بسمه تعالی

مستندات SDK ابرینگ
برای استفاده در یونیتی

نسخه 2.6.8

شهریور 1396

فهرست

مقدمه

ابرینگ به عنوان ابزار کامل و جامعی برای نگهداری و مدیریت اطلاعات بازیکنان و بازی ، در خارج از اپلیکیشن، قابلیتهای گسترده ای دارد که از طریق تعامل با وب سرویس های آماده شده ، میتوان از آن استفاده کرد . جهت سهولت استفاده از این سرویس ها در یونیتی ، این SDK آماده شده است که از طریق ساده سازی فرامین و حذف روند های تکرای ، سرعت پیاده سازی و استفاده از این سرویس ها را بالا برده است.

این روزها ، همه از خدمات سرور در بخشهای مختلف اپلیکیشن ها و بازی هایشان استفاده میکنند . خدمات Cloud ، لیدربرد و اچیومنت ، سرویس های آنالتیک یا Push Notification امکانات معمول بسیاری از اپلیکیشن ها و بازی های پر فروش این روز های مارکت های مختلف است .لیست اپلیکیشن ها و بازی های پر فروش های کافه بازار یا دیگر مارکت ها ،گواه این ادعاست.

همچنین بازی های آنلاین ، به راحتی اقدام به جذب یوزر کرده و بسیاری از آنها ، درآمد بالایی از پرداخت های درون برنامه ای دارند. برای اضافه کردن این امکانات به اپلیکیشن و یا بازی ها خود ، به خدمات سرویس دهنده ها نیاز دارید.

به کمک ابرینگ ، شما میتوانید مجموعه وسیعی از امکانات سمت سرور را به اپلیکیشن خود اضاف کنید . و همچنین از راه دور ، رفتار یوزر را زیر نظر داشته و اصلاحاتی روی اپلیکیشن خود انجام دهید . همچنین با ارایه پیشنهادات جذاب و اخبار یا جوایز نرخ user retention را افزایش دهید .

ابرینگ یک سرویس دهنده یا MBAAS (Mobile Backend As Service) ایرانی است که یه صورت کامل در ایران پیاده سازی شده و از سرور های قدرتمند داخلی و خارجی برای کاربران ایرانی استفاده میکند .همچنین ابرینگ ابزاری است که به سازندگان اپلیکیشن ها و بازی ها این امکان را میدهد که ارتباط خود را با یوزر خود در طول مدت زمان استفاد ه از محصول حفظ کنند و همچنین به یوزر ها امکان رقابت و تعامل با دیگران را میدهد و وضعیت آنها را در این رقابت نشان میدهد

به صورت معمول ، پیاده سازی بخش های مرتبط با سرور ، به دلیل دشواری های فنی و پیچیدگی های مختلف کمی سخت است . به این منظور ، سرویس دهنده ابرینگ علاوه بر RestAPI برای استفاده مستقیم از سرور ، SDK های مختلفی برای پلتفرم های مختلف منجمله ANDROID، IOS،UNITY3D و جاوااسکریپت فراهم کرده است .

در کنار استفاده از API های آماده شده ، شما میتوانید برای استفاده راحت تر از امکانات ابرینگ ، از UI های آماده آن برای استفاده در بازی یا اپلیکیشن خود استفاده کنید . به این معنی که فقط به کمک یک خط کد ، یک فرم کامل ثبت نام با امکانات مختلف در اختیار توسعه دهنده قرار میدهد. فرم های لیدر برد آماده ، update و forceupdate و دیگر فرمها ، همکنون آماده به اضافه شدن به پروژه های شما هستند.

تعاریف

در هنگام استفاده از SDK و در این Document با اصطلاحات زیر مواجه خواهید شد . برای داشتن تعریف مشترک ، این کلمات به شکل زیر تعریف و استفاده میشوند :

- Login

  میگوییم . اگرچه تعداد معدودی از فرامین Loginبه فرایند ورود کاربر به سیستم ، بدون نیاز به این فرایند هم قابل فراخوانی هستند ولی بیشتر دستورات ، به نیاز دارند . بنابر این بهتر است در ابتدای بازی این فرایند حتما انجام Login شود.

- Token

  عبارت منحصر به فردی است که توسط سرور بعد از عملیات ثبت نام یا ورود کاربر بوجود میاید و به برنامه ارسال میشود . بعد از دریافت Token ، برای مشخص شدن فرد ارسال کننده دستور به سرور ، باید از این token در دستورات استفاده شود. . تا هنگامی که فرایند Logout اتفاق نیافتد ، token معتبر است و قابلیت استفاده دارد . بنابراین با ثبت این رشته در PlayerPref میتوانید در دفعات بعد ورود به بازی ازآن استفاده کنید .

- Leaderboard

  Leaderboard جدول نگهداری امتیاز ها است که انها را به ترتیب از بالاترین به کمترین Sort میکند و نگه میدارد . بنابراین نفر اول لیدربرد ، بالاترین امتیاز بازی را کسب کرده است.

- Score

امتیاز بازیکن هر بار و معمولا بعد از پایان بازی یا باختن ، در سرور ثبت میشود تا در لیدربرد اضافه شود.

- app_id

نام بازی که در پنل ثبت بازی ، به بازی خود داده اید.

- Player data

اطلاعاتی است که مختص بازیکن بوده و فقط خود بازیکن به آنها دسترسی دارد. مثلا امتیاز یا تعداد ستاره های کسب شده و غیره.

- App data

اطلاعات بازی که همه بازیکنان به آن دسترسی دارند . شامل مواردی مثل اخبار ، ورژن بازی ، اطلاعیه ها ، جوایز ویا برندگان جوایز و امثال این موارد است.

نصب و استفاده

کافی است پکیج مربوطه را در یونیتی Drag & Drop کنید تا به پروژه شما اضافه شود . در این حالت تمامی سرویس های قابل ارائه توسط ابرینگ ، به شکل

Abring.Instance. اسم سرویس (پارامتر ها)

قابل دسترسی است.

تمامی فرامین به SDK به دارای Event هایی هستند که بر اساس پاسخ سرور فراخوانی میشوند . شما میتوانید بنا بر نیاز هایتان از این Event ها استفاده کنید یا نکنید.

همچنین پاسخ سرور در قالب Json به اپلیکیشن ارسال میشود که پس از parse شدن قابل استفاده خواهد بود.

برای استفاده از ابرینگ و همچنین Json کافی است خطوط زیر به کد اضافه شود:

```
using SimpleGamehub;
using SimpleJSON;
```

در زیر یک نمونه به عنوان مثال آورده میشود:

```
Abring.Instance.register(username, password, app_id, "+98912123456",
"aa@bb.cc",
(string s)=> { Debug.Log("success=" + str); },//on successfull
(string s)=> { Debug.Log("fail=" + str);}, //on fail
(string s)=> { Debug.Log("noconnect=" + str); },//on no network
 false);
```

همچنین برای استفاده از Json میتوانید از روش زیر استفاده کنید :

```
var loginJson = JSONNode.Parse(str);
username = loginJson["result"]["username"];
```

راهنمای دستورات

در اکثر دستوراتی که در ادامه میآید قسمتهای مشترکی وجود دارد که در این قسمت به آنها اشاره میشود . در صورتی که یک دستور،از این قاعده مستثنی باشد ، در داکیومنت های آن دستور ، ذکر خواهد شد .

در ابتدای تمام کد هایی که از ابرینگ استفاده میکنند باید این دو خط قرار گیرد

```
using SimpleGameHub;
using SimpleJSON
```

برای دریافت پاسخ های سرور ، یک deligate به شکل زیر تعریف شده است :

```
public delegate void serverresponse(string responses);
```

مقدار responses جوابی است که از سمت سرور در پاسخ به درخواست اجرای دستور باز میگردد که به شکل json است.

و در تمامی دستورات ، سه پارامتر به شکل Successfull , fail , connectionfail قرار دارد که بر اساس این deligate تعریف شده اند .

بنا براین دستوری مانند لاگین به شکل زیر تعریف شده است :

```
Abring.Instance. Login(string username, string password ,
```

serverresponse success,
serverresponse fail,
serverresponse noconnection,
showdebug)

که قسمتهای success, fail,connectionfail به کمک این deligate تعریف شده اند .

success در هنگامی فراخوانی میشود که سرور دستور درخواست شده را اجرا کرده و جواب را برگردانده است . معمولا جواب به شکل json است و جزییات آن در کلید resault قرار دارد.

fail : هرگاه سرور قادر به اجرای دستورات به دلیل نقص در پارامتر های درخواست ارسالی نباشد این رویداد فراخوانی میشود و معمولا به دلیل بیدقتی توسعه دهنده اتفاق میافتد و نباید در اپلیکیشن نهایی دیده شود . در بسیاری از اوقات ، null بودن یکی از پارامتر ها یا شکل نادرست ارسال پارامتر ها ، باعث دیده شدن این رویداد است

connectionfail این رویداد در زمانی فراخوانی مشود که به دلیل مشل اتصال به اینترنت ، ارسال یا دریافت پاسخ با مشکل مواجه شده است

و به عنوان نمونه شکل ساده استفاده از این دستور به شکل زیر میتواند باشد :

```
Abring.Instance. Login("username", "password", success,
fail,connectionfail,false) ;

   void success(string str)
   {
```

```
    Debug.Log("success=" + str);
}

void fail(string str)
{
    Debug.Log("fail=" + str);
}

void connectionfail(string str)
{
    Debug.Log("connectionfail=" + str);
}
```

همچنین یه شکل زیر :

```
Abring.Instance. Login("username", "password", (string s)=>{
Debug.Log("success=" + str);},(string s)=>{ Debug.Log("fail=" + str);},(string s)=>{
Debug.Log("noconnection=" + str);},false) ;
```

اطلاعات خطایابی

در انتهای هر دستور یک متغیر boolean به نام showdebug وجود دارد که در صورت true بودن ، جزییات نحوه اتصال و اطلاعات برگشتی از سرور را به کمک Debug.Log در کنسول یونیتی نمایش میدهد. همچنین تعداد دفعات سعی برای کانکت و اطلاعات لازم در هنگام توسعه اپلیکیشن در این قسمت نمایش داده میشود .

دسته بندی API ها

سرویس های پیاده سازی شده در گروه های زیر قابل دسته بندی هستند :

1- API های مرتبط با بازیکن
   a. Register
   b. Login
   c. Logout
   d. Sessions
   e. Playerget
   f. playergetglobal
   g. playerset
   h. PlayerSetImage
   i. PlayerSetGlobal
   j. playerImageSetGlobal
   k. playerget
2- API های مرتبط با اپلیکیشن
   a. AppGet
   b. AppGetAll

3- API های مرتبط با Leaderboard:
   a. LeaderBoardGet
   b. LeaderBoardGetAll
   c. LeaderboardSetScore
4- API های مرتبط با Friends
   a. Friendlist
   b. FriendsSearch
   c. friendsInvite
   d. friendsAccept
   e. FriendsReject
   f. Friends Cancell Request
   g. Friends UnFriends
5- API های مرتبط با سرویس دهنده ابرینگ
   a. ping

در ادامه لیست این سرویس ها به همراه توضیحات لازم ارائه میشود:

سرویس Register

توضیح : ثبت نام کاربر جدید

با اجرای این دستور ، یک کاربر جدید در سیستم اضافه میشود و با دریافت Token ، بازی ، امکان ارسال فرامینی که به Token نیاز دارند را به دست میآورد .

درصورتی که کاربری با این نام قبلا ثبت نام کرده باشد ، با پیغام خطا مواجه میشود و رویداد OnRegisterFail فراخوانی میشود

```
public void register(string username, string password, string appname, string phone, string email, serverresponse success,serverresponse fail, serverresponse noconnection, bool showdebug)
```

در پایان اجرای این دستور و اگر این دستور موفقیت آمیز باشد :
Userdata.token حاوی توکن دریافتی از سرور است مثلا ممکن است حاوی رشته ای به شکل 0b8d9d91a6c5d7807e980108999c31b4 باشد
UserData.name :نام پلیر
UserData.app_id :نام اپلیکیشن
UserData.email ایمیل کاربر

پاسخ نمونه از سمت سرور :

//on successful

```
{"code":"200","result":"c31b49af0b8d9d91a6c5d7807e980108","message":"Action executed successfully!","debug":{"execution_time":"385 ms"}}
```

//onfail

{"code":"400","result":"","message":"This player already exists","debug":{"execution_time":"26 ms"}}

کد نمونه :

```
Abring.Instance.register(username, password, app_id, "+98912123456",
"aa@bb.cc",
(string s)=> { },//on successfull
(string s)=> { }, //on fail
(string s)=> { },//on no network
 false);
```

سرویس login

توضیح : ورود کاربری که قبلا ثبت نام کرده است

با اجرای این دستور ، کاربر وارد سیستم میشود و با دریافت Token ، بازی ، امکان ارسال فرامینی که به Token نیاز دارند را به دست میآورد .

Token جدید از طریق UserData.token قابل دستیابی خواهد بود

```
Abring.Instance.Login(
string username,
 string password,
string app_id,
serverresponse success,
serverresponse fail,
serverresponse noconnection,
bool showdebug)
```

در پایان اجرای این دستور و اگر این دستور موفقیت آمیز باشد :

Userdata.token حاوی توکن دریافتی از سرور است مثلا ممکن است حاوی رشته ای به شکل 0b8d9d91a6c5d7807e980108999c31b4 باشد

UserData.name :نام پلیر

UserData.app_id :نام اپلیکیشن

سرویس loginWithDeviceID

توضیح : ثبت نام یا ورود از طریق DeviceID

اگر قبلا ثبت نام نکرده باشد ، ثبت نام میکند و توکن برمیگرداند ولی اگر ثبت نام شده باشد فقط توکن جدید میدهد

```
void loginWithDeviceID(string app_id, string deviceID, string guest_login_is_enabled, string name, serverresponse success, serverresponse fail, serverresponse noconnection, bool showdebug)
```

پاسخ نمونه از سمت سرور :

//on successful

{"code":"200","result":"8f08fe4526cb0b325439b7413a96e394","message":"Action executed successfully!","debug":{"execution_time":"354 ms"}}

//on fail

{"code":"400","result":"","message":"login failed","debug":{"execution_time":"71 ms"}}

کد نمونه :

```
Abring.Instance.loginWithDeviceID(UserData.app_id,
SystemInfo.deviceUniqueIdentifier, "1", “name”,
/*success*/(string s) =>
{Abring.Instance.Analytic.Add(“NewInstallation”, false); },

/*fail*/(string s) =>{},

/*nonetwork*/(string s) =>{
Abring.Instance.Analytic.Add(“InstallationFailed”,false);},
false);
```

سرویس logout

توضیح : خروج کاربر از سیستم .

با اجرای این دستور ، کاربر از سیستم خارج شده و Token دریافت شده غیر معتبر خواهد بود

```
Abring.Instance.Logout(success, fail, noconnectregister, false);
```

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا Token ، با پیغام خطا مواجه میشود و رویداد OnLogoutFail فراخوانی میشود

در صورتی عملیات ورود به سیستم موفقیت آمیز باشد ، رویداد OnLogoutSuccessFull فراخوانی میشود .

```
Abring.Instance.Logout(token, app_id, false);
```

رویدادها :

OnLogoutSuccessFull

در صورتی که خروج از سیستم با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnLogoutFail

درصورتی که نام بازی یا Token وارد شده نامعتبر باشد این رویداد فراخوانی میشود

OnLogoutConnecttoserverFail

در صورتی که در مراحل خروج از سیستم ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

```
//on successful

{"code":"200","result":"Done","message":"Action executed
successfully!","debug":{"execution_time":"39 ms"}}

//on fail
```

کد نمونه :

```
public void logout()
    {
        Debug.Log("logout");
        Abring.Instance.OnLogoutSuccessFull += success;
        Abring.Instance.OnLogoutFail += fail;
        Abring.Instance.OnLogoutConnecttoserverFail += noconnectAction;
        Abring.Instance.Logout(token, app_id, false);

    }

    void success(string str)
    {
        Debug.Log("success=" + str);
    }

    void fail(string str)
    {
        Debug.Log("fail=" + str);
    }

    void noconnect(string str)
    {
        Debug.Log("noconnect=" + str);
    }

    void noconnectAction(string str)
    {
```

```
    Debug.Log("noconnectregister=" + str);
}
```

سرویس playerget

توضیح : درخواست اطلاعاتی که در سرور به عنوان اطلاعات بازیکن ذخیره شده.

```
PlayerGet(
string player_id,
bool showdebug,
string[] variables,
serverresponse success, serverresponse fail, serverresponse connectionfail);
```

درصورتی که player_id خالی باشد ، اطلاعات پلیر فعلی نمایش داده میشود وگرنه اطلاعات عمومی پلیر معرفی شده نمایش داده میشود

Variables شامل یک آرایه از متغییر هایی است که از سرور درخواست میشود. اگر خالی باشد ، همه متغییر ها بازگشت داده میشوند

سرویس playergetglobal

توضیح : درخواست اطلاعاتی که در سرور به عنوان اطلاعات بازیکن به شکل عمومی ذخیره شده.

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا Token ، با پیغام خطا مواجه میشود و رویداد OnPlayerGetFail فراخوانی میشود

در صورتی عملیات دریافت اطلاعات پلیر موفقیت آمیز باشد ، رویداد OnPlayerGetSuccessFull فراخوانی میشود .

Abring.Instance. PlayerGetglobal (token, app_id, false);

رویدادها :

OnLogoutSuccessFull

در صورتی که خروج از سیستم با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnLogoutFail

درصورتی که نام بازی یا Token وارد شده نامعتبر باشد این رویداد فراخوانی میشود

OnLogoutConnecttoserverFail

در صورتی که در مراحل خروج از سیستم ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

{"code":"200","result":"Done","message":"Action executed successfully!","debug":{"execution_time":"39 ms"}}

//on fail

کد نمونه :

```
public void PlayerGetglobal ()
   {
      Debug.Log("PlayerGetglobal");
      Abring.Instance.OnLogoutSuccessFull += success;
      Abring.Instance.OnLogoutFail += fail;
      Abring.Instance.OnLogoutConnecttoserverFail += noconnectAction;
      Abring.Instance. PlayerGetglobal (token, app_id, false);

   }

   void success(string str)
   {
      Debug.Log("success=" + str);
   }

   void fail(string str)
   {
      Debug.Log("fail=" + str);
   }

   void noconnect(string str)
```

```
{
    Debug.Log("noconnect=" + str);
}
```

سرویس playerset

توضیح : ثبت اطلاعات جدید یا آپدیت اطلاعاتی که در سرور به عنوان اطلاعات بازیکن ذخیره شده.

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا Token و یا وجود تفاوت در تعداد متغیر ها و مقادیر، با پیغام خطا مواجه میشود و رویداد OnPlayerSetFail فراخوانی میشود

در صورتی عملیات دریافت اطلاعات پلیر موفقیت آمیز باشد ، رویداد OnPlayerSetSuccessFull فراخوانی میشود .

```
Abring.Instance.PlayerSet(token, app_id, vars[], vals[], false);
```

رویدادها :

OnPlayerSetSuccessFull

در صورتی که ثبت ویا اپدیت متغیر ها با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnPlayerSetFail

درصورتی که به دلیل ارسال اشتباه نام بازی یا Token و یا وجود تفاوت در تعداد متغیر ها و مقادیر وارد شده اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnPlayerSetConnecttoserverFail

در صورتی که در مراحل ثبت اطلاعات ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

{"code":"200","result":"Done","message":"Action executed successfully!","debug":{"execution_time":"136 ms"}}

//on fail

کد نمونه :

```
public void playerset()
 {
    Debug.Log("playerset");
    Abring.Instance.OnPlayerSetSuccessFull += success;
    Abring.Instance.OnPlayerSetFail += fail;
    Abring.Instance.OnPlayerSetConnecttoserverFail += noconnectAction;
    string[] vars = { "vars1", "vars2" };
    string[] vals = { "20", "40" };
    Abring.Instance.PlayerSet(token, app_id, vars, vals, false);
 }
 void success(string str)
 {
    Debug.Log("success=" + str);
 }

 void fail(string str)
 {
    Debug.Log("fail=" + str);
 }
```

```
void noconnect(string str)
{
    Debug.Log("noconnect=" + str);
}

void noconnectAction(string str)
{
    Debug.Log("noconnectregister=" + str);
}
```

سرویس playersetImages

توضیح : ثبت یک تصویر مانند آواتار یا اپدیت تصاویری که در سرور به عنوان اطلاعات بازیکن ذخیره شده.

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا Token و یا اشکال در ارسال فایل تصویر ، با پیغام خطا مواجه میشود و رویداد OnPlayerSetFail فراخوانی میشود

در صورتی عملیات دریافت اطلاعات پلیر موفقیت آمیز باشد ، رویداد OnPlayerSetSuccessFull فراخوانی میشود .

```
Abring.Instance.PlayerSet(token, app_id, vars[], vals[], false);
```

رویدادها :

OnPlayerSetSuccessFull

در صورتی که ثبت ویا اپدیت متغیر ها با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnPlayerSetFail

درصورتی که به دلیل ارسال اشتباه نام بازی یا Token و یا وجود تفاوت در تعداد متغیر ها و مقادیر وارد شده اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnPlayerSetConnecttoserverFail

در صورتی که در مراحل ثبت اطلاعات ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

{"code":"200","result":"Done","message":"Action executed successfully!","debug":{"execution_time":"136 ms"}}

//on fail

کد نمونه :

```
public void playerset()
{
    Debug.Log("playerset");
    Abring.Instance.OnPlayerSetSuccessFull += success;
    Abring.Instance.OnPlayerSetFail += fail;
    Abring.Instance.OnPlayerSetConnecttoserverFail += noconnectAction;
    string[] vars = { "vars1", "vars2" };
    string[] vals = { "20", "40" };
    Abring.Instance.PlayerSet(token, app_id, vars, vals, false);
}
void success(string str)
{
    Debug.Log("success=" + str);
}

void fail(string str)
{
    Debug.Log("fail=" + str);
}
```

```
void noconnect(string str)
{
    Debug.Log("noconnect=" + str);
}

void noconnectAction(string str)
{
    Debug.Log("noconnectregister=" + str);
}
```

سرویس playersetglobal

توضیح : ثبت اطلاعات جدید یا آپدیت اطلاعاتی که در سرور به عنوان اطلاعات بازیکن ذخیره شده و در اختیار دوستان یا بقیه اپ ها قرار میگیرد

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا Token و یا وجود تفاوت در تعداد متغیر ها و مقادیر، با پیغام خطا مواجه میشود و رویداد OnPlayerSetFail فراخوانی میشود

در صورتی عملیات دریافت اطلاعات پلیر موفقیت آمیز باشد ، رویداد OnPlayerSetSuccessFull فراخوانی میشود .

Abring.Instance. PlayerSetGlobal (token, app_id, vars[], vals[], false);

رویدادها :

OnPlayerSetSuccessFull

در صورتی که ثبت ویا اپدیت متغیر ها با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnPlayerSetFail

درصورتی که به دلیل ارسال اشتباه نام بازی یا Token و یا وجود تفاوت در تعداد متغیر ها و مقادیر وارد شده اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnPlayerSetConnecttoserverFail

در صورتی که در مراحل ثبت اطلاعات ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

{"code":"200","result":"Done","message":"Action executed successfully!","debug":{"execution_time":"136 ms"}}

//on fail

کد نمونه :

```
public void playerset()
    {
        Debug.Log("PlayerSetGlobal ");
        Abring.Instance.OnPlayerSetSuccessFull += success;
        Abring.Instance.OnPlayerSetFail += fail;
        Abring.Instance.OnPlayerSetConnecttoserverFail += noconnectAction;
        string[] vars = { "vars1", "vars2" };
        string[] vals = { "20", "40" };
        Abring.Instance. PlayerSetGlobal (token, app_id, vars, vals, false);
    }
    void success(string str)
    {
        Debug.Log("success=" + str);
    }

    void fail(string str)
    {
        Debug.Log("fail=" + str);
```

```
}

void noconnect(string str)
{
   Debug.Log("noconnect=" + str);
}

void noconnectAction(string str)
{
   Debug.Log("noconnectregister=" + str);
}
```

سرویس playersetImagesglobal

توضیح : ثبت یک تصویر مانند آواتار یا اپدیت تصاویری که در سرور به عنوان اطلاعات بازیکن ذخیره شده و و در اختیار دوستان یا بقیه اپ ها قرار میگیرد

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا Token و یا اشکال در ارسال فایل تصویر ، با پیغام خطا مواجه میشود و رویداد OnPlayerSetFail فراخوانی میشود

در صورتی عملیات دریافت اطلاعات پلیر موفقیت آمیز باشد ، رویداد OnPlayerSetSuccessFull فراخوانی میشود .

Abring.Instance. PlayerSetImageGlobal (token, app_id, vars[], vals[], false);

رویدادها :

OnPlayerSetSuccessFull

در صورتی که ثبت ویا اپدیت متغیر ها با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnPlayerSetFail

درصورتی که به دلیل ارسال اشتباه نام بازی یا Token و یا وجود تفاوت در تعداد متغیر ها و مقادیر وارد شده اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnPlayerSetConnecttoserverFail

در صورتی که در مراحل ثبت اطلاعات ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

{"code":"200","result":"Done","message":"Action executed successfully!","debug":{"execution_time":"136 ms"}}

//on fail

کد نمونه :

```
public void PlayerSetImageGlobal ()
{
    Debug.Log("PlayerSetImageGlobal ");
    Abring.Instance.OnPlayerSetSuccessFull += success;
    Abring.Instance.OnPlayerSetFail += fail;
    Abring.Instance.OnPlayerSetConnecttoserverFail += noconnectAction;
    string[] vars = { "vars1", "vars2" };
    string[] vals = { "20", "40" };
    Abring.Instance. PlayerSetImageGlobal (token, app_id, vars, vals, false);
}
void success(string str)
{
    Debug.Log("success=" + str);
}

void fail(string str)
{
    Debug.Log("fail=" + str);
}

void noconnect(string str)
{
    Debug.Log("noconnect=" + str);
}

void noconnectAction(string str)
```

```
{
    Debug.Log("noconnectregister=" + str);
}
```

سرویس playerget

توضیح : دریافت اطلاعاتی که در سرور به عنوان اطلاعات بازیکن ذخیره شده.

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا Token ، با پیغام خطا مواجه میشود و رویداد OnPlayerGetFailفراخوانی میشود

در صورتی عملیات دریافت اطلاعات پلیر موفقیت آمیز باشد ، رویداد OnPlayerGetSuccessFullفراخوانی میشود .

```
Abring.Instance.PlayerGet(token, app_id, false);
```

رویدادها :

OnPlayerGetSuccessFull

در صورتی که ثبت ویا اپدیت متغیر ها با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnPlayerGetFail

درصورتی که به دلیل ارسال اشتباه نام بازی یا Token و یا وجود تفاوت در تعداد متغیر ها و مقادیر وارد شده اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnPlayerGetConnecttoserverFail

در صورتی که در مراحل ثبت اطلاعات ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

```
//on successful
```

{"code":"200","result":{"leaderboard_profile":"{\"name\": \"masood16\"}","profile":"{ \"title\": \"test\" , \"name\": \"masood16\"}","create_time":"2016-10-05 12:29:27","vars1":"20","vars2":"40","username":"masood16"},"message":"Action executed successfully!","debug":{"execution_time":"114 ms"}}

//on fail

کد نمونه :

```
public void playerget()
{
    Debug.Log("playerget");
    Abring.Instance.OnPlayerGetSuccessFull += success;
    Abring.Instance.OnPlayerGetFail += fail;
    Abring.Instance.OnPlayerGetConnecttoserverFail += noconnectAction;
    Abring.Instance.PlayerGet(token, app_id, false);
}

void success(string str)
{
    Debug.Log("success=" + str);
}

void fail(string str)
{
    Debug.Log("fail=" + str);
}
```

```
void noconnect(string str)
{
    Debug.Log("noconnect=" + str);
}

void noconnectAction(string str)
{
    Debug.Log("noconnectregister=" + str);
}
```

سرویس AppGet

توضیح : دریافت اطلاعاتی که در سرور به عنوان اطلاعات بازی ذخیره شده.

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا متغیر ، با پیغام خطا مواجه میشود و رویداد OnAppGetFail میشود

در صورتی عملیات دریافت اطلاعات پلیر موفقیت آمیز باشد ، رویداد OnAppGetSuccessFullفراخوانی میشود .

```
Abring.Instance.AppGet(app_id, varname , false);
```

رویدادها :

OnAppGetSuccessFull

در صورتی که دریافت متغیرها با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnAppGetFail

درصورتی که به دلیل ارسال اشتباه نام بازی اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnAppGetConnecttoserverFail

در صورتی که در مراحل دریافت اطلاعات ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

```
//on successful

{"code":"200","result":"http:\/\/shared.asemangame.dev\/img\/applications\/test\/icon.png
","message":"Action executed successfully!","debug":{"execution_time":"50 ms"}}
```

//on fail

کد نمونه :

```
public void AppGet()
{
    Debug.Log("AppGet");
    Abring.Instance.OnAppGetSuccessFull += success;
    Abring.Instance.OnAppGetFail += fail;
    Abring.Instance.OnAppGetConnecttoserverFail += noconnectAction;
    Abring.Instance.AppGet(app_id, "icon", false);
}
void success(string str)
{
    Debug.Log("success=" + str);
}

void fail(string str)
{
    Debug.Log("fail=" + str);
}

void noconnect(string str)
{
    Debug.Log("noconnect=" + str);
}

void noconnectAction(string str)
```

```
{
    Debug.Log("noconnectregister=" + str);
}
```

سرویس AppGetAll

توضیح : دریافت همه اطلاعاتی که در سرور به عنوان اطلاعات بازی ذخیره شده. درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی ، با پیغام خطا مواجه میشود و رویداد OnAppGetAllFail میشود

در صورتی عملیات دریافت اطلاعات پلیر موفقیت آمیز باشد ، رویداد OnAppGetAllConnecttoserverFail فراخوانی میشود .

```
Abring.Instance.AppGetAll(app_id, false);
```

رویدادها :

OnAppGetAllSuccessFull

در صورتی که دریافت همه متغیرها با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnAppGetAllFail

درصورتی که به دلیل ارسال اشتباه نام بازی اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnAppGetAllConnecttoserverFail

در صورتی که در مراحل دریافت اطلاعات ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

```
//on successful
```

{"code":"200","result":{"name":" اپلیکیشن
تست","icon":"http:\/\/shared.asemangame.dev\/img\/applications\/test\/icon.png"},"mes
sage":"Action executed successfully!","debug":{"execution_time":"27 ms"}}

//on fail

{"code":"200","result":[],"message":"Action executed
successfully!","debug":{"execution_time":"35 ms"}}

کد نمونه :

```
public void AppGetAll()
{
    Debug.Log("AppGetAll");
    Abring.Instance.OnAppGetAllSuccessFull += success;
    Abring.Instance.OnAppGetAllFail += fail;
    Abring.Instance.OnAppGetAllConnecttoserverFail += noconnectAction;
    Abring.Instance.AppGetAll(app_id, false);
}

void success(string str)
{
    Debug.Log("success=" + str);
}

void fail(string str)
{
```

```
    Debug.Log("fail=" + str);
}

void noconnect(string str)
{
    Debug.Log("noconnect=" + str);
}

void noconnectAction(string str)
{
    Debug.Log("noconnectregister=" + str);
}
```

سرویس LeaderboardGet

توضیح : دریافت اطلاعات لیدر بورد موردنظر که در سرور ذخیره شده.

```
last_score;
Most_score;
rate;
```

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا نام لیدربرد ، با پیغام خطا مواجه میشود و رویداد OnLeaderboardGetFailمیشود

در صورتی که عملیات دریافت اطلاعات لیدربرد ، موفقیت آمیز باشد ، رویداد OnLeaderboardGetSuccessFull فراخوانی میشود .

Abring.Instance.LeaderboardGet(app_id, "default leaderboard", token, false);

رویدادها :

OnLeaderboardGetSuccessFull

در صورتی که دریافت لیدربرد با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnLeaderboardGetFail

درصورتی که به دلیل ارسال اشتباه نام بازی یا نام لیدر برد و یا Tokne ، اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnLeaderboardGetConnecttoserverFail

در صورتی که در مراحل دریافت اطلاعات ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

{"code":"200","result":{"title":"default leaderboard","max_show":3,"top_scores":[{"player_id":"kia447617267","leaderboard_profile":"{\"name\":\"no name\",\"avatar\":\"no avatar\"}","score":"170"}],"update":1475332489,"rate":1},"message":"Action executed successfully!","debug":{"execution_time":"92 ms"}}

//on fail

{"code":"400","result":"","message":"Leaderboard not found","debug":{"execution_time":"42 ms"}}

کد نمونه :

```
public void LeaderboardGet()
    {
        Debug.Log("LeaderboardGet");
        Abring.Instance.OnLeaderboardGetSuccessFull += success;
        Abring.Instance.OnLeaderboardGetFail += fail;
        Abring.Instance.OnLeaderboardGetConnecttoserverFail += noconnectAction;
        Abring.Instance.LeaderboardGet(app_id, "default leaderboard", token, false);
    }

    void success(string str)
    {
        Debug.Log("success=" + str);
    }
```

```
void fail(string str)
{
    Debug.Log("fail=" + str);
}

void noconnect(string str)
{
    Debug.Log("noconnect=" + str);
}

void noconnectAction(string str)
{
    Debug.Log("noconnectregister=" + str);
}
```

سرویس LeaderboardGetAll

توضیح : دریافت اطلاعات همه لیدر بورد ها که در سرور ذخیره شده.

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا Token، با پیغام خطا مواجه میشود و رویداد OnLeaderboardGetAllFail میشود

در صورتی که عملیات دریافت اطلاعات لیدربرد ، موفقیت آمیز باشد ، رویداد OnLeaderboardGetAllSuccessFull فراخوانی میشود .

```
Abring.Instance.LeaderboardGetAll(app_id, token, false);
```

رویدادها :

OnLeaderboardGetAllSuccessFull

در صورتی که دریافت لیدربرد با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnLeaderboardGetAllFail

درصورتی که به دلیل ارسال اشتباه نام بازی یا Tokne ، اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnLeaderboardGetAllConnecttoserverFail

در صورتی که در مراحل دریافت اطلاعات ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

```
//on successful
```

{"code":"200","result":{"leaderboard_59":{"title":"leaderboard","max_show":20,"":"","rate":-1, "top_scores": [{"player_id":"masood14","leaderboard_profile":"{\"name\":\"masood14\"}","score":"20"}],"update":1475667249},"default":{"title":"default leaderboard","max_show":3,"top_scores":[{"player_id":"kia447617267","leaderboard_profile":"{\"name\":\"no name\",\"avatar\":\"no avatar\"}","score":"170"}],"update":1475332489,"rate":1}},"message":"Action executed successfully!","debug":{"execution_time":"19 ms"}}

//on fail

{"code":"400","result":"","message":"Leaderboard not found","debug":{"execution_time":"40 ms"}}

کد نمونه :

```
public void LeaderboardGetAll()
{
    Debug.Log("LeaderboardGetAll");
    Abring.Instance.OnLeaderboardGetAllSuccessFull += success;
    Abring.Instance.OnLeaderboardGetAllFail += fail;
    Abring.Instance.OnLeaderboardGetAllConnecttoserverFail += noconnectAction;
    Abring.Instance.LeaderboardGetAll(app_id, token, false);
}
void success(string str)
{
    Debug.Log("success=" + str);
}

void fail(string str)
```

```
{
    Debug.Log("fail=" + str);
}

void noconnect(string str)
{
    Debug.Log("noconnect=" + str);
}

void noconnectAction(string str)
{
    Debug.Log("noconnectregister=" + str);
}
```

سرویس LeaderboardSetScore

توضیح : ثبت امتیازبازیکن در لیدربرد مربوطه.

درصورت موفقیت آمیز نبودن اجرای دستور به دلیل ارسال اشتباه نام بازی یا Token و یا نام لیدربرد و یا Signature ، با پیغام خطا مواجه میشود و رویداد OnLeaderboardSetScoreGetFail میشود

در صورتی که عملیات دریافت اطلاعات لیدربرد ، موفقیت آمیز باشد ، رویداد OnLeaderboardSetScoreSuccessFull فراخوانی میشود .

مساله مهم در این قسمت وجود signature است که برای جلوگیری از هک لیدربرد ساخته شده و به کمک فرمولی که سازنده اپ ، در پانل لیدر برد ، ثبت کرده ، ساخته میشود و هربار باید همراه امتیاز ، ارسال شود تا معلوم شود امتیاز توس بازی به دست آمده و یا در اثر هک ، ثبت شده است و به عنوان مثال در صفحه لیدربرد در پنل بازی ، در قسمت signature به عنوان نمونه نوشته شده است :

```
$signature=$score-1;
```

و این به این معنی است که هربار که امتیاز ثبت میشود باید یک واحد کمتر از امتیاز هم به عنوان signature ارسالی شود یعنی اگر امتیاز20 است باید signature با مقدار 19 هم ارسال شود .

بدیهی است این ساده ترین فرمول است و امکان پیچیده سازی این فرمول به کمک توابعی مانند md5 , sin , cos و غیره در پنل موجود است .

```
Abring.Instance.LeaderboardSetScore(token, app_id, "leaderboard_59", "20", "19", false);
```

رویدادها :

OnLeaderboardSetScoreSuccessFull

در صورتی که ثبت امتیاز با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnLeaderboardSetScoreGetFail

درصورتی که در مراحل ثبت امتیاز ، اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnLeaderboardSetScoreConnecttoserverFail

در صورتی که در مراحل ثبت اطلاعات ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

{"code":"200","result":{"top_scores":[{"player_id":"masood16","leaderboard_profile":"{\"name\":\"masood16\"}","score":"20"},{"player_id":"masood14","leaderboard_profile":"{\"name\": \"masood14\"}","score":"20"}],"rate":1},"message":"Action executed successfully!","debug":{"execution_time":"398 ms"}}

//on fail

{"code":"400","result":"","message":"Invalid signature!\n default signature will results 3416a75f4cea9109507cacd8e2f2aefc","debug":{"execution_time":"20 ms"}}

کد نمونه :

```
public void LeaderboardSetScore()
{
    Debug.Log("LeaderboardSetScore");
    Abring.Instance.OnLeaderboardSetScoreSuccessFull += success;
    Abring.Instance.OnLeaderboardSetScoreGetFail += fail;
```

```
        Abring.Instance.OnLeaderboardSetScoreConnecttoserverFail += noconnectAction;
        Abring.Instance.LeaderboardSetScore(token, app_id, "leaderboard_59", "20", "19",
false);
    }
    void success(string str)
    {
        Debug.Log("success=" + str);
    }

    void fail(string str)
    {
        Debug.Log("fail=" + str);
    }

    void noconnect(string str)
    {
        Debug.Log("noconnect=" + str);
    }

    void noconnectAction(string str)
    {
        Debug.Log("noconnectregister=" + str);
    }
```

سرویس FriendsList

توضیح : دریافت لیست دوستان بازیکن

درصورت موفقیت آمیز نبودن اجرای دستور ، با پیغام خطا مواجه میشود و رویداد OnFriendslistFail فراخوانی میشود

در صورتی که عملیات دریافت اطلاعات لیدربرد ، موفقیت آمیز باشد ، رویداد OnFriendslistSuccessFull فراخوانی میشود .

Abring.Instance.Friends.FriendsList(Type , offset,limit,showdebug);

رویدادها :

OnFriendslistSuccessFull

در صورتی که دریافت لیست دوستان با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnFriendslistFail

درصورتی که در مراحل دریافت لیست دوستان ، اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnFriendslistConnecttoserverFail

در صورتی که در مراحل دریافت لیست دوستان ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

```
//on successful


//on fail
```

کد نمونه :

```
public void Friendslist()
    {
        Debug.Log("Friendslist");
        Abring.Instance.Friends.OnFriendslistSuccessFull += success;
        Abring.Instance.Friends.OnFriendslistFail += fail;
        Abring.Instance.Friends.OnFriendslistConnecttoserverFail +=
noconnectNoConnectFriendsList;
        Abring.Instance.Friends.FriendsList("", "", "", false);
    }


    void success(string str)
    {
        Debug.Log("success=" + str);
    }

    void fail(string str)
    {
```

```
    Debug.Log("fail=" + str);
}

void noconnectNoConnectFriendsList(string str)
{
    Debug.Log("noconnectregister=" + str);
                                                                    }
```

سرویس FriendsSearch

توضیح : جستجوی یک بازیکن

درصورت موفقیت آمیز نبودن اجرای دستور ، با پیغام خطا مواجه میشود و رویداد OnFriendsearchFail فراخوانی میشود

در صورتی که عملیات دریافت اطلاعات ، موفقیت آمیز باشد ، رویداد OnFriendsearchSuccessFullفراخوانی میشود .

Gamehub.Instance.Friends.FriendsSearch(Pattern , limit, showdebug);

رویدادها :

OnFriendsearchSuccessFull

در صورتی که دریافت لیست دوستان با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnFriendsearchFail

درصورتی که در مراحل دریافت لیست دوستان ، اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnFriendsearchConnecttoserverFail

در صورتی که در مراحل دریافت لیست دوستان ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

{"code":"200","result":[{"app":"test","player_id":"teh","name":"No name","avatar":"No avatar","leaderboard_profile":"{\"name\":\"teh\"}"}],"message":"Action executed successfully!","debug":{"execution_time":"16 ms"}}

//on fail

کد نمونه :

```
public void FriendsSearch()
{
    Debug.Log("FriendsSearch");
    Abring.Instance.Friends.OnFriendsearchSuccessFull += success;
    Abring.Instance.Friends.OnFriendsearchFail += fail;
    Abring.Instance.Friends.OnFriendsearchConnecttoserverFail += noconnectNoConnect;
    Abring.Instance.Friends.FriendsSearch("*", "5", false);
}

void success(string str)
{
    Debug.Log("success=" + str);
}

void fail(string str)
{
    Debug.Log("fail=" + str);
```

```
}

void noconnectNoConnect(string str)
{
    Debug.Log("noconnectregister=" + str);
                                                                                }
```

سرویس friendsInvite

توضیح : دعوت یک بازیکن به دوستی

درصورت موفقیت آمیز نبودن اجرای دستور ، با پیغام خطا مواجه میشود و رویداد OnFriendsinviteFail فراخوانی میشود

در صورتی که عملیات دریافت اطلاعات ، موفقیت آمیز باشد ، رویداد OnFriendsinviteSuccessFull فراخوانی میشود .

Abring.Instance.Friends.Friendsinvite(other player id, showdebug);

رویدادها :

OnFriendsinviteSuccessFull

در صورتی که دریافت لیست دوستان با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnFriendsinviteFail

درصورتی که در مراحل دریافت لیست دوستان ، اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnFriendsinviteConnecttoserverFail

در صورتی که در مراحل دریافت لیست دوستان ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

//on fail

کد نمونه :

```
    public void friendsInvite()
    {
        Debug.Log("friendsInvite");
        Abring.Instance.Friends.OnFriendsinviteSuccessFull += success;
        Abring.Instance.Friends.OnFriendsinviteFail += fail;
        Abring.Instance.Friends.OnFriendsinviteConnecttoserverFail +=
noconnection;
        Abring.Instance.Friends.Friendsinvite(pleaseinvite, true);
    }
    void success(string str)
    {
        Debug.Log("success=" + str);
    }

    void fail(string str)
    {
        Debug.Log("fail=" + str);
    }

    void noconnection(string str)
    {
        Debug.Log("noconnectregister=" + str);
    }
```

سرویس friendsaccept

توضیح : قبول درخواست دوستی یک بازیکن

درصورت موفقیت آمیز نبودن اجرای دستور ، با پیغام خطا مواجه میشود و رویداد OnFriendsAcceptFail فراخوانی میشود

در صورتی که عملیات دریافت اطلاعات ، موفقیت آمیز باشد ، رویداد OnFriendsAcceptSuccessFull فراخوانی میشود .

Abring.Instance.Friends.FriendsAccept(other player id,showdebug);

رویدادها :

OnFriendsAcceptSuccessFull

در صورتی که دریافت لیست دوستان با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnFriendsAcceptFail

درصورتی که در مراحل دریافت لیست دوستان ، اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnFriendsAcceptConnecttoserverFail

در صورتی که در مراحل دریافت لیست دوستان ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

//on fail

کد نمونه :

```
    public void friendsaccept()
    {
        Debug.Log("friendsaccept");
        Abring.Instance.Friends.OnFriendsAcceptSuccessFull += success;
        Abring.Instance.Friends.OnFriendsAcceptFail += fail;
        Abring.Instance.Friends.OnFriendsAcceptConnecttoserverFail +=
noconnectregister;
        Abring.Instance.Friends.FriendsAccept(pleaseinvite, true);
    }
    void success(string str)
    {
        Debug.Log("success=" + str);
    }

    void fail(string str)
    {
        Debug.Log("fail=" + str);
    }

    void noconnection(string str)
    {
        Debug.Log("noconnectregister=" + str);
    }
```

سرویس friendsrject

توضیح : رد کردن درخواست دوستی یک بازیکن

درصورت موفقیت آمیز نبودن اجرای دستور ، با پیغام خطا مواجه میشود و رویداد OnFriendsAcceptFail فراخوانی میشود

در صورتی که عملیات دریافت اطلاعات ، موفقیت آمیز باشد ، رویداد OnFriendsAcceptSuccessFull فراخوانی میشود .

Abring.Instance.Friends.FriendsAccept(other player id,showdebug);

رویدادها :

OnFriendsAcceptSuccessFull

در صورتی که دریافت لیست دوستان با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnFriendsAcceptFail

درصورتی که در مراحل دریافت لیست دوستان ، اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnFriendsAcceptConnecttoserverFail

در صورتی که در مراحل دریافت لیست دوستان ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

//on successful

//on fail

کد نمونه :

```
    public void friendsaccept()
  {
    Debug.Log("friendsaccept");
    Abring.Instance.Friends.OnFriendsAcceptSuccessFull += success;
    Abring.Instance.Friends.OnFriendsAcceptFail += fail;
    Abring.Instance.Friends.OnFriendsAcceptConnecttoserverFail +=
noconnection;
    Abring.Instance.Friends.FriendsAccept(pleaseinvite, true);
  }
  void success(string str)
  {
    Debug.Log("success=" + str);
  }

  void fail(string str)
  {
    Debug.Log("fail=" + str);
  }

  void noconnection(string str)
  {
    Debug.Log("noconnectregister=" + str);
  }
```

سرویس friendsCancellRequest

توضیح : لغو درخواست دوستی با یک بازیکن

درصورت موفقیت آمیز نبودن اجرای دستور ، با پیغام خطا مواجه میشود و رویداد OnFriendsCancelRequestFail فراخوانی میشود

در صورتی که عملیات دریافت اطلاعات ، موفقیت آمیز باشد ، رویداد OnFriendsCancelRequestSuccessFull فراخوانی میشود .

```
Abring.Instance.Friends.FriendsCancelRequest(other player id, true);
```

رویدادها :

OnFriendsCancelRequestSuccessFull

در صورتی که دریافت لیست دوستان با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnFriendsCancelRequestFail

درصورتی که در مراحل دریافت لیست دوستان ، اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnFriendsCancelRequestConnecttoserverFail

در صورتی که در مراحل دریافت لیست دوستان ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

```
//on successful
```

//on fail

کد نمونه :

```
    public void friendsCancellRequest()
    {
        Debug.Log("friendsCancellRequest");
        Abring.Instance.Friends.OnFriendsCancelRequestSuccessFull += success;
        Abring.Instance.Friends.OnFriendsCancelRequestFail += fail;
        Abring.Instance.Friends.OnFriendsCancelRequestConnecttoserverFail +=
noconnectregister;
        Abring.Instance.Friends.FriendsCancelRequest(pleaseinvite, true);
    }
    void success(string str)
    {
        Debug.Log("success=" + str);
    }

    void fail(string str)
    {
        Debug.Log("fail=" + str);
    }

    void noconnection(string str)
    {
        Debug.Log("noconnectregister=" + str);
    }
```

سرویسfriendsUnfriends

توضیح : لغو دوستی با یک بازیکن

درصورت موفقیت آمیز نبودن اجرای دستور ، با پیغام خطا مواجه میشود و رویداد OnFriendsUnfriendFail فراخوانی میشود

در صورتی که عملیات دریافت اطلاعات ، موفقیت آمیز باشد ، رویداد OnFriendsUnfriendSuccessFullفراخوانی میشود .

```
Abring.Instance.Friends.FriendsUnfriends(other player id, true);
```

رویدادها :

OnFriendsUnfriendSuccessFull

در صورتی که دریافت لیست دوستان با موفقیت تکمیل شود این رویداد فراخوانی میشود

OnFriendsUnfriendFail

درصورتی که در مراحل دریافت لیست دوستان ، اجرا با مشکل مواجه شود این رویداد فراخوانی میشود

OnFriendsUnfriendConnecttoserverFail

در صورتی که در مراحل دریافت لیست دوستان ، امکان اتصال به سرور موجود نباشد یا اینترنت قطع شده باشد این رویداد فراخوانی میشود

پاسخ نمونه از سمت سرور :

```
//on successful
```

//on fail

کد نمونه :

```
    public void friendsUnfriends()
    {
        Debug.Log("friendsUnfriends");
        Abring.Instance.Friends.OnFriendsUnfriendSuccessFull += success;
        Abring.Instance.Friends.OnFriendsUnfriendFail += fail;
        Abring.Instance.Friends.OnFriendsUnfriendConnecttoserverFail +=
noconnectregister;
        Abring.Instance.Friends.FriendsUnfriends(pleaseinvite, true);
    }
    void success(string str)
    {
        Debug.Log("success=" + str);
    }

    void fail(string str)
    {
        Debug.Log("fail=" + str);
    }

    void noconnection(string str)
    {
        Debug.Log("noconnectregister=" + str);
    }
```

سرویس ping

توضیح : بررسی سرویس دهنده ابرینگ برای اینکه ایا انلاین است یا نه و بررسی زمان دریافت پاسخ

هدف این سرویس ، فقط این است که متن ارسالی را برگرداند . از کاربرد های آن میتوان به

1- تست انلاین بودن ابرینگ
2- بررسی اتصال به اینترنت
3- بررسی سرعت سرور

روش استفاده :

1- تست اتصال به سرویس دهنده:

```
Abring.Instance.Ping("", success, fail, noconnect, true);
```

در این روش ، فقط اتصال به سرور بررسی میشود و درصورت فراخوانی Success ، به معنی اتصال موفقیت آمیز به سرور پاسخ گویی سرور است.

2- بررسی مدت زمان پاسخ سرور:

```
Abring.Instance.Ping((Time.time * 1000).ToString(),
            (string s) =>
            {
                Debug.Log("success=" + s);
                float deltatime = (Time.time * 1000) -
float.Parse(AbringRespHealper.JsonGetString(s, "result"));
                Debug.Log("deltatime=" + deltatime+" ms");
            }

            , fail, noconnect, true);
```

Analytic

آنالیتیک ابزار بررسی انبوه داده های ارسالی از طرف بازیکنان به توسعه دهنده است و کمک میکنید تا بفهمیم بازیکنان بیشتر چکار میکنند ، کجاها بیشتر می بازند ویا چگونه و برای چه پول خرج میکنند .

روش کار به این شکل است که هر کجا که لازم است میتوانید از دستورات به این شکل استفاده کنید

```
void Abring.Instance.Analytic.Add(string AnalyticVariable, bool showdebug)
```

برای نمونه

```
Abring.Instance.Analytic.Add("test1", true);
```

در این صورت هرگاه که برنامه به این خط از دستورات برسد ، به سرور اطلاع میدهد که در همان لحظه یک به متغیر test1 یک واحد اضافه کند .

روش ارسال نیز به این صورت است که برای کاهش تعداد اتصال به شبکه و کاهش مصرف اینترنت همه ارسال ها در یک صف ذخیره شده و هر 60 ثانیه و یا اگر تعداد آنها به بیش از 20 عدد برسد به صورت خودکار اقدام به ارسال داده ها به سرور میکند.

بر خلاف بیشتر دیگر دستورات SDK ابرینگ ، این دستور میتواند بدون هندل کردن وضعیت های قطعی شبکه و یا خطا استفاده شود زیرا این موضوع در داخل SDK کنترل میشود و در صورت قطع بودن شبکه ، نام متغییر و زمان وقوع را نگه داشته و در هنگام اتصال ، به سرور ارسال میکند.

برای تعامل به شکل پیشرفته تر ، این دستورات نیز وجود دارند:

1- Init بهتر است در ابتدای اجرای برنامه فراخوانی شود (الزامی نیست)

شکل استفاده:

```
void  Abring.Instance.Analytic.init(int Flush_max, int flushAutomatic, bool showDebug)
```

Flush_max : در صورتی که تعداد داده ها در صف از این تعداد بیشتر شود ، اقدام به ارسال داده ها به سرور میکند

flushAutomatic : مدت زمان به ثانیه برای سعی در ارسال داده ها به سرور

showDebug : نمایش اطلاعات اضافی برای زمان دیباگ

2- Addپیشرفته برای ثبت داده ها

```
void Add(string AnalyticVariable, string amount, string date, bool showdebug)
```

AnalyticVariable :متغیری که لازم است اطلاعاتش بر روی سرور ثبت شود

Amount : تعداد افزایش مقدار این متغیر

Date : تاریخ اتفاق افتادن (date format must be 'Y-m-d H')

Showdebug : نمایش اطلاعات اضافی برای زمان دیباگ

3- Flush برای ارسال سریع تمام داده های ثبت شده تا کنون

```
void Flush(serverresponse success, serverresponse fail, serverresponse noconnection, bool showdebug)
```

نحوه استفاده:

```
Abring.Instance.Analytic.Flush(Multiple_Success, Multiple_fail, Multiple_NoConnect, false);


  void Multiple_Success(string s)
  {
     //Analytic flushed suceeessfully
  }
  void Multiple_fail(string s)
  {
     //Analytic flushing with error
  }
  void Multiple_NoConnect(string s)
  {
     //Analytic flushing Nonetwork.add to queue to retry later

  }
```

کلاسهای کمکی:

NetworkSimulation

به کمک این کلاس میتوانید وضعیت شبکه را به صورت ناپایدار شبیه سازی کرده و رفتار برنامه را در چنین وضعیتی بررسی کنید:
بخش فعال یا غیر فعال کردن این کلاس:

bool isEnable()	بررسی فعال یا غیر فعال بودن شبیه ساز
void Enable()	فعال کردن شبیه ساز
void Disable()	غیر فعال کردن شبیه ساز

بخش شبیه سازی تاخیر ارسال یا دریافتها:

void DelayEnable(float delay, int percent)	فعال سازی تاخیر ارسالها Delay : مدت زمان تاخیر بر حسب ثانیه Percent : درصد اعمال این تغییر بر روی پیامهای ارسالی یا دریافتی
void DelayDisable()	غیر فعال سازی تاخیر در ارسال یا دریافتها
float GetSimmulationDelay()	دریافت میزان تاخیر

بخش شبیه سازی عدم ارسال یا دریافت

void DropEnable(int percent)	روشن کردن عدم ارسال یا دریافت درخواستها Percent : درصد عدم ارسال یا دریافت
void DropDisable()	غیر فعال سازی

بخش شبیه سازی قطع بودن اینترنت

void OfflineEnable()	فعال شدن شبیه سازی قطع بودن اینترنت
void OfflineDisable()	غیر فعال شدن شبیه سازی قطع بودن اینترنت

نحوه استفاده :

```
NetworkSimulation.Enable();
NetworkSimulation.OfflineEnable();
NetworkSimulation.DelayEnable(0.5f, 70);
NetworkSimulation.DropEnable(40);
```

Userdata

این ساختار داده ای با هدف نگهداری اطلاعات ارسالی از سمت سرور آماده شده و اطلاعات پرکاربردی مانند ایتم های زیر را برای استفاده آتی نگهداری میکند.
هر کدام از این آیتمها در شرایط خاصی داده ها را دریافت کرده و نگهداری میکنند . SDK بعد از بستن برنامه ، این داده ها را نگهداری نمیکند

item	Description	Filled When
string app_id	نام اپلیکیشن	Register - login
string name	نام بازیکن	Register - login
string avatarURL	ادرس تصویر بازیکن فعلی	در هنگام آپلود عکس
string email	ایمیل بازیکن	Register - login
string tel	شماره تلفن بازیکن در هنگام ثبت نام	Register - login
string rate	رتبه در لیدر برد	دریافت لیدربرد
string token	توکن دریافتی از سرور	Register - login
Texture2D AvatarTexture	تصویر آواتار بازیکن	در هنگام آپلود عکس

LeaderBoard

در هنگامی که لیست لیدر برد را از سرور دریافت میکنیم ، اطلاعات دریافتی از سرور به شکل Json است و البته بسیار بزرگ و شامل دیتا های زیادی است که نوشتن parse آن پرزحمت است . برای این منظور کلاسی تهیه شده که به پاسخ سرور را دریافت کرده و دیتا های لازم را در یک Structure وارد میکند.

برای این منظور یک نمونه از کلاس leaderboard_data_class بسازید و به Constructor آن مقدار برگشتی از LeaderboardGet را ارسال کنید.
کلاس به شکل تعریف شده است :

```
public class leaderboard_data_class
  {
    [System.Serializable]
    public class opponent
    {
      public string score;//امتیاز
      public string leaderboard_profile;
      public string name;//نام بازیکن
      public string rate;//رتبه در لیدربرد
    }
    public List<opponent> dataList;
    public opponent me;

    public leaderboard_data_class(string jsonstr);
  }
```

همانطور که میبینید ، اطلاعات افراد ، در یک لیست به نام dataList ذخیره میشود .
هر کدام ازعناصر این لیست شامل مقادیر نام ، امتیاز و رتبه بازیکن است و
سپس

leaderboardData.dataList[i].rate

رتبه نفر i ام را به ما میدهد
همچنین me ، همین اطلاعات را در مورد خود بازیکن دراختیار قرار میدهد.

نحوه استفاده :
برای این کار در بخش success تابع LeaderboardGet ، مقدار برگشتی از سرور را به سازنده کلاس leaderboard_data_class ارسال میکنیم:

```
Abring.Instance.LeaderboardGet(appname, leaderboardname,
    (string s) =>//successfull
    {
        leaderboardData = new leaderboard_data_class(s);
    },
    (string s) =>//fail
    {},
    (string s) =>//no connection
    {},
    true);
```

.

AbringRespHealper

درحال تهیه:**********

```
try to find the key in json (resault part) and return its float value
static float GetFloat(string json, string key)


try to find the key in json (resault part) and return its string value
static string GetString(string json, string key)

try to find the key in json and return its string value
static string JsonGetString(string json, string key)

return resault part of a json
static string getresault(string json)

update a key with newvalue in a json
static string UpdateKey(string Resaultjson, string key, string newValue)
```

Dialog Component

این کلاس ، رفتار تعدادی پنجره Dialog آماده را هندل میکند.این Dialogها به شکل prefab در پوشه

Abring\Resources

قرار دارند و به نام DialogUI ذخیره شده اند.

نحوه استفاده به این صورت است:

1- نمایش پنجره با دو دکمه ok و Cancel

```
DialogComponent showDialogBoxOkCancel(
string title, // عنوان اصلی
string text, // متن قابل نمایش
Canvas root, // آبجکت روت
dialogEventdelegate Onclick_Ok, //رویداد در هنگام کلیک روی دکمه
dialogEventdelegate Onclick_Cancel)//رویداد در هنگام کلیک روی دکمه
```

2- نمایش پنجره با دکمه ok

```
DialogComponent showDialogBoxOk(
string title, // عنوان اصلی
string text, // متن قابل نمایش
Canvas root, // آبجکت روت
dialogEventdelegate Onclick_Ok, //رویداد در هنگام کلیک روی دکمه
)
```

3- پنهان کردن Dialog Box

```
DialogComponent.HideCurrentDialog();
```

نمونه کد:

```
DialogComponent.showDialogBoxOkCancel(
"You Have To Update Your Game", " There is a better version of your Game", canvas,
() =>{ Debug.Log(updateURL); Application.OpenURL(updateURL); Application.Quit(); }, ok button
() => { Application.Quit(); }//cancell
);
```

```
DialogComponent.showDialogBoxOk("ACTIVATION CODE", "An Activation code Sent for you", canvas,
() => { DialogComponent.HideCurrentDialog(); });
```

Update Checking

برای بررسی اینکه آیا نسخه جدیدی از اپلیکیشن وجود دارد یا نه میتوانید از این کلاس استفاده کنید:

در سرور ابرینگ در قسمت app data یک متغیر با نام update و با مقدار زیر ایجاد کنید:

{"currentversion":"1.10","forceupdateforversionLessthan":"1.04","updateurl":"www","dlcversion":"1"}

و به کمک دستور زیر میتوانید نیاز به آپدیت را بررسی کنید:

```
CheckforUpdate(
Canvas canvasRoot,
string appname,
UpdateCheckResault startoffline,
UpdateCheckResault noneedtoupdate
)
```

در این حالت چند اتفاق ممکن است روی دهد:

Function	STATE
startoffline	اتصال به اینترنت قطع است
startoffline	پلیر انتخاب کرده آفلاین بازی کند
نمایش پنجره برای اجبار به آپدیت	force update
نمایش پنجره برای پیسنهادبه آپدیت	please update And select update
noneedtoupdate	please update And select not to update
noneedtoupdate	no need to update

به عنوان مثال:

```
UpdateCheck.CheckforUpdate(canvas, "3dcar",() => {
Debug.Log("offline"); }, () => { Debug.Log("is update"); });
```

ورود به سیستم (login – register)

در ابتدای اجرای برنامه ، وقتی که برنامه برای اولین بار اجرا میشود ، لازم است که پروسه ورود به ابرینگ برای یوزر طی شود .

برای این منظور ، از یکی از روش های زیر میتوان استفاده کرد :

1- به کمک نام کاربری و پسورد
در این روش ، از یوزر برنامه ، نام کاربری و پسورد را دریافت کرده و به کمک آن نام کاربری و پسورد ، این کاربر در سیستم ثبت میشود . در صورت LogOut ویا Login در یک دیوایس دیگر ، سیستم تشخیص میدهد که همان یوزر قبلی است و همان اطلاعات ذخیره شده را به او برمیگرداند
2- به کمک DeviceID
در این روش با ارسال Device ID به سرور ، عملیات ثبت نام انجام میشود.
3- به کمک شماره تلفن
در این روش ، با ارسال شماره تلفن ، یک کد به آن شماره ارسال میشودو در هنگامی که کد ارسالی به سرور برگردانده شود ، یوزر در سیستم ثبت میشود

Token

عبارت منحصر به فردی است که توسط سرور بعد از عملیات ثبت نام یا ورود کاربر بوجود میاید و به برنامه ارسال میشود . بعد از دریافت Token ، برای مشخص شدن فرد ارسال کننده دستور به سرور ، باید از این token در دستورات استفاده شود. . تا هنگامی که فرایند Logout اتفاق نیافتد ، token معتبر است و قابلیت استفاده دارد . بنابراین با ثبت این رشته در PlayerPref میتوانید در دفعات بعد ورود به بازی ازآن استفاده کنید .

بعد از عملیات ثبت نام در بار اول ، سرور توکن را برگردانده و این عبارت در Userdata.token قرار میگیرد . در هنگام استفاده از Rest API آماده شده توسط سرویس ابرینگ ، تقریبا همیشه باید این توکن را در دستورات خود قرار دهید ولی در هنگام استفاده از SDK ، مقدار توکن از Userdata.token خوانده شده و در دستورات قرار میگیرد و نیازی نیست که دولوپر ، در این مورد نگرانی داشته باشد. ولی چون بعد از هربار بسته شدن اپلیکیشن ، Userdata.token پاک میشود ، بهتر است هربار ، در ابتدای برنامه توسط برنامه از playerpref خوانده شود و در Userdata.token نوشته شود.

بنابراین روند شروع اپلیکیشن ، میتواند به این شکل باشد :

```
string token = PlayerPrefs.GetString("token", "");

if (string.IsNullOrEmpty(token))
  {
   شروع ثبت نام
  }
  Else
  UserData.token = token;
```