weight	Weight
ID Prize (any value except null means a winner)	ID_Prize
Prize name	Prize
Entry source is a widget	IsWidget
Optional field 1 from a widget	Optional1
Optional field 2 from a widget	Optional2
Optional field 3 from a widget	Optional3
Optional field 4 from a widget	Optional4
Optional field 5 from a widget	Optional5

Start drawing

ProjectUpdateDraw – conduct a draw

Inputs:

Access token	ID_Login	
Project ID	ID_Project	
Final or test drawing	Final	**true** (means final drawing) false (means test drawing)

Output – array of winners:

Entry ID	ID
Public Information	PublicInfo
Private information	PrivateInfo
Entry weight	Weight
Random number	Sort
Prize ID	ID_Prize
Prize name	Prize
Prize order	PrizeOrder
Error message	Error
Validation errors	ValidationMessages

Input Validation

An input validation is executed before each SQL query. Each class has a property ValidationMessages (List<ValidateMessage>):

- *NULL* if the input is correct.
- If there are any validation errors:
  - The service returns only one row (if an array is returned).
  - ValidationMessages contains a list of error messages.
  - Other properties of the output class are set to *NULL*.

ValidateMessage properties

- string **Property** - the name of the output property related to an error
- string **DisplayName** – error description
- string **ID_Error** – programmatically readable ID of the error
- string **Args** – error arguments, e.g. max. number of prize categories etc.

Example – PHP

```php
// login to API
  $client = new SoapClient("https://app.randompicker.com/Webservice/User.asmx?wsdl");

  $input = array(
      "UserName" => 'your username',
      "Password" => 'your password',
  );

  $login_result = $client->LoginInsert(array("loginInsertInput" => $input));
  $token = $login_result->LoginInsertResult->ID;

  $client   = new SoapClient("https://app.randompicker.com/Webservice/Project.asmx?wsdl");

// create a new draw project
  $input = array(
          "ID_Login"         => $token,
          "DisplayName"      => 'New project name',
          "ID_Method"        => 'guid',
          "Conditions"       => 'conditions description',
          "PublicResults"    => true,
          "WWW"              => '',
          "ID_ProjectType"   => 'PrizeWinner',
          "ID_DrawType"      => 'Nonprofit',
          "Prizes"           => array(0 => array('Count' => 1,'PrizeName' => 'First'),1 =>
array('Count' => 1,'PrizeName' => 'Second')),
      );

  $result = $client->ProjectInsert(array("projectInsertInput" => $input));
  $project_id = $result->ProjectInsertResult->ID;

// add one entry
  $input = array(
                  "ID_Login"         => $token,
                  "ID_Project"       => $project_id,
                  "PublicInfo"       => 'user 1 public',
                  "PrivateInfo"      => 'user 1 private',
                  "Weight"           => 1,
              );

  $result = $client->ParticipantInsertProject(array("participantInsertProjectInput" =>
$input));

// add multiple entries
  $input = array(
                  "ID_Login"         => $token,
                  "ID_Project"       => $project_id,
                  "Participants"     => array(0 => array('PublicInfo' => 'user 2 public',
                                                         'PrivateInfo' => 'user 2 private',
                                                         'Weight' => 1),
                                             1 => array('PublicInfo' => 'user 3 public',
                                                         'PrivateInfo' => 'user 3 private',
                                                         'Weight' => 1)),
                                       );

  $result = $client->ParticipantInsertBatch(array("participantInsertBatchInput" => $input));

// conduct final drawing
  $input = array(
                  "ID_Login"         => $token,
                  "ID"               => $project_id,
                  "Final"            => true,
              );

  $result = $client->ProjectUpdateDraw(array("projectUpdateDrawInput" => $input));
```

Example – C#

For using RandomPicker webservices, add following web references in a standard way into your project in Visual Studio:

- https://app.randompicker.com/Webservice/User.asmx
- https://app.randompicker.com/Webservice/Project.asmx

Classes UserService.User and ProjectService.Project are used for calling the webservices. They are generated by Visual Studio when the web references are added.

```
//Generated class for calling the webservice
https://app.randompicker.com/Webservice/User.asmx
UserService.User UserService = new UserService.User();

//login to API
var login_result = UserService.LoginInsert(new UserService.LoginInsertInput()
{
    UserName = "your username",
    Password = "your password"
});
Guid token = login_result.ID.Value;

//Generated class for calling the webservice
https://app.randompicker.com/Webservice/Project.asmx
ProjectService.Project ProjectService = new ProjectService.Project();

//create a new draw project
var ProjectOutput = ProjectService.ProjectInsert(new ProjectService.ProjectInsertInput()
{
    ID_Login = token,
    DisplayName = "New project name",
    ID_Method = "guid",
    Conditions = "conditions description",
    PublicResults = true,
    WWW = "",
    ID_ProjectType = "PrizeWinner",
    ID_DrawType = "Nonprofit",
    Prizes = new ProjectService.Prize[]
    {
        new ProjectService.Prize() { Count = 1, PrizeName = "First" },
        new ProjectService.Prize() { Count = 1, PrizeName = "Second" }
    }
});

int project_id = ProjectOutput.ID.Value;

//add one entry
var ParticipantOutput = ProjectService.ParticipantInsertProject(new
ProjectService.ParticipantInsertProjectInput()
{
    ID_Login = token,
    ID_Project = project_id,
    PublicInfo = "user 1 public",
    PrivateInfo = "user 1 private",
    Weight = 1
});


var DrawOutput = ProjectService.ProjectUpdateDraw(new ProjectService.ProjectUpdateDrawInput()
{
    ID_Login = token,
    ID = project_id,
    Final = true
});
```