Novel Games

Compiled Multi Player Games
Installation Guide

Version 1.5.1

1. Introduction

This document describes the way to install the compiled multi player games (HTML5, Java JAR) into your web server.

2. Directory Structure

The files are arranged in the following directory structure:

```
+
+-- instructions.pdf
+-- LICENSE.TXT
+-- games
+-- icons
+-- versions
+-- PHP
+-- WebSocket
```

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
- `PHP`
  - This folder contains the server side PHP files and for you to upload to your web server
- `WebSocket`
  - This folder contains the WebSocket server files

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. Upload everything under the `PHP` to the same directory in your web server. Note that the files in the `games` directory and the files in the `PHP` directory are in the same directory in your web server. If you would like the server side scripts to be in a different directory, please refer to Section 10.

3. Make sure that the web server can write to the `lobbyFiles` and the `pictures` directories. For example if you uploaded the files using FTP, you may need to set the permission of that folder so that everyone can write.

4. 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.

5. The setup is now done. Now navigate to game HTM files in your web browser and test the games. You can also open a few browser windows to try the multiplayer gaming.

6. You can navigate to `restart.php` to restart the server; by default the username and password are both **admin**. You can modify `config.php` to change these.

7. Different players of a game will now communicate with each other by making HTTP calls to the PHP scripts. A better way to communicate is through the WebSocket server. The next section covers the setup of the WebSocket server.

4. WebSocket Server

This section walks through a step-by-step example setting up the WebSocket server.

1. Upload the files and folders in the `WebSocket` folder to a directory in your server. It is better not to put these under a publicly accessible location.

2. Open a terminal to your server and run the following command:

   ```
   java -Dconfig=pathToConfigDotTxt -jar pathToLobbyWebSocketServerDotJar > /dev/null 2>&1 &
   ```

   if you have put the `WebSocket` folder at `/home/WebSocket`, then `pathToConfigDotTxt` will be `/home/WebSocket/config.txt` and `pathToLobbyWebSocketServerDotJar` will be `/home/WebSocket/lobbyWebSocketServer.jar`

3. Now the WebSocket server is running and is listening to port 9123 (as set in `config.txt`). The next step is to configure the games to connect to this port.

4. Update `config.php` and set the default public URL of the first room to the URL of the WebSocket server:

   ```
   $defaultRoomPublicURLs[0] = 'ws://hostname:9123/';
   ```

   here `hostname` is the publicly accessible hostname of the server

5. Update `config.php` and set the default private URL of the first room to the URL of the WebSocket server that the web server can connect to. If your web server is the same as the server running the WebSocket server, then you should point to localhost:

   ```
   $defaultRoomPrivateURLs[0] = 'ws://localhost:9123/';
   ```

6. Make sure that no firewall is blocking your connection to port 9123, you can now navigate to the game HTM file to test the game.

7. During deployment to your live servers, you will not want to use port 9123 as the public connection port because it is likely to be blocked by firewalls. You will want to use either port 80 or port 443. If you have a dedicated server to run the WebSocket server, then it is fine. Otherwise if your WebSocket server and your web server are the same server, or if you want to use secure WebSocket (wss), then you need to configure your web server to act as a WebSocket reverse

proxy. In this scenario, the WebSocket server will still listen to port 9123 (or any port other than 80 or 443), and you will set up your reverse proxy to connect to localhost at port 9123. And the setting in `config.php` will be something like this:

```
$defaultRoomPublicURLs[0] = 'wss://hostname/path/';
$defaultRoomPrivateURLs[0] = 'ws://localhost:9123/';
```

Instructions about setting up reverse proxy in your web server are out of scope of this document. Please consult the documentation of your web server.

5. Parameters on the Server Side

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

Parameter	Description
`$defaultRoomNames`	The default names of the game rooms (see also `$gameRoomNames`).
`$defaultRoomCapacities`	The default capacities of the game rooms (see also `$gameRoomCapacities`).
`$defaultRoomEnableds`	The default enabled information about the game rooms (see also `$gameRoomEnableds`).
`$defaultRoomPublicURLs`	The default public URL of the game rooms (see also `$gameRoomPublicURLs`).
`$defaultRoomPrivateURLs`	The default private URL of the game rooms (see also `$gameRoomPrivateURLs`).
`$gameRoomNames`	The names of the game rooms for each particular game. For example, if `$defaultRoomNames` is `array('Gold', 'Silver')`, and if `$gameRoomNames['rockpaperscissors']` is `array('Pro', 'Amateur')`, the game rooms displayed to the player playing the game with gameID `'rockpaperscissors'` will be “Pro” and “Amateur”, while the game rooms displayed to the players playing other games will be “Gold” and “Silver”.
`$gameRoomCapacities`	The capacities of the game rooms for each particular game. Use -1 for unlimited capacity. For example if `$defaultRoomCapacities` is `array(10, 20)`, and if `$gameRoomCapacities['rockpaperscissors']` is `array(5, 10)`, the game rooms displayed to the player playing the game with gameID `'rockpaperscissors'` will have capacities 5 and 10, while the game rooms displayed to the players playing other games will have capacities 10 and 20.
`$gameRoomEnableds`	The enabled information of the game rooms for each particular game. For example if `$defaultRoomEnableds` is `array(false, true)`, and if `$gameRoomCapacities['rockpaperscissors']` is `array(true, false)`, the game rooms displayed to the player playing the game with gameID `'rockpaperscissors'` will be enabled and disabled respectively, while the game rooms

	displayed to the players playing other games will be disabled and enabled respectively.
`$gameRoomPublicURLs`	The public URLs of the game rooms for each particular game. The URL should either be `null` (in this case the game will connect to the PHP files in the same server), an HTTP or HTTPS URL to the folder containing the PHP files (e.g. `'https://hostname/path/'`), or a WS or WSS URL (e.g. `'wss://hostname/path/'`, please refer to the previous section about WebSocket server).
`$gameRoomPrivateURLs`	The private URLs of the game rooms for each particular game. The URLs should be accessible from this server.
`$controlUsername`	The username for the restart.php script.
`$controlPassword`	The password for the restart.php script.
`$duplicateNameTreatment`	The action to take when a player with the same name as an existing player in the Lobby enters the Lobby. The possible values are: “addIndex” - an index will be appended to the end of the name; “showAll” - all the players will be shown with the same name; “disconnectOld” - the existing player with the same name will be disconnected; “disconnectNew” - the new player will not be able to join the room.
`$hashKey`	The hash key used. Do not change it if the hash key is already set.
`$extraKeyLength`	The length of the extra key generated for each player.
`$filesPath`	Used only if $databaseType is set to “files”. This parameter controls the location of the files used to store the data.
`$picturesPath`	The path to the folder where the pictures will be stored.
`$inactiveTimeout`	When a player is inactive for the specified number of seconds, the player will be considered as disconnected.
`$shareWinText`	The text used for sharing if the player wins the game.
`$shareDrawText`	The text used for sharing if the player draws in the game.
`$shareRankText`	The text used for sharing if the player loses the game.
`$shareFacebookAppID`	The Facebook app ID. This is required only if login through Facebook is allowed.
`$shareFacebookAppSecret`	The Facebook app secret. This is required if login through Facebook is allowed.

`$twitterConsumerKey`	The twitter consumer key. This is required if login through twitter is allowed.
`$twitterConsumerSecret`	The twitter consumer secret. This is required if login through twitter is allowed.
`$googleAPIClientID`	The Google API client ID. This is required if login through Google is allowed.
`$googleAPIClientSecret`	The Google API client secret. This is required if login through Google is allowed.

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) is a part of the name entered by the user, the name will be rejected.

Alternatively you can implement `checkNames` function in config.php to implement your own algorithm to filter out obscene words.

7. Recording Results

When the game has ended, you can choose to record the results of the game. To do this, implement the `recordResult` function in `config.php`

Method	`function recordResult($siteID, $gameID, $roomID, $tableUID, $tableID, $playIndex, $playerNames, $playerEmails, $playerIDs, $playerFacebookUserIDs, $playerTwitterUserIDs, $playerGoogleUserIDs, $ranks, $playerIndex, $customExtra)`
Description	This function will be called from each of the players when the game has ended.
Parameters	`siteID` - the site ID `gameID` - the game ID `roomID` - the room ID `tableUID` - the unique ID of the table. The combination of siteID, gameID, roomID and tableUID is guaranteed to be unique `tableID` - the table ID. This is relevant only if the tableID is passed through FlashVars `playIndex` - the index of the play, or the number of replays at this table. If it is the first play at this table it will be zero `playerNames` - an array of names of the playing players `playerEmails` - an array of emails of the playing players. This is relevant only when `enterType` is set to nameEmail `playerIDs` - an array of player IDs of the playing players. This is relevant only when `enterType` is set to login `playerFacebookUserIDs` - an array of Facebook user IDs of the playing players `playerTwitterUserIDs` - an array of Twitter user IDs of the playing players `playerGoogleUserIDs` - an array of Google user IDs of the playing players `ranks` - the ranks of the players. The ranks begin at zero. If for example the ranks is an array of [0, 2, 0, 3], the first player and the third player have both won the game, while the second player is the second runner-up, and the fourth player is the third runner-up `playerIndex` - the index (beginning at zero) of the player sending this result `customExtra` - the string passed to the customExtra parameter when gameEnded is called

If you are using the WebSocket server, then you need to set the `recordResultsCommandArguments` parameter in `config.txt`. The value of this parameter should be an array of command and arguments in JSON format. E.g.

```
recordResultsCommandArugments = [ 'php a.php', '-f' ]
```

When the game has ended, the command will be called with the results passed in through standard input in JSON format.

8. Getting Runtime Information

You can call some scripts to get the runtime information of the server, for example, you can use the information returned to show the number of playing players of each game in your web page.

Script	`getSiteInfo.php`
POST parameters	`siteID` - the siteID of the site you want to query
Description	Call this script to get an XML text showing the information about all the sites queried.

9. 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:

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 set, it will override the site ID set in the SWF file.
gameID	If this parameter is set, it will override the game ID set in the SWF file.
gameName	If this parameter is set, it 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 entering the Lobby. If this parameter is present, the player will enter the Lobby using this name.
`playerPictureURL`	This parameter controls the URL of the player's picture. If this parameter is present, the player's avatar in the Lobby will be loaded from this URL.
`playerFacebookUserID`	This parameter sets the Facebook user ID of the playing player.
`playerTwitterUserID`	This parameter sets the Twitter user ID of the playing player.
`playerGoogleUserID`	This parameter sets the Google user ID of the playing player.

`tablePlayerNames0`, `tablePlayerNames1`, `etc`	This parameter is used to pass the names of the players to the game. It is useful only if you have your own player matching system. If this parameter is not present, the player will need to manually open a table, join a table or invite other players to play. If this parameter is present, the player will not need to manually open a table or invite players as the game will automatically start with the predetermined players.
	
	e.g. if your player matching system determines that John and Mary are to start a game, the parameters to pass to the game opened by John will be:
	
	`playerName: 'John',`
	`tablePlayerNames0: 'John',`
	`tablePlayerNames1: 'Mary'`
	
	and the parameters to pass to the game opened by Mary will be:
	
	`playerName: 'Mary',`
	`tablePlayerNames0: 'John',`
	`tablePlayerNames1: 'Mary'`

`tableID`	This parameter is used to pass the table ID to join to the game. It is useful only if you have your own player matching system. If this parameter is present, the players will not need to manually open a table or invite other players as the game will automatically start with players with the same table ID value.
`tableNoOfPlayers`	This parameter is used only when `tableID` is used. If this parameter is not set, the game will start as soon as it can (e.g. if the game supports 2, 3 and 4 players, it will start as soon as 2 players have joined), otherwise the game will start only when the specified number of players have joined.
`tableNoOfRobots`	This parameter is used only if both the `tableID` and `tableNoOfPlayers` parameters are used. If this parameter is set, the game will start with no more than the specified number of robots.
`robotName`	This parameter controls the name of the robot.

10. Troubleshooting

Q) The Lobby is not working.

A) Please check

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

If it still cannot work:

1. Check the error log of the web server to see if you can find anything;
2. If you are using SELinux, check to see if SELinux is configured to allow the web server to access the files in the lobbyFiles directory;
3. Check to see if PHP safe mode is turned off. If it is turned on, turn it off.

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

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

11. Document Modification History

Version	Date	Description
1.0.0	2014-07-09	First Draft
1.1.0	2014-08-20	Updated HTML5 codes
1.2.0	2015-07-06	Added information about memcached server
1.3.0	2020-03-26	Removed outdated information Updated information about obscene names filter
1.4.0	2020-06-30	Removed memcached server Added information about WebSocket server
1.5.0	2021-07-21	Removed some outdated information
1.5.1	2022-07-27	Removed googleAPIKey which is no longer needed