Version 1.2

Published on 18th October, 2016

Table of Content

Change Log

Version 1.2 (18th October 2016)

1. Added new method to refresh User Information manually for guest handling and sign out/sign in purposes.

Version 1.1 (20th February 2016)

1. Added PlayMaker custom actions for Super API Kongregate.
2. New Client API supported - Use Item Instance (Virtual Goods).
3. New class ItemInstance introduced to store Item Instances used in following methods:
    a. Kongregate.RequestUserItemList
    b. Kongregate.GetPurchasedItems
4. Following methods will now accept UnityAction instead of GameObject & Method String:
    a. Kongregate.PurchaseItem
    b. Kongregate.PurchaseItems
    c. Kongregate.RequestUserItemList
    d. Kongregate.UseItemInstance

Plugin Overview

Super API plugin provides interfaces for any game developed with **Unity** to integrate to **Kongregate API** with less worries on the technical complexities when integrating towards Kongregate's **Konduit Platform**. This plugin supports functionalities which are listed as supported by **Kongregate** for Unity such as Leaderboards, Virtual Goods, Social and Chat integrations.

Setting Up (Unity C#)

First of all, in order to make use of Super API functions, you will first need to import the plugin from Asset store and add the following line into the script which you intend to implement the function:

```
using UnityEngine;
using SuperApi;
```

Code 1: Setting Up

Then, in the class which handles the Kongregate integration (usually in the Score manager or Statistic manager class), initialize by invoking “Kongregate.Initialize()” in Start method:

```
void Start()
{
    Kongregate.Initialize();
}
```

Code 2: Initializing

Once it has been initialized, the class can start to make use of any particular function by invoking the static method and passing in required parameters. For detailed steps, please see individual function list in Client API List section.

Client API List

Following are the list of client API functions supported by this plugin.

1. Submit Statistics (Leaderboards & Highscores)
2. Purchase Items (Virtual Goods)
3. Request User Item List (Virtual Goods)
4. Use Item Instance (Virtual Goods)
5. Show Reload Kreds Dialog (Virtual Goods)
6. Show Feed Post Box (Social Integration)
7. Show Shout Post Box (Social Integration)
8. Show Custom Tab (Chat Integration)
9. Close Custom Tab (Chat Integration)
10. Display Message on Custom Tab (Chat Integration)
11. Refresh User Info (Account Information) - **New in Version 1.2**

Details of prerequisites, implementations and results of each function will be explained in following sections.

1. Submit Statistics (Leaderboards & Highscores)

This function allows your game to submit statistics which are used to implement Leaderboards & Highscores feature in Kongregate.

The below screenshot shows the High Score tab which displays the player with the highest score achieved for this week:

Image: High Score

Prerequisites

Statistics can be configured as one of the step when you uploading your game on Kongregate. There are 3 types of statistics with different scoring mechanisms.

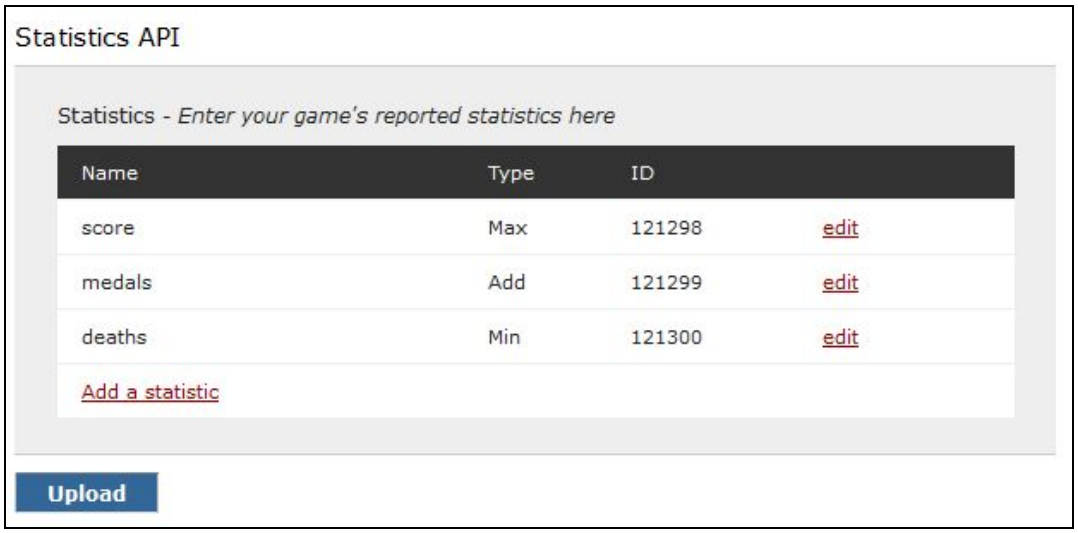

Image: Statistics Setup

Statistics type and their usage are explained below, choose one of the type which relates to the type of statistic you would like to implement:

- **Max:** The value on the server will be replaced if the new value is higher, for example a high score.
- **Min:** The value on the server will be replaced if the new value is lower, for example the lowest time for completing a lap.
- **Add:** The new value will be added to the value stored on the server, for example total number of coins collected. This can be used for statistics which are cumulative.
- **Replace:** The new value will always overwrite the value on the server, this can be useful for statistics that need to go either up or down, such as a player ranking.

Important: Make sure the following is checked in order for the statistics to appear in High Score board.

Image: Display in Leaderboards

Implementation

The static method Kongregate.SubmitStatistics(string statName, string statValue) can be invoked to submit the statistic values to Kongregate. See below for reference:

```
public void SubmitStatistic()
{
    string statName = "score";
    string statValue = ScoreManager.currentScore;
    Kongregate.SubmitStatistic(statName, statValue);
}
```

Code: Submit Statistics

Result

Once the statistic has been submitted, the new submitted value will be available in the High Score board. If it isn't working as expected, please see Troubleshooting step below.

Troubleshooting

To ensure that the Submit Statistic function was invoked, perform the following steps:

1. Append the browser URL with "?debug_level=4" at the back. For example:

   http://www.kongregate.com/games/<username>/<yourgamename>**?debug_level=4**

2. Open Developer Web Console in your browser which is available in popular browsers (e.g. Chrome, Firefox, IE)
3. Run the game up until the point of Submit Statistic expected to be called.
4. In the Web Console log, there should be similar logging as below:

```
[Game] StatServices.submit ( score, 950000 )
[Konduit] [Game->Service]: stat.submit: {"stats":[{"value":950000,"name":"score"}],"req.id":1}
Object { flushmedium: Array[1] }
Object { statList: 1 }
[Konduit] Kongregate Packet[kong_web] stat.submit: {"stats":{"121298":950000},"success":true}
[Konduit] [score-ServerUpdate]: submitted=950000
[Konduit] [Service->Game]: stat.submit: {"stats":{"121298":950000},"success":true}
```

Image: Statistic Submitted Console Logs

If the logs are appearing, it means that your statistics are being submitted to server. If it doesn't, please following general troubleshooting guide at the General Troubleshooting section.

2. Purchase Items (Virtual Goods)

This function allows your game to request for a Purchase Items window which is used to implement Virtual Goods feature in Kongregate.

The below screenshot shows the Purchase Items window when the function is invoked:

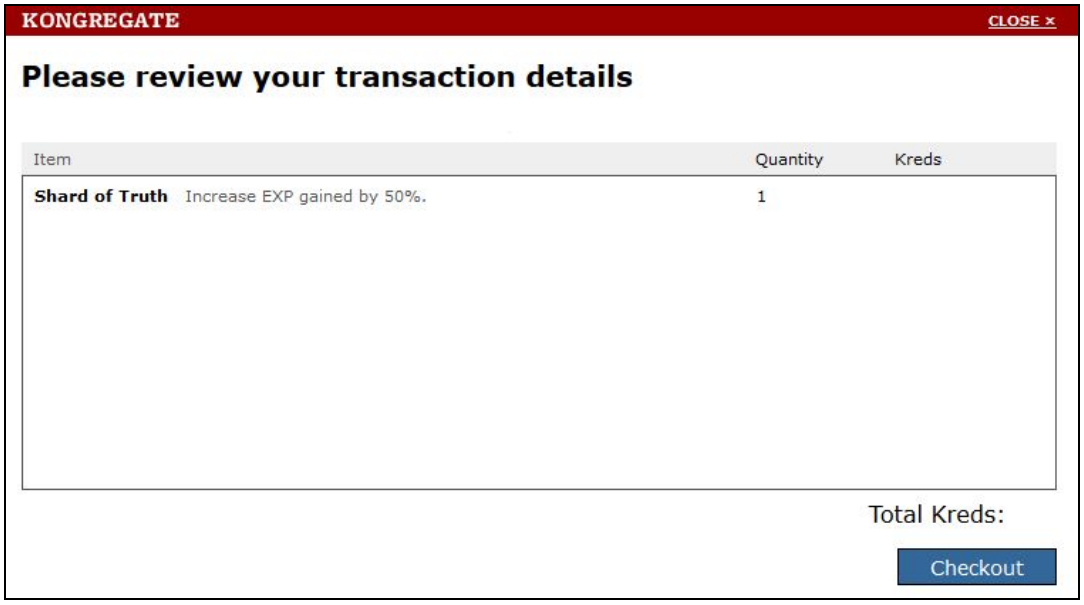

Image: Purchase Items Window

Prerequisites

Virtual Goods can be configured only after you have uploaded your game with or without preview mode in the “Manage Item” page. You can access the page by appending “/items” at the back of your game’s URL. For example:

http://www.kongregate.com/games/<username>/<yourgamename>**/items**

The following page shows that an item with identifier “0_sword” has been added to the Virtual Goods with price of 100 Kreds:

Image: Virtual Items Setup

Important: The unique “identifier” value will be the value used to refer to the particular Virtual Item when invoking Purchase Items function. So, do ensure the identifiers are kept accessible by your code when invoking Purchase Items function.

Implementation

There are two static methods to invoke this function, see below:

1. Kongregate.PurchaseItem(string itemId, UnityAction<bool> eventCallback) - To purchase one item with callback

```
public void PurchaseItem()
{
    string itemId = ItemManager.selectedItemId;
    Kongregate.PurchaseItem(itemId, OnPurchaseItemResult);
}
```

Code: Purchase Single Item with Callback

2. Kongregate.PurchaseItems(string[] itemIdList, UnityAction<bool> eventCallback) - To purchase list of items with callback

```
public void PurchaseItems()
{
    string[] itemIdList = ItemManager.selectedItemIds;
    Kongregate.PurchaseItems(itemIdList, OnPurchaseItemResult);
}
```

Code: Purchase List of Items with Callback

The callback will be invoked once the transaction completes, thus, you could implement a method to wait for the callback with a boolean result:

```
public void OnPurchaseItemResult(bool success)
{
    if(success)
    {
        UIManager.DisplayPurchaseSuccess(); // This could be some UI method
        GetPurchasedItems(); // Retrieving list of purchased items
    }
    else
    {
        UIManager.DisplayPurchaseFailed();  // This could be some UI method
    }
}
```

Code: Purchase Item Callback Method

Result

Once user has either completed the transaction or cancelled the transaction and returned to the game window, the callback should happen with either successful or unsuccessful result. On top of that, the Request User Item List function will now return result with the purchased item instances.

Troubleshooting

To ensure that the Purchase Items function was invoked, perform the following steps:

1. Append the browser URL with “?debug_level=4” at the back. For example:

   http://www.kongregate.com/games/<username>/<yourgamename>**?debug_level=4**

2. Open Developer Web Console in your browser which is available in popular browsers (e.g. Chrome, Firefox, IE)
3. Run the game up until the point of Purchase Items expected to be called.
4. In the Web Console log, there should be similar logging as below:

```
[Konduit] [Game->Service]: mtx.checkout: {"items":["a0001","a0002"],"req.id":1}
"[Konduit] URL: http://www.kongregate.com/games/ <username> / <game title> /checkouts/
```

Image: Purchase Item Logs

5. Once the user completed or cancelled the transaction, the callback method should be invoked and there should be similar logging as below:

```
[Konduit] Event received from JS: purchase_result
Object { eventReceived: Object }
[Konduit] Processing external message: purchase_result, data: [object Object]
[Konduit] External message received: purchase_result: {"success":false}
[Konduit] [Service->Game]: ext.msg: {"opcode":"purchase_result","data":{"success":false}}
[Game] Purchase result: purchase_result: {"success":false}
```

Image: Purchase Item Callback Logs

If the logs are appearing and purchase item window appears with the correct items and callback was received, it means that your purchase item function and its callback method was invoked successfully.

If window appeared without any items listed, it means that the item identifiers provided were incorrect or in wrong format.

If purchase window did not appear or callback did not happen, please following general troubleshooting guide at the General Troubleshooting section.

3. Request User Item List (Virtual Goods)

This function allows your game to request for the list of purchased items which is used to implement Virtual Goods feature in Kongregate.

Prerequisites

Items successfully purchased through Purchase Items function will be retrievable through this function.

Important: The value returned from the callback will be an array of item identifier in string type.

Implementation

The static method Kongregate.RetrievePurchasedItems(UnityAction<ItemInstance[]> eventCallback) can be used to invoke this function, see below:

```
public void GetPurchasedItems()
{
    Kongregate.RequestUserItemList(OnGetPurchasedItems);
}
```

Code: Request User Item List with Callback

The callback will be invoked once the transaction completes, thus, you could implement a method to wait for the callback with an array of ItemInstance as a result:

```
public void OnGetPurchasedItems(ItemInstance[] itemInstanceList)
```

```
{
    UIManager.RefreshInventory(itemInstanceList); // This could be some UI
method
}
```

Code: Request User Item List Callback Method

If you would like to access to the cached item list retrieved after the callback without retrieving from the server again, it is possible by calling Kongregate.GetPurchasedItems() as per below:

```
public void LoadItemList()
{
    if(Kongregate.IsInitialized())
    {
        ItemInstance[] itemInstances = Kongregate.GetPurchasedItems();
        // Your implementations here.
    }
}
```

Code: Get Purchased Item Method

Result

Once the Request User Item List function is invoked, Kongregate API will retrieve user items and the callback method passed in will be received upon retrieving successfully.

Troubleshooting

To ensure that the Request User Item List function was invoked, perform the following steps:

1. Append the browser URL with “?debug_level=4” at the back. For example:

http://www.kongregate.com/games/<username>/<yourgamename>**?debug_level=4**

2. Open Developer Web Console in your browser which is available in popular browsers (e.g. Chrome, Firefox, IE)
3. Run the game up until the point of Request User Item List expected to be called.
4. In the Web Console log, there should be similar logging as below:

```
[Konduit] [Game->Service]: mtx.item_instances: {"user":"kevinyap87","req.id":2}
[Konduit] Requesting item instance list from server for user kevinyap87
```

Image: Request User Item List Logs

6. Once the retrieval result returned, the callback method should be invoked and there should be similar logging as below:

```
[Konduit] Item instance list received for kevinyap87
[Konduit] [Service->Game]: mtx.item_instances: {"success"
{"data":null,"id":10784966,"identifier":"a0000","descript
cost gold.","remaining_uses":1,"name":"Champion's Lance"}
{"data":null,"id":10786211,"identifier":"a0001","descript
50%.","remaining_uses":1,"name":"Shard of Truth"},{"data"
{"data":null,"id":10809508,"identifier":"a0001","descript
50%.","remaining_uses":null,"name":"Gladiator's Fame"},{"
{"data":null,"id":10793803,"identifier":"a0002","descript
50%.","remaining_uses":1,"name":"Gladiator's Fame"}],"use
[Game] Item instance list received
```

Image: Request User Item List Callback Logs

If the logs are appearing and callback was received, it means that your request user item list function and its callback method was invoked successfully. If it doesn't or callback did not happen, please following general troubleshooting guide at the General Troubleshooting section.

4. Use Item Instance (Virtual Goods)

This function allows your game to use an item instance which is used to implement Virtual Goods feature in Kongregate.

Prerequisites

Items retrieved from Request User Items List can be used with the Use Item Instance function.

Important: The value uniquely representing each Item Instance is the Instance Id returned from Request User Items List - where the Instance Id will be sent when invoking Use Item Instance function.

Implementation

The static method Kongregate.UseItemInstance(string itemInstanceId, UnityAction<bool> eventCallback) can be used to invoke this function, see below:

```
public void UseItemInstance(ItemInstance itemInstance)
{

    Kongregate.UseItemInstance(itemInstance.GetInstanceId(),
OnUseItemInstanceResult);
}
```

Code: Use Item Instance with Callback

The callback will be invoked once the transaction completes, thus, you could implement a method to wait for the callback with a boolean result:

```
public void OnUseItemInstanceResult(bool success)
{
    if(success)
    {
        UIManager.DisplayUseItemSuccess(); // This could be some UI method
        PerformItemEffect(); // Perform effects for the item used
        GetPurchasedItems(); // Refresh list of purchased items
    }
    else
    {
        UIManager.DisplayUseItemFailed();  // This could be some UI method
    }
}
```

Code: Use Item Instance Callback Method

Result

Once the Use Item Instance function is invoked, Kongregate API will use the item instance and the callback method passed in will be received upon item used successfully.

Troubleshooting

To ensure that the Use Item Instance function was invoked, perform the following steps:

5. Append the browser URL with “?debug_level=4” at the back. For example:

   http://www.kongregate.com/games/<username>/<yourgamename>**?debug_level=4**

6. Open Developer Web Console in your browser which is available in popular browsers (e.g. Chrome, Firefox, IE)
7. Run the game up until the point of Use Item Instance expected to be called.
8. In the Web Console log, there should be similar logging as below:

```
[Konduit] [Game->Service]: mtx.use_item_instance: {"req.id":4,"id":10976704}
[Konduit] Requesting item instance use for kevinyap87, item ID: 10976704
```

Image: Use Item Instance Logs

7. Once the item use result returned, the callback method should be invoked and there should be similar logging as below:

```
[Konduit] Item instance use result for item instance id:10976704
[Konduit] [Service->Game]: mtx.use_item_instance: {"req.id":4,"id":10976704,"success":true}
[Game] Item use result
```

Image: Use Item Instance Callback Logs

If the logs are appearing and callback was received, it means that your use function and its callback method was invoked successfully. If it doesn't or callback did not happen, please following general troubleshooting guide at the General Troubleshooting section.

5. Show Reload Kreds Dialog (Virtual Goods)

This function allows your game to request for Reload Kreds Dialog to be shown which is used to implement Virtual Goods feature in Kongregate.

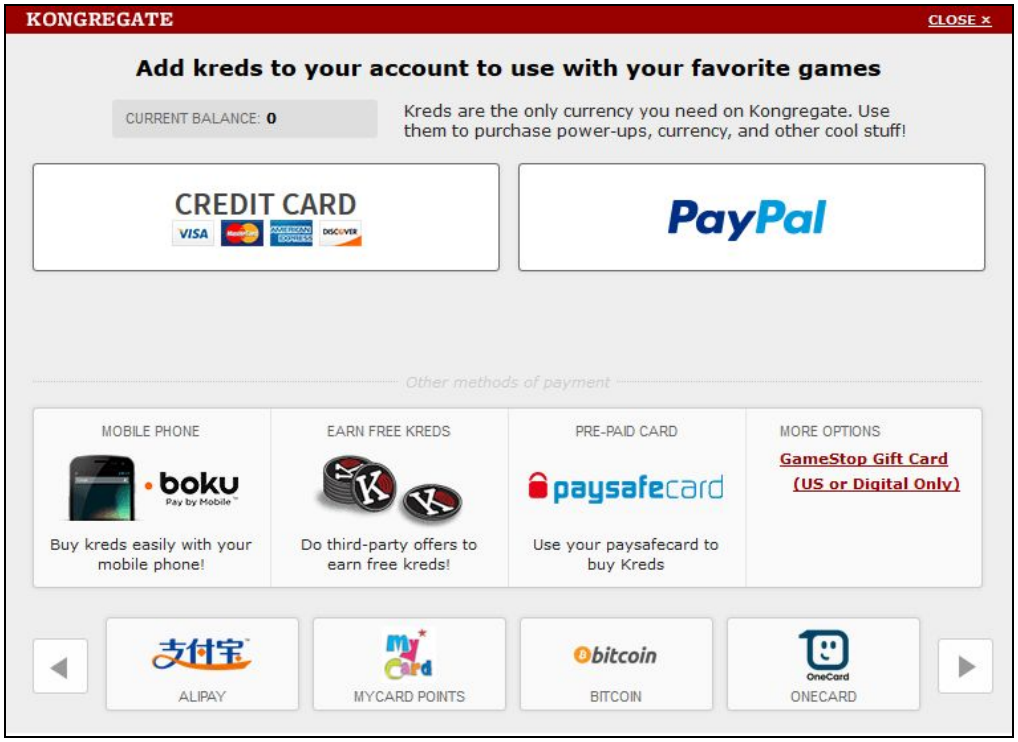

Image: Show Reload Kreds Dialog

Important: Kongregate does not provide callback for this function, it is important for your game to react accordingly disregard of the reload kreds progress.

Implementation

There are two static methods can be used to invoke this function, see below:

1. Kongregate.ShowReloadKredsDialog() - this should be used if user is running game from a Desktop platform.

```
public void ShowReloadKredsDialog()
{
    Kongregate.ShowReloadKredsDialog();
}
```

Code: Show Reload Kreds Dialog

2. Kongregate.ShowReloadKredsDialog() - this should be used if user is running game from a Mobile platform.

```
public void ShowMobileReloadKredsDialog()
{
    Kongregate.ShowMobileReloadKredsDialog();
}
```

Code: Show Mobile Reload Kreds Dialog

Result

Once the Show Reload Kreds Dialog function is invoked, the dialog should appear with payment options available based on the user country.

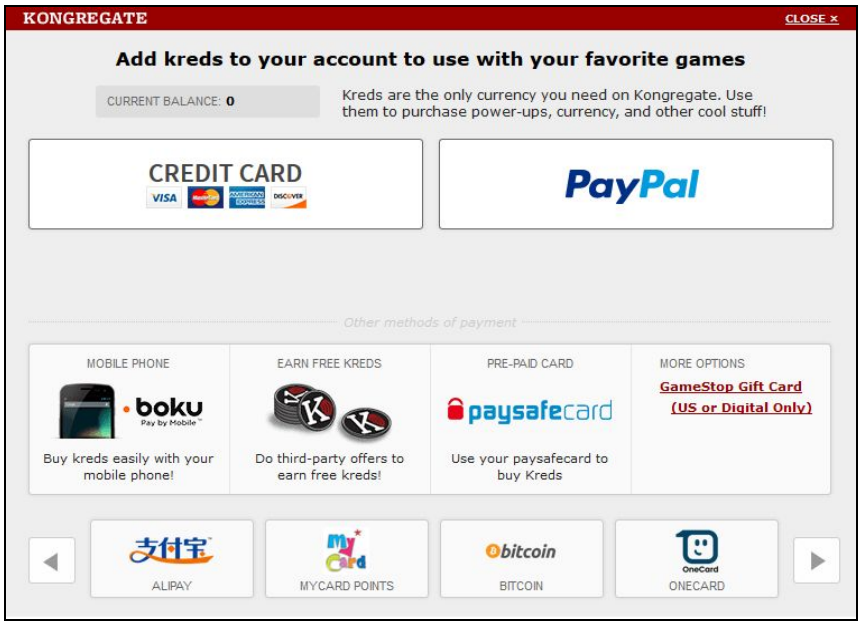

Image: Show Reload Kreds Dialog Result

Troubleshooting

To ensure that the Show Reload Kreds Dialog function was invoked, perform the following steps:

1. Append the browser URL with “?debug_level=4” at the back. For example:

   http://www.kongregate.com/games/<username>/<yourgamename>**?debug_level=4**

2. Open Developer Web Console in your browser which is available in popular browsers (e.g. Chrome, Firefox, IE)
3. Run the game up until the point of Show Reload Kreds Dialog expected to be called.
4. In the Web Console log, there should be similar logging as below:

```
[Konduit] [Game->Service]: mtx.kred_purchase: {"purchase_method":"mobile","req.id":3}
```

Image: Show Reload Kreds Dialog Logs

If the reload kred dialog and logs are appearing, it means that your Show Reload Kreds Dialog was invoked successfully. If it doesn't, please following general troubleshooting guide at the General Troubleshooting section.

6. Show Feed Post Box (Social Integration)

This function allows your game to request for Feed Post Box to be shown with a message which is used to implement Social Integration feature in Kongregate. This function is usually used to post in-game achievement events in Kongregate feeds.

Image: Show Feed Post Box with Custom Image

Important: Kongregate does not provide callback for this function, it is important for your game to react accordingly disregard of the feed post box progress.

Implementation

There are two static methods can be used to invoke this function, see below:

1. Kongregate.ShowFeedPostBox(string feedMessage) - this will post feed message without custom image

```
public void ShowFeedPostBox()
{
    string feedMessage = SocialManager.GetAchieveLevelFeedMessage();
    Kongregate.ShowFeedPostBox(feedMessage);
```

```
}
```

Code: Show Feed Post Box without Custom Image

2. Kongregate.ShowFeedPostBoxWithImage(string feedMessage, string imageUri) - this will post feed message without custom image

```
public void ShowFeedPostBoxWithImage()
{
    string feedMessage = SocialManager.GetAchieveLevelFeedMessage();
    string imageUri = SocialManager.GetAchieveLevelFeedImageUri();
    Kongregate.ShowFeedPostBoxWithImage(feedMessage, imageUri);
}
```

Code: Show Feed Post Box with Custom Image

Result

Once the Show Feed Post Box function is invoked, the dialog should appear with the feed message and custom image.

Image: Show Feed Post Box with Custom Image

Troubleshooting

To ensure that the Show Feed Post Box function was invoked, perform the following steps:

1. Append the browser URL with “?debug_level=4” at the back. For example:

   http://www.kongregate.com/games/<username>/<yourgamename>**?debug_level=4**

2. Open Developer Web Console in your browser which is available in popular browsers (e.g. Chrome, Firefox, IE)
3. Run the game up until the point of Show Feed Post Box expected to be called.
4. In the Web Console log, there should be similar logging as below:

Image: Show Feed Post Box Logs

If the feed post box dialog and logs are appearing, it means that your Show Feed Post Box function was invoked successfully. If it doesn’t, please following general troubleshooting guide at the General Troubleshooting section.

7. Show Shout Box (Social Integration)

This function allows your game to request for Shout Box to be shown with a message which is used to implement Social Integration feature in Kongregate. This function is usually used to post social interaction event shout-out to friends in Kongregate feeds.

Image: Show Shout Box

Important: Kongregate does not provide callback for this function, it is important for your game to react accordingly disregard of the shout box progress.

Implementation

The static method Kongregate.ShowShoutBox(string shoutMessage) can be used to invoke this function, see below:

```
public void ShowShoutBox()
{
    string shoutMessage = SocialManager.GetCityOutOfGoldShoutMessage();
    Kongregate.ShowShoutBox(shoutMessage);
}
```

Code: Show Shout Box

Result

Once the Show Shout Box function is invoked, the dialog should appear with the shout message.

Image: Show Shout Box

Troubleshooting

To ensure that the Show Shout Box function was invoked, perform the following steps:

1. Append the browser URL with “?debug_level=4” at the back. For example:

   http://www.kongregate.com/games/<username>/<yourgamename>**?debug_level=4**

2. Open Developer Web Console in your browser which is available in popular browsers (e.g. Chrome, Firefox, IE)
3. Run the game up until the point of Show Shout Box expected to be called.
4. In the Web Console log, there should be similar logging as below:

```
[Konduit] [Game->Service]: chat.shoutbox: {"req.id":4,"shout_message":"City is poor!"}
```

Image: Show Shout Box Logs

If the shout box dialog and logs are appearing, it means that your Show Shout Box function was invoked successfully. If it doesn't, please following general troubleshooting guide at the General Troubleshooting section.

8. Show Custom Tab (Chat Integration)

This function allows your game to request for Custom Tab to be shown which is used to implement Chat Integration feature in Kongregate. This function is usually used to implement Custom notifications or match chat feature.

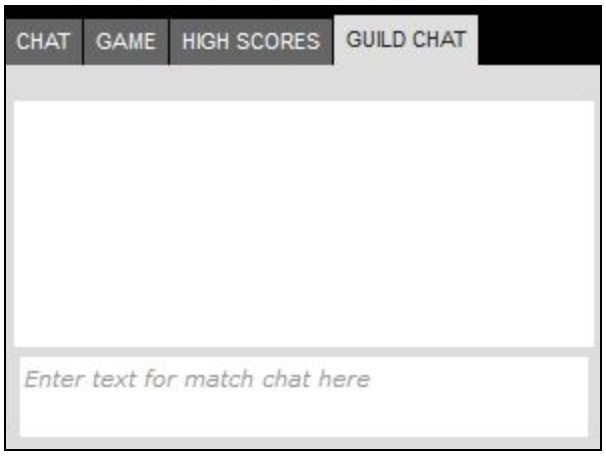

Image: Show Custom Tab

Important: Kongregate does not provide callback for this function, it is important for your game to react accordingly disregard of the custom tab progress.

Prerequisites

This function will only work if Kongregate Developer Support team has enabled the Chat Integration feature for your game. You can request for it to be enabled if you want to implement this function for your game.

Implementation

The static method Kongregate.ShowCustomTab(string tabName, string tabDesc, double canvasSize) can be used to invoke this function, see below:

```
public void ShowCustomTab()
{
    string tabName = GuildManager.GetCustomChatTabName();
    string tabDesc = GuildManager.GetCustomChatTabDesc();
    double canvasSize = GuildManager.GetCustomChatCanvasSize();
    Kongregate.ShowCustomTab(tabName, tabDesc, canvasSize);
}
```

Code: Show Custom Tab

Result

Once the Show Custom Tab function is invoked, the new custom tab should appear on the right-hand side of your game window.

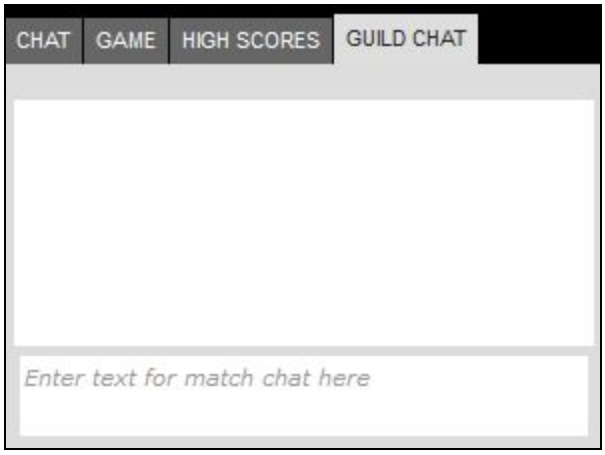

Image: Show Custom Tab

Troubleshooting

To ensure that the Show Shout Box function was invoked, perform the following steps:

1. Append the browser URL with “?debug_level=4” at the back. For example:

   http://www.kongregate.com/games/<username>/<yourgamename>**?debug_level=4**

2. Open Developer Web Console in your browser which is available in popular browsers (e.g. Chrome, Firefox, IE)
3. Run the game up until the point of Show Custom Tab expected to be called.
4. In the Web Console log, there should be similar logging as below:

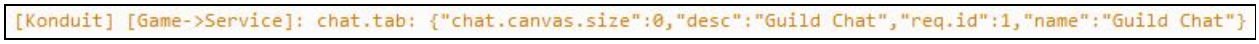

Image: Show Custom Tab Logs

If the custom tab and logs are appearing, it means that your Show Custom Tab function was invoked successfully. If it doesn’t, please following general troubleshooting guide at the General Troubleshooting section.

9. Close Custom Tab (Chat Integration)

This function allows your game to request for Custom Tab to be closed which is used to implement Chat Integration feature in Kongregate that was shown with Show Custom Tab function.

Important: Kongregate does not provide callback for this function, it is important for your game to react accordingly disregard of the custom tab progress.

Prerequisites

This function will only work if Kongregate Developer Support team has enabled the Chat Integration feature for your game. You can request for it to be enabled if you want to implement this function for your game.

Implementation

The static method Kongregate.CloseCustomTab() can be used to invoke this function:

```
public void CloseCustomTab()
{
    Kongregate.CloseCustomTab();
}
```

Code: Close Custom Tab

Result

Once the Close Custom Tab function is invoked, the custom tab will disappear.

Troubleshooting

To ensure that the Close Custom Tab function was invoked, perform the following steps:

1. Append the browser URL with “?debug_level=4” at the back. For example:

   http://www.kongregate.com/games/<username>/<yourgamename>**?debug_level=4**

2. Open Developer Web Console in your browser which is available in popular browsers (e.g. Chrome, Firefox, IE)
3. Run the game up until the point of Close Custom Tab expected to be called.
4. In the Web Console log, there should be similar logging as below:

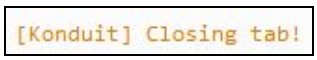

Image: Close Custom Tab Logs

If the custom tab and logs are appearing, it means that your Close Custom Tab function was invoked successfully. If it doesn’t, please following general troubleshooting guide at the General Troubleshooting section.

10. Display Message on Custom Tab (Chat Integration)

This function allows your game to request for a message to be displayed in Custom Tab which is used to implement Chat Integration feature in Kongregate.

Image: Display Message on Custom Tab

Important: Kongregate does not provide callback for this function, it is important for your game to react accordingly disregard of the display message progress.

Prerequisites

This function will only work if Kongregate Developer Support team has enabled the Chat Integration feature for your game. You can request for it to be enabled if you want to implement this function for your game.

Implementation

The static method Kongregate.DisplayMessage(string message, string name) can be used to invoke this function:

```
public void DisplayMessage()
{
    string message = GuildManager.GetRecentMessage();
    string name = GuildManager.GetRecentName();
    Kongregate.DisplayMessage(message, name);
}
```

Code: Display Message

Result

Once the Display Message function is invoked, the message will be displayed in Custom Tab.

Image: Display Message on Custom Tab

Troubleshooting

To ensure that the Display Message function was invoked, perform the following steps:

1. Append the browser URL with “?debug_level=4” at the back. For example:

   http://www.kongregate.com/games/<username>/<yourgamename>**?debug_level=4**

2. Open Developer Web Console in your browser which is available in popular browsers (e.g. Chrome, Firefox, IE)
3. Run the game up until the point of Display Message expected to be called.
4. In the Web Console log, there should be similar logging as below:

```
[Konduit] [Game->Service]: chat.disp: {"data":"Guys, raid now!","user":"Guild Leader","req.id":5}
```

Image: Display Message Logs

If the custom tab and logs are appearing, it means that your Display Message function was invoked successfully. If it doesn’t, please following general troubleshooting guide at the General Troubleshooting section.

11. Refresh User Info (Account Information)

New in Version 1.2

This function allows your game to refresh the below user information:

1. User Id
2. Username
3. Game Authentication Token

Prerequisites

This function will only work if Kongregate API has been initialized.

Implementation

The static method Kongregate.RefreshUserInfo() can be used to invoke this function:

```
public void RefreshUserInfo()
{
      Kongregate.RefreshUserInfo(OnUserInfoReceived);
}
```

Code: Refresh User Info

The callback will be invoked once the refresh completes, thus, you could implement a method to wait for the callback (without parameters, user info can be accessed through general functions):

```
public void OnUserInfoReceived()
{
      userIdText.text = Kongregate.GetUserId ();
      userNameText.text = Kongregate.GetUserName ();
      gameAuthToken.text = Kongregate.GetGameAuthToken ();
}
```

Code: Use Item Instance Callback Method

Result

Once the Refresh User Info callback is received, the new information would be retrievable through the following general functions:

1. Kongregate.GetUserId() - returns the User Id if plugin has been connected.
2. Kongregate.GetUserName() - returns the User Name if plugin has been connected.
3. Kongregate.GetGameAuthToken() - returns the Game Auth Token if plugin has been connected.
4. Kongregate.IsGuest() - returns boolean indicating whether user is a guest.

Troubleshooting

Result should reflect the latest user information, if it doesn't, please ensure that you do not retrieve the user information prior to the callback event is received. The callback event essentially is invoked only after all updated user information is retrieved from Kongregate API.

If it doesn't work, please following general troubleshooting guide at the General Troubleshooting section.

General Functions

The plugin provides general information like User Id, Game Auth Token, etc., which can be accessed using the following:

1. Kongregate.IsInitialized() - returns true if plugin has been connected.
2. Kongregate.GetUserId() - returns the User Id if plugin has been connected.
3. Kongregate.GetUserName() - returns the User Name if plugin has been connected.
4. Kongregate.GetGameAuthToken() - returns the Game Auth Token if plugin has been connected.
5. Kongregate.IsGuest() - returns boolean indicating whether user is a guest.

Note: Refresh User Info function can be used to refresh the user information manually for guest handling or sign-in/sign-out purposes. **- New in Version 1.2**

General Troubleshooting

For verbose logs, the plugin provides cumulative debug logs which can be accessed using the following:

1. Kongregate.GetLog() - returns the cumulative log string.
2. Kongregate.ClearLog() - clears the log string.

The following sample code displays the logs during OnGUI:

```
public void OnGUI()
{
    string debugStr = "Initialized: " + Kongregate.IsInitialized().ToString()
+ "\n";
    debugStr = debugStr + "UserId: " + Kongregate.GetUserId() + "\n";
    debugStr = debugStr + "UserName: " + Kongregate.GetUserName() + "\n";
```

```
    debugStr = debugStr + "GameAuthToken: " + Kongregate.GetGameAuthToken()
+ "\n";
    debugStr = debugStr + "Logs:\n" + Kongregate.GetLog();
    GUIStyle style = new GUIStyle();
    style.wordWrap = true;
    style.stretchHeight = false;
    style.stretchWidth = false;
    GUI.Box(new Rect(10, 10, 500, 500), debugStr, style);
}
```

Code: Display Logs

PlayMaker Support

To complement the use of Super API Kongregate within PlayMaker framework, the following have been created.

Important: To run the use Super Kongregate API with PlayMaker, you need to install the following:

1. PlayMaker 1.8.0
2. Import UnityPackage: Plugins/SuperApi/PlayMakerSupport/SuperKongregateAPI_PlayMakerSupport_V1.1.0
3. Import Globals: Plugins/SuperApi/PlayMakerSupport/PlaymakerGlobals_EXPORTED
4. u Gui Proxy Full - Download from Ecosystem Browser (Optional - only required for Demo Scene)

FSM Template

With this single Super API Kongregate Event Receiver FSM Template, all required events and transitions setup are already in place. You can customize the FSM to suit your project needs.

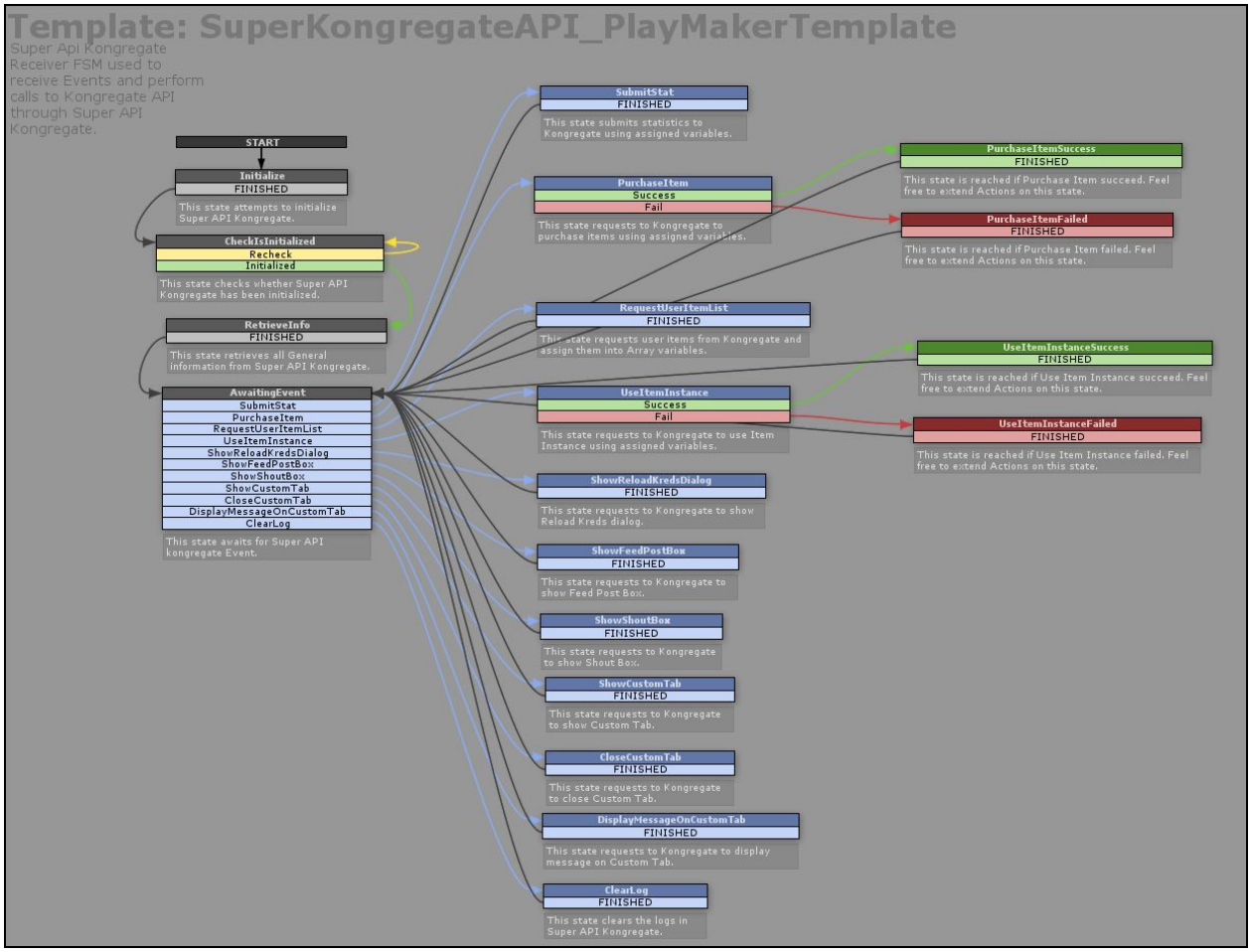

Image: Super API Kongregate - PlayMaker FSM Template

This FSM implements the complete features of Super API Kongregate, whenever the particular event (e.g. SubmitStat) is received, the event properties for the action will be retrieved and the function will be invoked through the Super API Kongregate Custom Actions.

Custom Actions

The following list of custom actions are included in the package:

Image: Super API Kongregate - PlayMaker Custom Actions

Global Variables

The following list of global variables are added to standardize the event properties variables name required for each of the action:

Super Kongregate API		
CUSTOM_TAB_CANVAS_SIZE	-	String
CUSTOM_TAB_DESC	-	String
CUSTOM_TAB_NAME	-	String
DISPLAY_USER_MESSAGE	-	String
DISPLAY_USER_NAME	-	String
FEED_IMAGE_URL	-	String
FEED_MESSAGE	-	String
PUCHASE_ITEM_ID	-	String
SHOUT_MESSAGE	-	String
SUBMIT_STAT_NAME	-	String
SUBMIT_STAT_VALUE	-	String
USE_ITEM_INSTANCE_ID	-	String

Image: Super API Kongregate - PlayMaker Global Variables

Example Demo Scene - PlayMaker

An example demo scene of Super API Kongregate implementation using PlayMaker framework is included in the package.

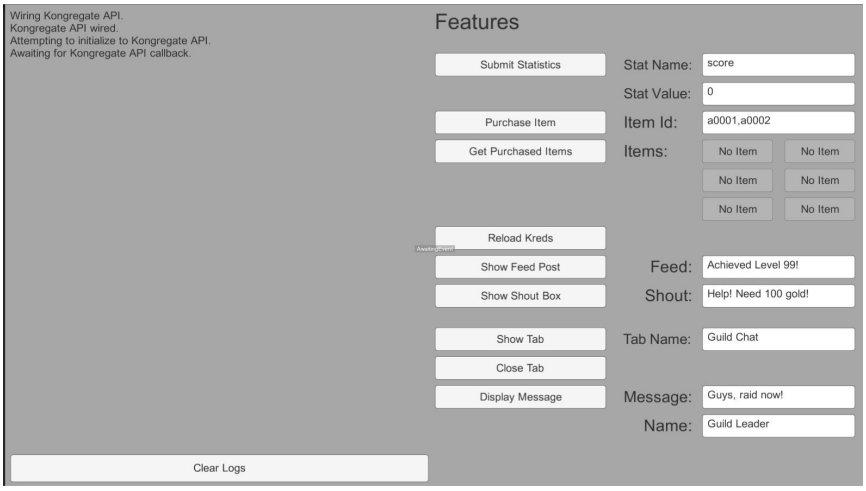

Image: Super API Kongregate - PlayMaker Demo Scene