Compiled Single Player Games
Installation Guide

Version 1.1.0

Table of Contents

1. Introduction

This document describes the way to install the compiled single player games (Flash ActionScript 3.0 SWF, Flash ActionScript 2.0 SWF, Java JAR, HTML5) into your web server.

2. Directory Structure

The files are arranged in the following directory structure:

```
+
+-- instructions.pdf
+-- LICENSE.TXT
+-- games
+-- icons
+-- versions
+-- PHPMySQL  (if using PHP + MySQL High Scores)
+-- ASP.NETAccess  (if using ASP.NET + Access High Scores)
```

The following are the descriptions of each of the files / folders:

- `instructions.pdf`
    - This file
- `LICENSE.TXT`
    - The license agreement
- `games`
    - The folder containing the game files and sample HTML pages
- `icons`
    - The folder containing the icons and screenshots of each of the games, you can use these files in your web page or other areas
- `versions`
    - The folder containing the release notes of each of the games, this is for your reference
- `PHPMySQL`
    - This folder exists only if you are using PHP + MySQL high scores table. This folder contains the server side PHP files and SQL schema for you to upload to your web server and set up the database
- `ASP.NETAccess`
    - This folder exists only if you are using ASP.NET + Access high scores table. This folder contains the server side ASPX files and MDB database file for you to upload to your web server

3. Step-by-Step Example

This section walks through a step-by-step example of installing the games in your web server.

1. Upload the files and folders in the `games` folder to a directory in your web server
2. If you are not using a high scores table, then the process is complete. You can now open a web browser and browse to the HTML files in the directory to play the game. The HTML files are meant to be sample files only, you can modify them to suit your needs.
3. If you are using a high scores table, then there are more steps needed to set up the database and upload the server side scripts to your server. The remaining of this document are relevant only if you are using a high scores table.

For PHP and MySQL

1. Run the SQL statements in `PHPMySQL/table.sql` to create a table in your database to store the high scores.

2. Edit `$databaseHost`, `$databaseUser`, `$databasePassword` and `$databaseName` in `config.php` so that they contain the actual database host name, database user name, database password and database name respectively.

3. Upload all the files except `table.sql` under the `PHPMySQL` directory to your web server. The files should be put in the same directory you have put the games to. That is, the `games` directory and the `PHPMySQL` directory merge into one single directory in your web server. If you would like the server side scripts to be in a different directory, please refer to Section 9.

4. Modify the `pictures` directory so that the web server has read-and-write access to it.

5. Navigate to `test.php` in your web browser. This script will run a few tests to see if the settings are correct. If errors occur, please check the settings again.

6. That's all of the setup. Now navigate to the game HTM files in your web browser and test the games. The high scores of the games should now be able

to be saved and loaded.

For ASP.NET and Access

1. Upload all the files under the `ASP.NETAccess` directory to your web server. The files should be put in the same directory you have put the games to. That is, the `games` directory and the `ASP.NETAccess` directory merge into one single directory in your web server. If you would like the server side scripts to be in a different directory, please refer to Section 9.

2. Modify the `pictures` directory so that the web server has read-and-write access to it.

3. Navigate to test.aspx in your web browser, this script will run a few tests to see if the settings are correct. If errors occur, please check the settings again.

4. That's all of the setup. Now navigate to the game HTM files in your web browser and test the games. The high scores of the games should now be able to be saved and loaded.

4. Parameters on the Server Side (PHP)

This section explains the parameters that can be tuned on the server side. These parameters are put in the config.php file.

Parameter	**Description**
`$databaseHost`	The host of the database.
`$databaseUser`	The username of the database.
`$databasePassword`	The password of the database.
`$databaseName`	The name of the database.
`$databaseTableName`	The name of the table in the database containing the scores.
`$picturesPath`	The path to the folder where the pictures will be stored.
`$noOfScores`	The maximum number of high scores that the loadScores.php script will send to the game.
`$scoreRanges`	An array of date ranges of high scores reported to the game. The possible values are: “eternity” - the highest scores of all times are reported; “pastDay” - the highest scores within the past day are reported; “pastWeek” - the highest scores within the past week are reported; “pastMonth” - the highest scores within the past month are reported; “pastYear” - the highest scores within the past year are reported; “thisDay” - the highest scores for today are reported; “thisWeek” - the highest scores for this week are reported; “thisMonth” - the highest scores for this month are reported; “thisYear” - the highest scores for this year are reported.
`$duplicateNameTreatment`	The action to take if duplicate names are entered into the High Scores table. The possible values are: “showAll” - multiple records from the same name will be shown; “showHighest” - only the highest score from the same name will be shown; “showLatest” - only the latest record from the same name will be shown. “showFirst” – only the first record from the same name will be shown. “accumulate” – scores from the same name will be added up and accumulated

`$hashKey`	The hash key used. Do not change it if the hash key is already set.
`$hashKeyRequired`	Whether a check for the hash to authenticate the submitted score is required.
`$removeOldScores`	Whether outdated scores will be removed from the database.
`$shareText`	The text used when sharing.
`$shareGooglePlusAppID`	The Google+ app ID. This is required for Google+ sharing.
`$shareGooglePlusText`	The text to show on the Google+ sharing page
`$shareGooglePlusButtonText`	The text on the share button on the Google+ sharing page
`$shareGooglePlusCloseText`	The text on the close button on the Google+ sharing page
`$shareFacebookAppID`	The Facebook app ID. This is required only if login through Facebook is allowed. If a valid Facebook app ID exists, the player can post his scores on Facebook and send challenges to friends.
`$shareFacebookAppSecret`	The Facebook app secret. This is required if login through Facebook is allowed.
`$shareFacebookActionText`	The action text to show. Used only if `$shareFacebookAppID` is non-empty.
`$shareFacebookEnableChallenge`	Whether challenges are enabled. Used only if `$shareFacebookAppID` is non-empty. If enabled, you need to modify the canvas page of the app so that challenges can be dealt with correctly (see Section 7 for details).
`$shareFacebookThanksText`	The text to show on the Challenge page. Used only if `$shareFacebookAppID` is non-empty.
`$shareFacebookChallengeText`	The text shown on the “Challenge” button on the Challenge page. Used only if `$shareFacebookAppID` is non-empty.
`$shareFacebookCloseText`	The text shown on the “Close” button on the Challenge page. Used only if `$shareFacebookAppID` is non-empty.
`$shareFacebookRequestTitle`	The title on the “Send Request” page on Facebook. Used only if `$shareFacebookAppID` is non-empty.
`$shareFacebookRequestText`	The text content of the Facebook request. Used only if `$shareFacebookAppID` is non-empty.
`$emailSender`	The email address of the sender of the emails. This must be set to a valid email address in your domain (e.g. your own email address).
`$emailSubject`	The subject of the challenge email.
`$emailContent`	The content of the challenge email.

`$emailDefaultURL`	The default URL of the game, in case the game could not properly send its URL to the script.
`$namesFilterFile`	The file containing the obscene words.
`$namesFilterApplyToLogin`	Whether the obscene names filter is applied to login username.
`$namesFilterApplyToFacebook`	Whether the obscene names filter is applied to Facebook username.

5. Parameters on the Server Side (ASP.NET C#)

This section explains the parameters that can be tuned on the server side. These parameters are put in the config.php file.

Parameter	**Description**
`databaseFile`	The Access database file.
`databaseTableName`	The name of the table in the database containing the scores.
`picturesPath`	The path to the folder where the pictures will be stored.
`noOfScores`	The maximum number of high scores the loadScores.aspx script will send to the game.
`scoreRanges`	An array of date ranges of high scores reported to the game. The possible values are: “eternity” - the highest scores of all times are reported; “pastDay” - the highest scores within the past day are reported; “pastWeek” - the highest scores within the past week are reported; “pastMonth” - the highest scores within the past month are reported; “pastYear” - the highest scores within the past year are reported; “thisDay” - the highest scores for today are reported; “thisWeek” - the highest scores for this week are reported; “thisMonth” - the highest scores for this month are reported; “thisYear” - the highest scores for this year are reported.
`duplicateNameTreatment`	The action to take if duplicate names are entered into the high scores table. The possible values are: “showAll” - multiple records from the same name will be shown; “showHighest” - only the highest score from the same name will be shown; “showLatest” - only the latest record from the same name will be shown. “showFirst” – only the first record from the same name will be shown. “accumulate” – scores from the same name will be added up and accumulated
`hashKey`	The hash key used. Do not change it if the hash key is already set.
`hashKeyRequired`	Whether a check for the hash to authenticate the

	submitted score is required for all games.
`removeOldScores`	Whether outdated scores will be removed from the database.
`shareText`	The text used when sharing.
`shareGooglePlusAppID`	The Google+ app ID. This is required for Google+ sharing.
`shareGooglePlusText`	The text to show on the Google+ sharing page
`shareGooglePlusButtonText`	The text on the share button on the Google+ sharing page
`shareGooglePlusCloseText`	The text on the close button on the Google+ sharing page
`shareFacebookAppID`	The Facebook app ID. This is required only if login through Facebook is allowed. If a valid Facebook app ID exists, the player can post scores on Facebook and can send challenges to friends.
`shareFacebookAppSecret`	The Facebook app secret. This is required if login through Facebook is allowed.
`shareFacebookActionText`	The action text to show. Used only if `shareFacebookAppID` is non-empty.
`shareFacebookEnableChallenge`	Whether challenges are enabled. Used only if `shareFacebookAppID` is non-empty. If enabled, you need to modify the canvas page of the app so that challenges can be dealt with correctly (see Section 7 for details).
`shareFacebookThanksText`	The text to show on the Challenge page. Used only if `shareFacebookAppID` is non-empty.
`shareFacebookChallengeText`	The text shown on the “Challenge” button on the Challenge page. Used only if `shareFacebookAppID` is non-empty.
`shareFacebookCloseText`	The text shown on the “Close” button on the Challenge page. Used only if `shareFacebookAppID` is non-empty.
`shareFacebookRequestTitle`	The title on the “Send Request” page on Facebook. Used only if `shareFacebookAppID` is non-empty.
`shareFacebookRequestText`	The text content of the Facebook request. Used only if `shareFacebookAppID` is non-empty.
`emailSender`	The email address of the sender of the emails. This must be set to a valid email address in your domain (e.g. your own email address).
`emailSubject`	The subject of the challenge email.
`emailContent`	The content of the challenge email.
`emailDefaultURL`	The default URL of the game, in case the game could not properly send its URL to the script.
`namesFilterFile`	The file containing the obscene words.

`namesFilterApplyToLogin`	Whether the obscene names filter is applied to login user name.
`namesFilterApplyToFacebook`	Whether the obscene names filter is applied to Facebook user name.

6. Obscene Names Filter

An obscene names filter is installed so that obscene names will not be input to the database. You can edit `obscene.txt` to add or remove obscene words.

If an obscene word (ignoring spaces) appears in a part of the name entered by the user, the name will be rejected.

7. Facebook Sharing

When the player finished playing the game, he will have the option to share the game on Facebook. There are several ways to configure the High Scores Module for the players to share the game.

Basic Sharing: the `shareFacebookAppID` is set to an empty string so that only the URL is shared and the player's results will not be included in the sharing message (unless the player writes it himself).

Extended Sharing: the `shareFacebookAppID` is set to a valid Facebook app ID, and `facebookEnableChallenge` is set to false. In this case, the player's result will be posted in the sharing message, but after sharing the player will not have the option to challenge his friends.

Advanced Sharing: the `shareFacebookAppID` is set to a valid Facebook app ID, and `facebookEnableChallenge` is set to true. After sharing the game, the player can choose to challenge his friends. The challenges are sent by Facebook requests, the URL to share is put in the `data` field of the request object, and the `data` field will be a JSON string, with the `shareURL` field storing the URL to share. Please consult the documentation on Facebook for the way to handle the requests. If you are using PHP, the code to handle the request may look like this:

```
<?php

$FACBOOKAPPID = '';
$FACBOOKAPPSECRET = '';



$request_ids = $_GET['request_ids'];

$facebook = new Facebook(
 array(
      'appId'=>$FACBOOKAPPID,
      'secret'=>$FACBOOKAPPSECRET
 )
);
```

```
$user_id = $facebook->getUser();

$request_idsArray = explode(',', $request_ids);

for($i=0;$i<count($request_idsArray);$i++) {
 $result = $facebook->api("/$request_idsArray[$i]");
 if(!$result) continue;

 $data = json_decode($result['data']);

 $shareURL = $data->shareURL;

 $facebook->api("/$request_idsArray[$i]" . "_" . $user_id,
'DELETE');
}

echo '<script type="text/Javascript">top.location.href = "' .
$shareURL . '";</script>';

?>
```

8. Parameters to Pass to the Game

This section explains the parameters that can be passed to the game. These parameters should be passed from the HTML file to the game. Depending on the format of the games you licensed, the way to pass is slightly different:

Flash ActionScript 3.0 SWF or Flash ActionScript 2.0 SWF

In this case you pass the variables by FlashVars, e.g.

```
<object id="novelgames_flashGame" classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,0,0" width="600" height="400">
 <param name="movie" value="lightning_e.swf" />
 <param name="allowScriptAccess" value="always" />
 <param name="allowFullScreen" value="true" />
 <param name="loop" value="false" />
 <param name="menu" value="false" />
 <param name="quality" value="high" />
 <param name="wmode" value="window" />
 <param name="FlashVars" value="playerName=John+Smith" />
 <embed name="novelgames_flashGame" src="lightning_e.swf"
      width="600" height="400"
       allowScriptAccess="always"
       allowFullScreen="true"
       loop="false" menu="false" quality="high"  wmode="window"
       FlashVars="playerName=John+Smith"
       type="application/x-shockwave-flash"

 pluginspage="http://www.macromedia.com/go/getflashplayer">
 </embed>
</object>
```

Java JAR

In this case you pass the variables through applet parameters, e.g.

```
<script src="http://www.java.com/js/deployJava.js"></script>
<script>
var attributes = { code:'com.novelgames.spgames.lightning.Main',
width:600, height:400};
 var parameters = { jnlp_href: 'lightning_applet.jnlp',
playerName: 'John Smith'};
 deployJava.runApplet(attributes, parameters, '1.7');
</script>
```

HTML5

In this case you pass the variables through JavaScript, e.g.

```
<script>
nogic.initialize(
 document.getElementById('mainCanvas'),
 { playerName: 'John Smith' }
 );
</script>
```

Parameter	**Description**
`siteID`	If this parameter is used, its value will override the site ID set in the SWF file.
`gameID`	If this parameter is used, its value will override the game ID set in the SWF file.
`gameName`	If this parameter is used, its value will override the game name set in the SWF file.
`playerName`	This parameter controls the name of the player. If this parameter is not present, the player will need to enter his / her name before submitting the score to the High Scores table. If this parameter is present, the score will be submitted to the High Scores table using this name.
`playerPictureURL`	This parameter controls the URL of the player's picture. If this parameter is present, a picture of the player will be displayed in the High Scores table.
`playerFacebookUserID`	This parameter sets the Facebook user ID of the playing player. If this parameter is set, you should also set the `playerFacebookUserName`, `playerFacebookUserPicture`, `playerFacebookFriendIDs` parameters.

`playerFacebookUserName`	This parameter sets the Facebook username of the playing player. If this parameter is set, you should also set the `playerFacebookUserID`, `playerFacebookUserPicture`, `playerFacebookFriendIDs` parameters.
`playerFacebookUserPicture`	This parameter sets the Facebook profile picture of the playing player. If this parameter is set, you should also set the `playerFacebookUserID`, `playerFacebookUserName`, `playerFacebookFriendIDs` parameters.
`playerFacebookFriendIDs`	This parameter sets the Facebook user IDs of the playing player's friends. The value should be a comma delimited list of IDs. If this parameter is set, you should also set the `playerFacebookUserID`, `playerFacebookUserName`, `playerFacebookUserPicture` parameters.

9. Troubleshooting

Q) The High Scores Module is not working.

A) Please check

1. Whether you can run the `test.php` (or `test.aspx`) file without errors;
2. Whether all the files, PHP, JS, HTM, SWF, JAR, etc are under the same directory.

Q) We want to organize the scripts so that the server side scripts and the game files are in different folders. Is this possible?

A) Yes, you can add a “base” parameter to the game through the HTML page. The following examples show the case when the game is accessed by http://www.yoursite.com/games/game.htm and the server side scripts are put in http://www.yoursite.com/php/ folder

Flash ActionScript 3.0 SWF and Flash ActionScript 2.0 SWF

```
<object ......>
 ...
 <param name="base" value="http://www.yoursite.com/php/" />
 <embed ...
      base="http://www.yoursite.com/php/">
 </embed>
</object>
```

Java JAR

```
<script src="http://www.java.com/js/deployJava.js"></script>
<script>
var attributes = { code:'com.novelgames.spgames.lightning.Main',
width:600, height:400};
 var parameters = { jnlp_href: 'lightning_applet.jnlp', base: '
http://www.yoursite.com/php/'};
 deployJava.runApplet(attributes, parameters, '1.7');
</script>
```

HTML5

```
<script>
nogic.initialize(
 document.getElementById('mainCanvas'),
 { base: ' http://www.yoursite.com/php/' }
 );
</script>
```

Questions?

Please contact us at
support@novelgames.com

10. Document Modification History

Version	*Date*	*Description*
1.0.0	*2014-07-08*	*First Draft*
1.1.0	*2014-08-20*	*Updated HTML5 codes*