Goama Games Tournament Platform - Game Integration Requirements (Confidential)

Document Information

Version History

Version #	Date	Prepared By	Reviewed By	Description
1.0	07nd May 2020	Rezwanul Huda		Initial draft
1.1	08th July 2020	Shakib Hossain		Initial draft including all the sdk functions

Table of Contents

Introduction

This document provides a high level guideline on specific requirements for developing H5 games that can be integrated with the GoGames tournament platform. Specific api documentation, javascript library and other required SDK will be provided as required.

1 Basic requirements

1. The games should be delivered in HTML5 and any javascript files combined in 1 file that is minified and obfuscated
2. The uncompressed size of the payload should be less than 7 MB
3. The package should be hostable in a web server
4. The games should be able to work in full screen mode
5. Games should be able to include SDK provided by GoGames and interact with the SDK to communicate with the Tournament Platform.

2 Installing SDK

Game developers can install the SDK from goama in two ways.

2.1 Build File Provided By GOAMA

Goama SDK can be installed in a game by including the build file into the games HTML page. Contact to obtain the build file.

2.2 JavaScript Package

Goama SDK can also be included in the games through npm/yarn installation. As the package resided in a private repository in BitBucket game developers have to contact GOAMA to get the link for installing the SDK.

Install the package with the below command:

npm i git+https://goama_guest:zZLQPUAhGSLnQeL7jeJc@bitbucket.org/gogamesteam/js-game-integration.git

3 Game interaction with the Tournament Platform

The H5 game will be hosted in an iFrame within an HTML document. GoGames will provide a javaScript SDK that should be used to interact with the tournament platform.

The SDK provides methods for communicating with the parent window. Functionalities like creating/updating game and user specific data - e.g. high scores, perks earned etc, letting the Tournament Platform know that a

game has ended, paused or resumed etc. and listening to events passed on from the parent window, like pause, resume, quit etc.

3.1 Retrieving User Game Data

The H5 game should not store any user progress data in the browser's local storage for future retrieval. Instead, it should call the SDK method and wait for the user's data in a call back method that the game supplies and use the provided data to load the user's progress. The method supports a default data parameter. If a user's data is not available from before, the default data is returned to the game.

Example:

```
const defaultData = {"best": 123, "coins": 1234};

function gameDataCallback(data) {
  // Do something with the gameData
  console.log(data);
}

// Through build file
GGSDK.getGameData(defaultData, gameDataCallback);

// Through npm installation
import * as GGSDK from "js-game-integration";
GGSDK.getGameData(defaultData, gameDataCallback);
```

3.2 Sending Game Over Event

Whenever a game play has ended, the game should call a specific SDK function with the specific score for this game play. This function must be called with the score of the last game play session.

Example:

```
const score = 1234;
// Through build file
GGSDK.gameOver(score);

// Through npm installation
import * as GGSDK from "js-game-integration";
GGSDK.gameOver(score);
```

3.3 Saving User Game Data

Whenever the game needs to save the user's game data, e.g. when a game is over, then the H5 Game should call an SDK method to save the user's data. The data format can be game dependent and game developers are free to choose the data structure. The data should be provided in json format.

Example:

```
const dataToSave = {"best": 123, "coins": 1234};

// Through build file
GGSDK.saveGameData(defaultData);

// Through npm installation
import * as GGSDK from "js-game-integration";
GGSDK.saveGameData(defaultData);
```

3.4 Sending In Game Pause Event

Whenever a user pauses a game from inside the game, the game must send the paused event through the gamePaused function defined in the SDK.

Example:

```
// Through build file
GGSDK.gamePaused();

// Through npm installation
import * as GGSDK from "js-game-integration";
GGSDK.gamePaused();
```

3.5 Sending In Game Resume Event

Whenever a user resumes a game from inside the game, the game must send the resumed event through the gameResumed function defined in the SDK.

Example:

```
// Through build file
GGSDK.gameResumed();

// Through npm installation
import * as GGSDK from "js-game-integration";
GGSDK.gameResumed();
```

3.6 Sending Game Load Complete Event

Whenever a game has completely fetched all of it's necessary resources the game must send the game load finished event through the gameLoaded function defined in the SDK.

Example:

```
// Through build file
GGSDK.gameLoaded();

// Through npm installation
import * as GGSDK from "js-game-integration";
GGSDK.gameLoaded();
```

3.7 Listen Pause Event Passed From Parent

Games must listen for the pause event to handle accordingly when the event is passed from the parent window.

Example:

```
function callback() { // Handle Pause Event Accordingly }
// Through build file
GGSDK.listenPaused(callback);

// Through npm installation
import * as GGSDK from "js-game-integration";
GGSDK.listenPaused(callback);
```

3.8 Listen Resume Event Passed From Parent

Games must listen for the resume event to handle accordingly when the event is passed from the parent window.

Example:

```
function callback() { // Handle Resume Event Accordingly }
// Through build file
GGSDK.listenResumed(callback);

// Through npm installation
import * as GGSDK from "js-game-integration";
GGSDK.listenResumed(callback);
```

3.9 Listen Quit Event Passed From Parent

Games must listen for the pause event to handle accordingly when the event is passed from the parent window. Whenever a game receives the quit event, it must also send out a gameOver & saveGameData events with current progress and quit the game for the user.

Example:

```
function callback() { // Handle Quit Event Accordingly }
// Through build file
GGSDK.listenQuit(callback);

// Through npm installation
import * as GGSDK from "js-game-integration";
GGSDK.listenQuit(callback);
```