InboxGames API

Table of Contents

1 General description

The Inbox.GamesAPI allows third party Game Developers to connect to the Inbox.Games system.

The Developer registers e-mail address on site www.inbox.lt to receive credentials for authentication in management interface at URL http://games.inbox.lt/api/.

The Developer uses the link to register Web Game / Application (further – "the Game") in the Inbox.GamesAPI, by entering the required parameters (primary parameters are: a unique name of the Game, an IP address of a server where the Game is running).

The Game is hosted on the Developer's server.

The Inbox.GamesAPI provides the Developer an API (supported interfaces are XML, JSON and PHP) to interact with the Inbox.Games platform and an Inbox.PaymentAPI to monetize the Game.

XML, JSON (more info at http://www.json.org) and PHP interfaces are available by URLs:
http://api-games.inbox.lt/api/1.0/xml/

http://api-games.inbox.lt/api/1.0/json/

http://api-games.inbox.lt/api/1.0/php/

The Game is displayed in HTML IFRAME.

2 Stages of registration

After successful authorization in the portal www.inbox.lt, the Developer may access a management interface by link http://games.inbox.lt/api/

2.1 Registration of the Developer in the Inbox.Games management interface

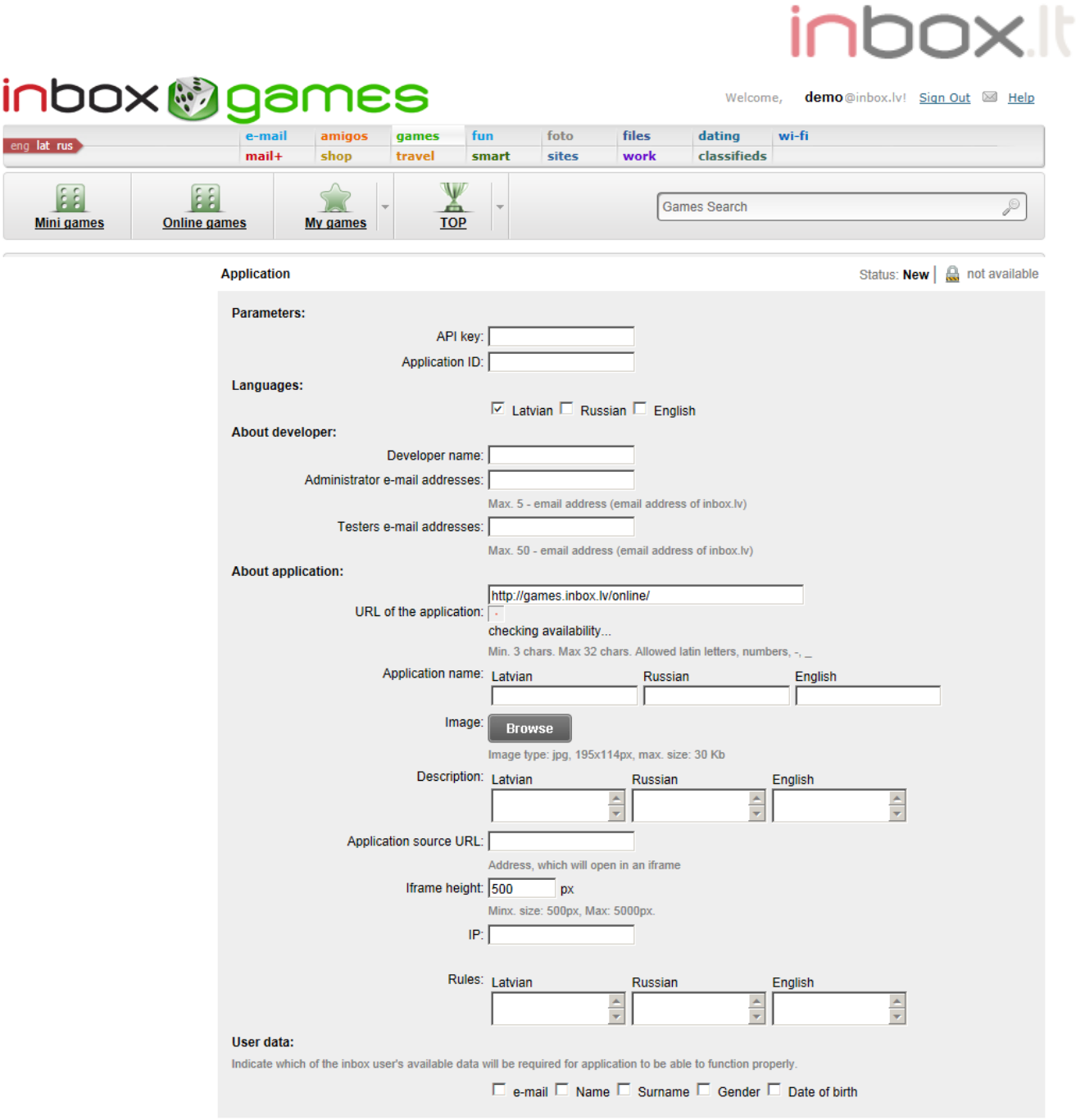
UUID is generated for the Developer on this step.

The functions of API are not available to the Developer on this step.

Authorization from the portal Inbox.lv is required to process the step.

2.2 Registration of the Game

The Developer completes a registration form for the Game and chooses where the Game requires an access to the User's data (access to the appropriate API functions).



Picture Nr.1

At this step the Game UUID (Application ID at picture Nr1) is generated.

2.2.1 Testing the Game

At this step the Game is available to the Developer and testers only. Accounts of testers may be specified on the Game registration step.

The Inbox.GamesAPI for the Game is available at this step.

2.2.2 Publishing the Game

Upon completion of the test, the Developer publishes the Game for further examination of the Portal support team.

2.2.3 Approvement of the Game

The Portal administrators verify and approve the Game.

The Game is published automatically when approved.

The Developer receives a notice of the reasons of refusal in case of rejection.

2.3 Accessing the Game

The User visits a page “On-Line Games” in http://games.inbox.lt, selects the Game and directed to a link of the Game (http://games.inbox.lt/online/<GameURI>).

Connecting to the Game for first time the User must be authorized or will be redirected to the login form. The User is prompted to confirm to access the Game and provide private data to the Developer (as specified at the registration of the Game). The data is sent to the Developer and is available to the Developer through functions of the Inbox.GamesAPI as well (details in section 2.4.3).

2.4 Playing the Game

When confirmation from the User is received (the User agrees to the Terms and Conditions), the User is redirected to the address the Developer specified for the Game in registration process (section 2.2.4). The “On-Line Games” section become loaded and iFrame contains the original URL of the Game, provided by the Developer in registration process of the Game. iFrame code is like following:

```
<iframe id="game_frame" scrolling="no" height="800" frameborder="0" width="980"
border="0" style="border-width: 0pt; overflow: hidden;"
src="http://games.com/integration/inbox/?uid=4e000f41-66fc-41cf-aae1-
1259aac51003&language=en"></iframe>
```

The parameters of iFrame - *width*, *height* and *src* are taken from the profile of the Game. Additional parameters are passed to iFrame URL – a *uid* and *an additional language*.

The *uid* parameter is the UUID of the User in the Game, and *language* is the language of the interface of games.inbox.lt from which the User comes to the Game.

2.4.1 Checking an access to API functions

After completing the Game registration process it is possible to request a list of API functions available. The Inbox.GamesAPI reports when required rights are available and returns a list of unavailable functions otherwise.

Sample:
JSON (request at http://api-games.inbox.lt/1.0/json/)

Request:

```
{
 "action": "check_permissions",
 "app": "4d907c4a-ec48-43a7-913d-52e3f22a4376",
 "apiKey": "4d9dd03a-fc98-4590-8d98-2730cfb25ae3",
```

```
 "data": "fname, lname"
 }
```

Answer:

```
{
"code":"200 OK"
}
```

Where *app* is UUID of the Game and *apiKey* is UUID of the User (*apiKey* UUID of the Developer)

2.4.2 Accessing User's data that requires authorization

The Developer may check whether the Game has a right to obtain specific private data of the User.

Sample:
JSON (request at http://api-games.inbox.lt/1.0/json/)

Request:

```
{
"action": "authorize",
"app": "4d907c4a-ec48-43a7-913d-52e3f22a4376",
"apiKey": "4d9dd349-c270-4d90-9339-2731e53a1a5d",
"data": "fname,lname"
}
```

Answer:

```
{
"code":"200 OK",
"apiKey":"4d9dd349-c270-4d90-9339-2731e53a1a5d",
"language":"en",
"inviter":"",
"inviteExtra":""
}
```

Where *app* is UUID of the Game, *apiKey* is UUID of the User, and *data* is specific private data.

2.4.3 Obtaining User's data

The Developer may collect data of player while the Game is allowed to access the functions of API (section 2.4.1) and the User's data (section 2.4.2),

Sample:
JSON (request at http://api-games.inbox.lt/1.0/json/)

Request:

```
{
"action": "userdata",
 "app": "4d907c4a-ec48-43a7-913d-52e3f22a4376",
```

```
"apiKey": "4d9dd349-c270-4d90-9339-2731e53a1a5d",
 "data": "mail,fname,lname"
}
```

Answer:

```
{
"code":"200 OK",
"users":[
{
"apiKey":"4d9dd349-c270-4d90-9339-2731e53a1a5d",
"mail":"31gdDEQbBZu8LN6ykA8NavyCQ==@openid.inbox.lv",
"fname":"sss",
"lname":"dddd"
}]
}
```

Where *app* is UUID of the Game, *apiKey* is UUID of the User, and *data* is specific data requested (possible parameters *fname*, *lname*, and *email address*).

In case of request for email address, a real e-mail address of the User is not presented. Instead Game-depended alias is returned in form like `31gdDEQbBZu8LN6ykA8NavyCQ==@openid.inbox.lt` .

This address is generated when the User confirms to transfer e-mail address to the Game. The address is active from the moment of confirmation till the end of subscription to the Game if the Developer is allowed to access it.

Example of PHP code to demonstrate the basic functions of the Inbox.GamesAPI

```
<?
error_reporting(E_ALL);
define('API_ENDPOINT', 'http://api-games.inbox.lt/1.0/json/');
define('API_ROOT', 'http:// api-games.inbox.lt/1.0/');
define('SELF_URI', 'http:// api-games.inbox.lt/1.0/demo1/');


//This is a DEMO application, it allows to run a series of tests on gaming API and
hopefuly explains how to use it.
//As it is extremely simplified, there's no program structure to speak of. Please
don't write your real apps like this.

function unchunkHttpResponse($str = null)
{
    if (!is_string($str) or strlen($str) < 1)
        return false;
    $eol = "\r\n";
    $add = strlen($eol);
    $tmp = $str;
    $str = '';
    do
    {
        $tmp = ltrim($tmp);
        $pos = strpos($tmp, $eol);
        if ($pos === false)
            return false;
        $len = hexdec(substr($tmp, 0, $pos));
        if (!is_numeric($len) or $len < 0)
            return false;
        $str .= substr($tmp, ($pos + $add), $len);
        $tmp  = substr($tmp, ($len + $pos + $add));
        $check = trim($tmp);
    }
    while(!empty($check));
    unset($tmp);
    return $str;
}

function doPostRequest($uri, $data)
{
    $eol = "\r\n";
    $uri = substr($uri, 7);
    $host = substr($uri, 0, strpos($uri, '/'));
    $uri = substr($uri, strpos($uri, '/'));
    $f = fsockopen($host, 80);
    $req = "POST " . $uri . " HTTP/1.1" . $eol;
    $req .= "Host: " . $host . $eol;
    $req .= "Content-Type: text/plain" . $eol;
    $req .= "Content-Length: " . strlen($data) . $eol;
    $req .= "Connection: close" . $eol . $eol;
    fwrite($f, $req);
    fwrite($f, $data);
    $str = '';
```

```
    if ($f)
    {
        while ($r = fread($f, 1024))
            $str .= $r;
        fclose($f);
        $tmp = explode($eol . $eol, $str, 2);
        $headers = $tmp[0];
        $str = $tmp[1];

        $headerArr = explode($eol, $headers);
        $chunked = false;
        foreach ($headerArr as $h)
        {
            if ($h == 'Transfer-Encoding: chunked')
                $chunked = true;
            $tmp = explode(':', $h, 2);
            $hName = trim($tmp[0]);
            //$hVal = trim(@$tmp[1]);
            if (preg_match('`HTTP/1\.. (\d*) (.*)`', $hName, $matches))
            {
                if ($matches[1] != 200)
                    return $matches[1] . ' ' . $hName;
            }
        }
    }
    else
        return null;

    if ($chunked)
        return unchunkHttpResponse($str);
    else
       return $str;
}

function checkperm()
{
    //be careful not to mix up te quotes
    $source = '
{
  "action": "check_permissions",
  "app": "0f1dcf5c-d3b5-3884-25c6-cc54b28714e7",
  "apiKey": "f8a2c9a3-5c94-5d24-cd6b-37bc9a394ff0",
  "data": "fname,lname"
}';
    $data = doPostRequest(API_ENDPOINT, $source);
    return $data;
}

function authorize()
{
    //be careful not to mix up te quotes
    $source = '
{
  "action": "authorize",
  "app": "0f1dcf5c-d3b5-3884-25c6-cc54b28714e7",
  "apiKey": "f8a2c9a3-5c94-5d24-cd6b-37bc9a394ff0",
  "data": "fname,lname"
}';
```

```
    $data = doPostRequest(API_ENDPOINT, $source);
    return $data;
}

function getfields()
{
    $source = '
{
  "action": "userdata",
  "app": "0f1dcf5c-d3b5-3884-25c6-cc54b28714e7",
  "apiKey": "f8a2c9a3-5c94-5d24-cd6b-37bc9a394ff0",
  "data": "fname,lname,email"
}';
    $data = doPostRequest(API_ENDPOINT, $source);
    return $data;
}

$out = '<h1>Awesome app</h1>';
$out .= '<h2>Test me if you will</h2>';

if (!isset($_GET['action']))
    $_GET['action'] = '';

switch ($_GET['action'])
{
    case 'checkperms':
    case 'getfields':
    case 'getfields2':
    case 'getusercount':
    case 'getstats':
    case 'getuserpayments':
    case 'viewtransaction':
    case 'createtransaction':
    case 'createandprocesstransaction':
        $out .= $_GET['action']();
        break;

}

$out .= '<br /><br /><a href="?action=checkperm">check user permission to access
private data (name and surname)</a><br />';
$out .= '<a href="?action=getfields">get user private data (name and
surname)</a><br />';

$out .= '<a href="?action=authorize"> get user authorization for name </a><br />';

?>
<?=$out;?>
```