小游戏联机对战引擎

SDK 文档

产品文档

腾讯云

文档目录

SDK 文档
前端 SDK
前端 SDK 下载

最近更新时间：2019-12-06 15:29:02

为方便开发者接入小游戏联机对战引擎产品，在下载 SDK 之前，请查阅 更新历史。

平台	更新时间	版本	SDK下载	文档
微信小游戏/QQ 小游戏/百度小游戏/oppo 小游戏 vivo 小游戏/H5 小游戏（JavaScript）	2019/12/6	v1.2.6	下载	快速入门 接口文档

SDK 使用流程

最近更新时间：2019-10-30 16:13:32

操作场景

本文档指导您如何使用小游戏联机对战引擎 SDK。

前提条件

- 已在小游戏联机对战引擎控制台创建小游戏实例，并 开通联机对战服务。
- 已获取游戏 gameID 和 secretKey。SDK 需要对这两个参数进行校验。

操作步骤

设置请求域名

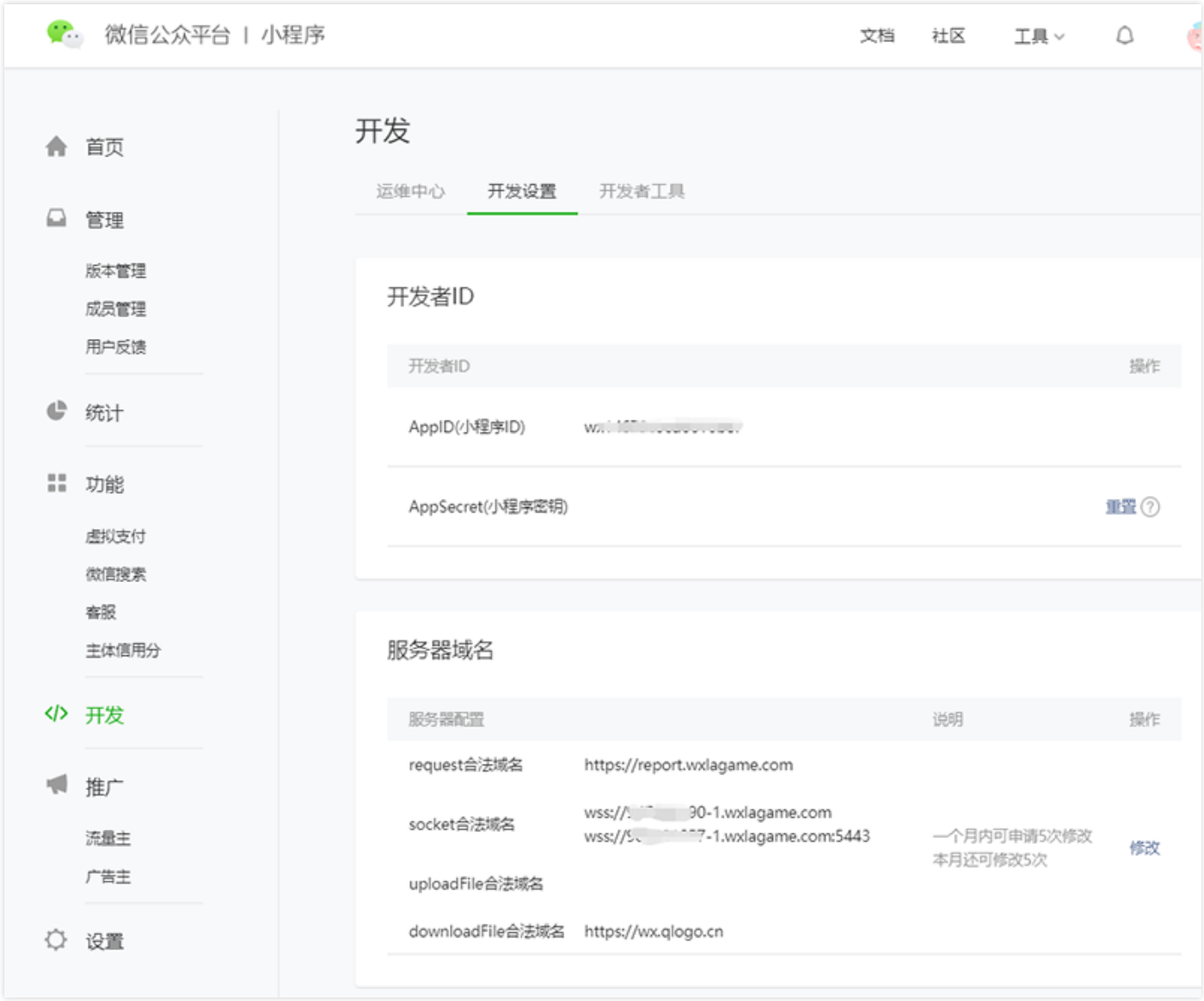
> 注意：
> 出于安全考虑，微信小程序/小游戏会限制请求域名，所有的 HTTPS、WebSocket、上传、下载请求域名都需要在 微信公众平台 进行配置。因此，在正式接入小游戏联机对战引擎 SDK 前，需要开发者在微信公众平台配置合法域名。

1. 需要配置的域名包含两条 socket 域名和一条 request 域名记录。开发者在 MGOBE 控制台上获取域名后，需要配置该域名的默认端口、 5443 端口两条记录。

```
// request 域名
report.wxlagame.com
// socket 域名
xxx.wxlagame.com
xxx.wxlagame.com:5443
```

2. 进入小游戏联机对战引擎控制台，将控制台获取的游戏域名信息复制保存。
3. 登录 微信公众平台，选择左侧菜单栏【开发】>【开发设置】。

4. 进入开发设置详情页，在“服务器域名”中添加合法域名记录。如下图所示：



导入 SDK

SDK 文件包含 MGOBE.js 和 MGOBE.d.ts，即源代码文件和定义文件。在 MGOBE.js 中，SDK 接口被全局注入到 window 对象下。因此，只需要在使用 SDK 接口之前执行 MGOBE.js 文件即可。单击进入 SDK 下载 页面。

微信小游戏原生环境

在微信原生环境中，您只需将 MGOBE.js 放到项目下任意位置，在 game.js 中 import SDK 文件后即可使用 MGOBE 的方法。导入示例代码如下：

```
// 只需要在使用 MGOBE 之前 import 一次该文件
import "./js/libs/MGOBE.js";
// 直接使用 MGOBE
const { Room, Listener, ErrCode, ENUM, DebuggerLog } = MGOBE;
```

界面示例如下图所示：

您也可以使用 import/from、require 语法显式导入 MGOBE 模块。
import/from 代码示例如下：

```
// 使用 import/from
import * as MGOBE from "./js/libs/MGOBE.js";
const { Room, Listener, ErrCode, ENUM, DebuggerLog } = MGOBE;

// 或者
import { Room, Listener, ErrCode, ENUM, DebuggerLog } from "./js/libs/MGOBE.js";
```

require 代码示例如下：

```
// 使用 require
const MGOBE = require("./js/libs/MGOBE.js");
const { Room, Listener, ErrCode, ENUM, DebuggerLog } = MGOBE;

// 或者
const { Room, Listener, ErrCode, ENUM, DebuggerLog } = require("./js/libs/MGOBE.js");
```

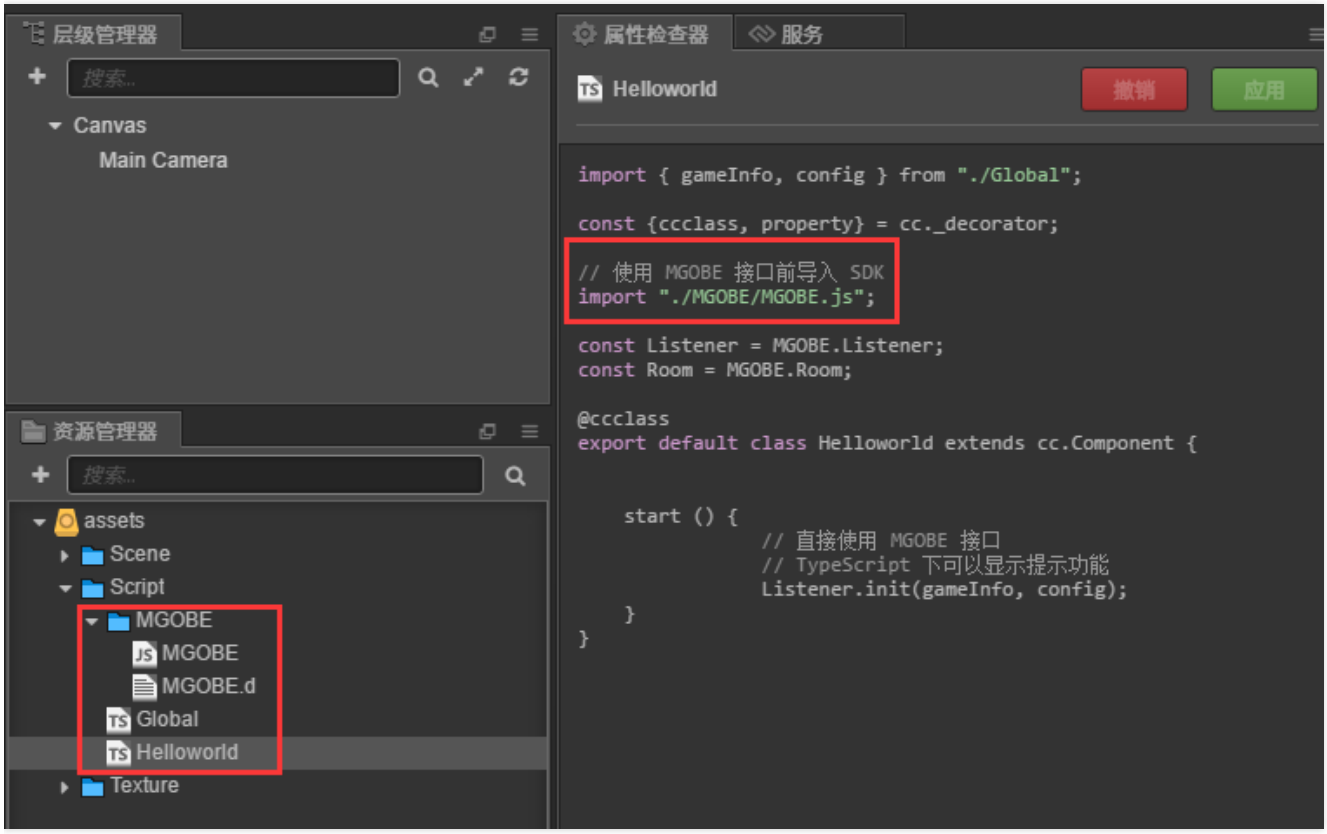
TypeScript 环境

在 Laya、Cocos 等支持直接使用 TypeScript 进行开发的集成环境中，您可以使用 TypeScript 自带的 import 语法导入 SDK。由于 TypeScript 支持 .d.ts 定义文件，为了方便开发，您可以将 MGOBE.js 和 MGOBE.d.ts 一同复制到项目中，再调用 import 语句即可。以 Cocos Creator 和 LayaAir IDE 为例：

Cocos Creator：

```
import { gameInfo, config } from "./Global";

const {ccclass, property} = cc._decorator;

// 使用 MGOBE 接口前导入 SDK
import "./MGOBE/MGOBE.js";

const Listener = MGOBE.Listener;
const Room = MGOBE.Room;

@ccclass
export default class Helloworld extends cc.Component {


start () {
// 直接使用 MGOBE 接口
// TypeScript 下可以显示提示功能
Listener.init(gameInfo, config);
}
}
```

界面示例如下图所示：

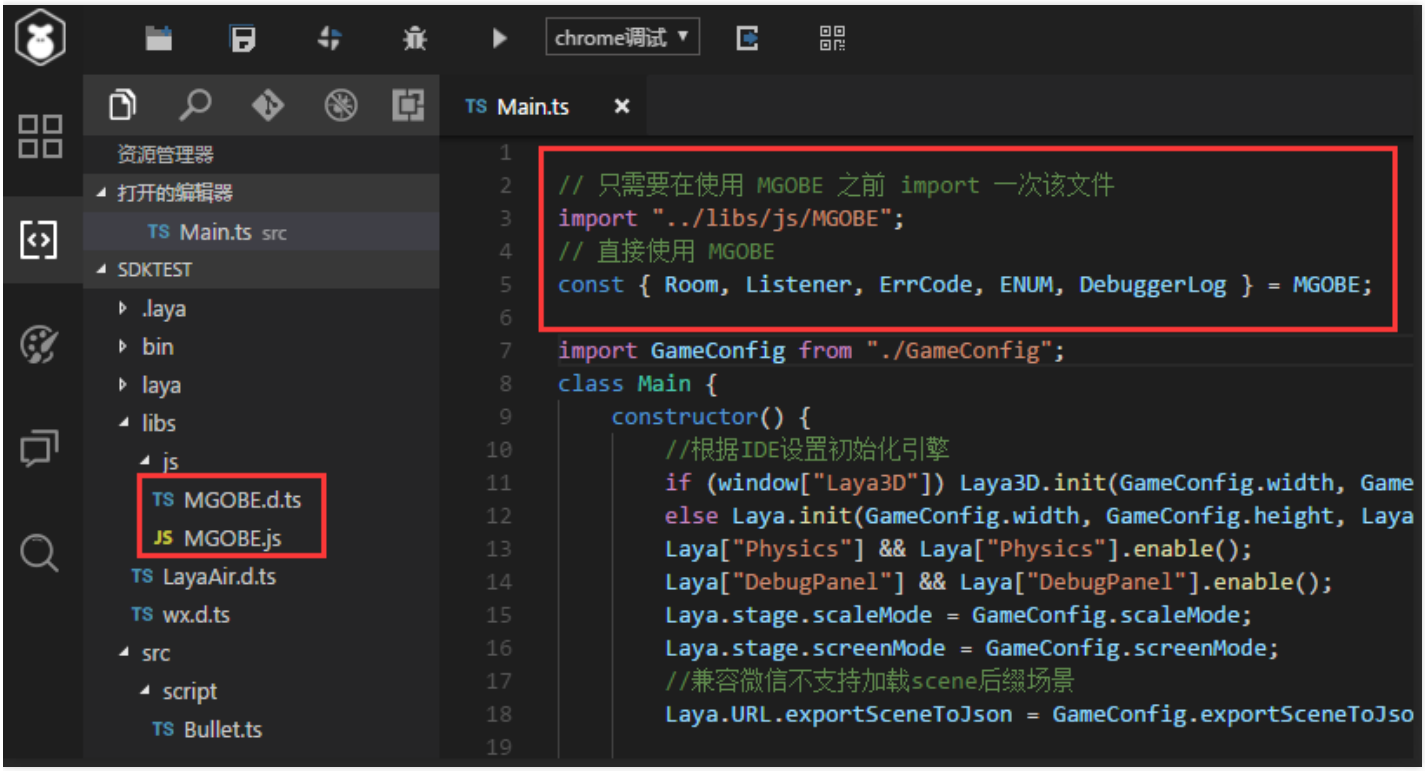


LayaAir IDE：

```
// 只需要在使用 MGOBE 之前 import 一次该文件
import "../libs/js/MGOBE";
```

```
// 直接使用 MGOBE
const { Room, Listener, ErrCode, ENUM, DebuggerLog } = MGOBE;
```

界面示例如下图所示：



此外，在 LayaAir IDE 中，您也可以直接在 bin/index.js 中直接使用 loadLib 函数导入 MGOBE.js，让 SDK 文件先执行一遍即可。在 TypeScript 环境中，您也可以使用 import/from 语法导入 MGOBE.js，但由于 TS 导入 .d.ts 的优先级高于导入 .js，所以您需要将 MGOBE.js 和 MGOBE.d.ts 文件放在不同文件夹，并使用 import/from 导入 .js 文件。

> 注意：
> 使用 import/from 语法方式导入 .js 将无法使用 .d.ts 提示。

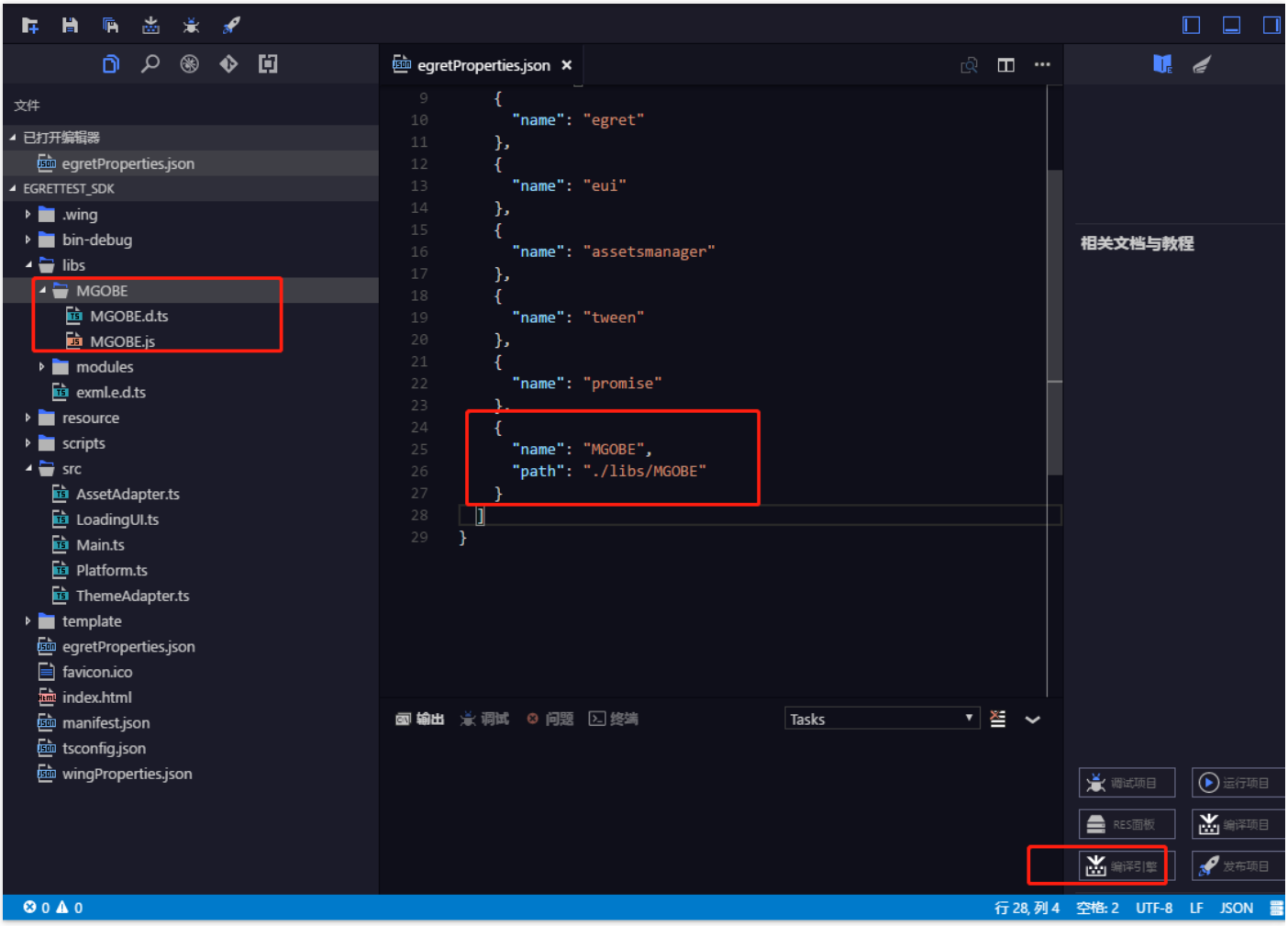
```
// import/from 导入 .js，无法使用 .d.ts 提示
import * as MGOBE from "../libs/js/MGOBE";
const { Room, Listener, ErrCode, ENUM, DebuggerLog } = MGOBE;
```

Egret 环境

1. 使用 Egret Wing 打开项目，在 libs 文件夹下，创建 MGOBE 文件夹，将 MGOBE.js、MGOBE.d.ts 拷贝到 MGOBE 文件夹。
2. 编辑 egretProperties.json 文件，在 modules 数组中新增 MGOBE 库的描述。

```
{
"name": "MGOBE",
"path": "./libs/MGOBE"
}
```

3. 运行编译引擎，完成 MGOBE SDK 的导入工作。在项目代码中可以直接使用 MGOBE 对象。



初始化监听器 Listener

在使用 MGOBE API 接口时，主要用到 Listener 对象和 Room 对象。导入 SDK 后，需要先初始化 Listener 对象。

```
const gameInfo = {
openId: 'xxxxxx',
gameId: "xxxxxx",// 替换为控制台上的“游戏ID”
secretKey: 'xxxxxx',// 替换为控制台上的“游戏key”“
};

const config = {
url: 'xxx.wxlagame.com',// 替换为控制台上的“域名”
reconnectMaxTimes: 5,
reconnectInterval: 1000,
resendInterval: 1000,
resendTimeout: 10000,
};

Listener.init(gameInfo, config, event => {
if (event.code === 0) {
```

```
// 初始化成功
// 继续在此添加代码
// ...
}
});
```

调用 Listener.init 时，需要传入游戏信息 gameInfo 和游戏配置 config 两个参数。

- gameInfo.gameId、gameInfo.secretKey 和 config.url 都需要根据控制台上的信息传入。
- gameInfo.openId 为玩家唯一 ID，例如，微信小游戏平台上的 OpenID。
- 其它字段由开发者自定义。

每个字段的具体含义可以参考 Listener 对象。

初始化成功后才能继续调用其他接口。因此，下文的 API 调用代码都应该在初始化回调函数内调用。

实例化 Room 对象

在开发游戏过程中的大部分业务接口都位于 Room 对象中。由于每个玩家只能加入一个房间，在游戏生命周期中可以实例化一个 Room 对象进行接口调用：

```
const room = new Room();
```

实例化 Room 后，可以调用 getMyRoom 接口来检查玩家是否已经加房，适用于应用重启后需要恢复玩家状态的场景。

```
// 查询玩家自己的房间
Room.getMyRoom(event => {
if (event.code === 0) {
// 设置房间信息到 room 实例
room.initRoom(event.data.roomInfo);
return console.log("玩家已在房间内：", event.data.roomInfo.name);
}

if (event.code === 20011) {
return console.log("玩家不在房间内");
}

return console.log("调用失败");
});
```

后续的创建房间、加房、匹配等接口调用直接利用 room 实例即可。

> 注意：
> getMyRoom、getRoomList、getRoomByRoomId 接口是 Room 对象的静态方法，您需要使用 Room.getMyRoom、Room.getRoomList、Room.getRoomByRoomId 进行调用。Room 的实例无法直接访问 getMyRoom、getRoomList、getRoomByRoomId。

Room 接收广播

一个房间对象会有很多广播事件与其相关，例如该房间有新成员加入、房间属性变化、房间开始对战等广播。Room 实例需要在 Listener 中注册广播监听，并且实现广播的回调函数。

注册广播监听

```
// 监听
Listener.add(room);
```

设置广播回调函数

```
// 广播：房间有新玩家加入
room.onJoinRoom = event => console.log("新玩家加入", event.data.joinPlayerId);
// 广播：房间有玩家退出
room.onLeaveRoom = event => console.log("玩家退出", event.data.leavePlayerId);
// 广播：房间被解散
room.onDismissRoom = event => console.log("房间被解散");
// 其他广播
// ...
```

其他广播接口可以参考 Room 对象。

移除监听

如果不想再接收该 Room 实例的广播，可以从 Listener 中移除：

```
// 移除监听
Listener.remove(room);
```

游戏对战

如果玩家已经加入房间，可以通过帧同步功能进行游戏对战。游戏过程中用到的接口有发送帧消息、帧广播，开发者可以利用这两个接口进行帧同步。帧广播的开始、结束需要使用房间的 startFrameSync、stopFrameSync 接口。

开始帧同步

使用 room.startFrameSync 接口可以开启帧广播。房间内任意一个玩家成功调用该接口将导致全部玩家开始接收帧广播。

```
// 发起请求
room.startFrameSync({}, event => {
if (event.code === ErrCode.EC_OK) {
console.log("请求成功");
}
});
// 广播：开始帧同步
room.onStartFrameSync = event => console.log("开始帧同步");
```

发送帧消息

玩家收到帧同步开始广播后可以发送帧消息，后台会将每个玩家的帧消息组合后广播给每个玩家。

```
const frame = { x: 100, y: 100, dir: 30, id: "xxxxxxxx" };
room.sendFrame({ data: frame }, event => {
if (event.code === ErrCode.EC_OK) {
console.log("发送成功");
}
});
```

接收帧广播

开发者可设置 room.onRecvFrame 广播回调函数获得帧广播数据。

```
// 广播：收到帧消息
room.onRecvFrame = event => {
console.log("帧广播", event.data);
};
```

停止帧同步

使用 room.stopFrameSync 接口可以停止帧广播。房间内任意一个玩家成功调用该接口将导致全部玩家停止接收帧广播。

```
// 发起请求
room.stopFrameSync({}, event => {
if (event.code === ErrCode.EC_OK) {
console.log("请求成功");
}
});
// 广播：停止帧同步
room.onStopFrameSync = event => console.log("停止帧同步");
```

API
SDK 模板类型说明

最近更新时间：2019-06-21 20:05:38

SDK 在使用过程中会收到两类消息，即响应消息和广播消息。

- 响应消息指由客户端主动发起请求后，服务器返回的响应，消息类型为 ResponseEvent。
- 广播消息指服务器主动向客户端发起的消息，消息类型为 BroadcastEvent。

每种响应、广播的数据由下文“响应消息”、“广播消息”定义。
客户端向服务器发起请求后，可以设置响应回调函数，回调函数类型由下文“响应回调函数”定义。

响应消息 MGOBE.types.ResponseEvent

MGOBE.types.ResponseEvent 的 TypeScript 定义如下：

```
interface ResponseEvent<T> {
code: number;
msg: string;
seq: string;
data?: T;
}
```

参数说明

参数名	类型	描述
code	number	消息错误码
msg	string	错误信息
seq	string	请求序列号
data	object	消息数据，由各消息回调接口定义

SDK 使用 Typescript 的模板类型定义了 data 字段，具体的 data 结构由 API 各接口定义。

- 如 `MGOBE.types.ResponseEvent<MGOBE.types.CreateRoomRsp>` 定义了创建房间的响应消息，其中 data 的类型为 `MGOBE.types.CreateRoomRsp` 。
- 由于有些响应消息没有 data 内容，API 将使用 `MGOBE.types.ResponseEvent<null>` 来表示这类响应消息。

广播消息 MGOBE.types.BroadcastEvent

MGOBE.types.BroadcastEvent 的 TypeScript 定义如下：

```
interface BroadcastEvent<T> {
data?: T;
}
```

参数说明

参数名	类型	描述
data	object	消息数据，由各消息回调接口定义

广播消息是由服务端主动发起，只含有 data 一个字段。Room 对象各个广播接口中有具体 data 定义。
如 `MGOBE.types.BroadcastEvent<MGOBE.types.DismissRoomBst>` 定义了解散房间广播消息，其中 data 的类型为 `MGOBE.types.DismissRoomBst` 。

响应回调函数 MGOBE.types.ReqCallback

MGOBE.types.ReqCallback 的 TypeScript 定义如下：

```
ReqCallback<T> = (event: MGOBE.types.ResponseEvent<T>) => any;
```

响应函数是使用 SDK 接口向后台发起请求后，后台返回消息时的回调函数。上述定义表明该函数的入参是响应消息 MGOBE.types.ResponseEvent，函数体由开发者自定义。

Player 对象

最近更新时间：2019-06-24 11:21:22

Player 对象为 MGOBE 的子属性，用来访问玩家的基本信息，如玩家 ID、openId 等。

对象描述

玩家信息。

参数描述

属性名	类型	描述
id	string	玩家 ID
openId	string	玩家 openId
name	string	玩家昵称
teamId	string	队伍 ID
customPlayerStatus	number	自定义玩家状态
customProfile	string	自定义玩家属性
commonNetworkState	MGOBE.types.NetworkState	房间网络状态
relayNetworkState	MGOBE.types.NetworkState	帧同步网络状态

说明：

- 该对象记录了玩家的基本信息，默认全部为空。成功初始化 Listener 之后，ID、openId 属性将生效。
- 玩家进入房间后，该对象的属性与 roomInfo.playerList 中当前玩家信息保持一致。
- 玩家 ID 是 MGOBE 后台生成的 ID，openId 是开发者初始化时候使用的 ID。openId 只有初始化 Listener 的时候使用，其它接口的"玩家 ID"均指后台生成的 ID。

Listener 对象

最近更新时间：2019-06-21 20:05:51

Listener 对象为 MGOBE 的子属性，该对象方法全为静态方法，不需要实例化。该对象主要用于给 Room 对象的实例绑定广播事件监听。

init

接口描述

初始化监听器。

参数描述

参数名	类型/值	描述
gameInfo	MGOBE.types.GameInfoPara	游戏信息
config	MGOBE.types.ConfigPara	游戏配置
callback	MGOBE.types.ReqCallback<null>	初始化回调函数

说明：

- 该方法为静态方法。初始化 Listener 时需要传入 gameInfo 和 config 两个参数。
- 初始化结果在 callback 中异步返回，错误码为 0 表示初始化成功。

返回值说明

无

使用示例

```
const gameInfo = {
gameId: "xxxxx",
openId: 'xxxxxx',
secretKey: 'xxxxxx',
};

const config = {
url: 'xxxxxxx.com',
reconnectMaxTimes: 5,
reconnectInterval: 1000,
resendInterval: 1000,
resendTimeout: 10000,
};

const room = new MGOBE.Room();

Listener.init(gameInfo, config, event => {
if (event.code === MGOBE.ErrCode.EC_OK) {
```

```
console.log("初始化成功");
// 初始化后才能添加监听
Listener.add(room);
} else {
console.log("初始化失败");
}
});
```

add

接口描述

为 Room 实例添加广播监听。

参数描述

参数名	类型/值	描述
room	Room	需要监听的房间对象

说明：

- 该方法为静态方法。实例化 Room 对象之后，需要通过该方法给 Room 注册广播事件监听。
- Listener 完成初始化之后才能添加监听。

返回值说明

无

使用示例

```
const room = new MGOBE.Room();
// 初始化后才能添加监听
Listener.add(room);
```

remove

接口描述

为Room实例移除广播监听。

参数描述

参数名	类型/值	描述
room	Room	需要移除监听的房间对象

说明：

- 该方法为静态方法。如果不再需要监听某个 Room 对象的广播事件，可以通过该方法进行移除。

返回值说明

无

使用示例

```
const room = new MGOBE.Room();
// 监听
Listener.add(room);
// 移除监听
Listener.remove(room);
```

clear

接口描述

移除全部Room对象的广播监听。

参数描述

无

> **说明：**
>
> 该方法为静态方法。

返回值说明

无

使用示例

```
Listener.clear();
```

Room 对象

概览

最近更新时间：2019-09-26 14:27:45

Room 对象属性概览表

构造器

名称	说明
constructor	构造器

房间管理相关接口

名称	说明
roomInfo	房间信息
networkState	该属性为只读属性，用于获取客户端本地 SDK 网络状态
initRoom	初始化 Room 实例的房间信息，即更新 roomInfo 属性
onUpdate	房间信息更新回调接口
createRoom	创建房间
createTeamRoom	创建团队房间
joinRoom	加入房间
joinTeamRoom	加入团队房间
leaveRoom	退出房间
dismissRoom	解散房间
changeRoom	修改房间信息
changeCustomPlayerStatus	修改玩家自定义状态
removePlayer	移除房间内玩家
getRoomDetail	获取 Room 实例的房间信息
getRoomByRoomId	根据房间 ID 获取房间
getMyRoom	查询玩家所在的房间信息
getRoomList	获取房间列表
onJoinRoom	新玩家加入房间广播回调接口

名称	说明
onLeaveRoom	玩家退出房间广播回调接口
onDismissRoom	房间被解散广播回调接口
onChangeRoom	房主修改房间信息广播回调接口
onRemovePlayer	房间内玩家被移除广播回调接口
onChangePlayerNetworkState	房间内玩家网络状态变化广播回调接口
onChangeCustomPlayerStatus	玩家自定义状态变化广播回调接口

匹配相关接口

名称	说明
matchPlayers	多人在线匹配
matchRoom	房间匹配
cancelPlayerMatch	取消玩家匹配

帧同步相关接口

名称	说明
startFrameSync	开始帧同步
stopFrameSync	停止帧同步
sendFrame	发送帧同步数据
requestFrame	请求补帧
onRecvFrame	房间帧消息广播回调接口
onStartFrameSync	开始帧同步广播回调接口
onStopFrameSync	停止帧同步广播回调接口
onAutoRequestFrameError	自动补帧失败回调接口
retryAutoRequestFrame	重试自动补帧

消息发送相关接口

名称	说明
sendToClient	发送消息给房间内玩家
onRecvFromClient	收到房间内其他玩家消息广播回调接口

名称	说明
sendToGameSvr	发送消息给自定义实时服务器
onRecvFromGameSvr	收到自定义服务消息广播回调接口

构造器

最近更新时间：2019-06-21 20:06:44

constructor

接口描述

构造器。

参数描述

参数名	类型/值	描述	可选
roomInfo	MGOBE.types.RoomInfo	房间信息	是

说明：

- 实例化 Room 对象时可以传入一个 MGOBE.types.RoomInfo 对象 roomInfo，后续接口调用都将基于该 roomInfo，例如修改该房间的属性、接收该房间的广播。
- 如果不传 roomInfo 参数，开发者可以通过直接调用 initRoom、createRoom、joinRoom 等方法获取 roomInfo。
- Room 对象会自动维护内部的 roomInfo 属性保持最新，开发者可以直接通过访问该属性获得最新的房间信息。

返回值说明

无

使用示例

```
// 示例1：不传 roomInfo 参数
// 该 Room 实例没有房间信息，room1.getRoomDetail 将查询玩家所在的房间
const room1 = new MGOBE.Room();

// 示例2：传 roomInfo 参数
// 该 Room 实例代表 ID 为 xxx 的房间，room2.getRoomDetail 将查询 xxx 房间信息
const roomInfo = {
id: "xxx",
// 其他字段
// ...
};
const room2 = new MGOBE.Room(roomInfo);
```

房间管理相关接口

最近更新时间：2019-11-05 09:49:32

roomInfo

对象描述

房间信息。

参数描述

无

> 说明：
> roomInfo 为 Room 实例的属性，类型为 MGOBE.types.RoomInfo，调用 Room 相关的接口会导致该属性发生变化。

networkState

对象描述

该属性为只读属性，用于获取客户端本地 SDK 网络状态。

参数描述

无

> 说明：
>
> - 该属性类型为 `{ COMMON: boolean, RELAY: boolean }` 。COMMON 表示房间网络状态；RELAY 表示帧同步网络状态。为 true 时表示网络已连接，为 false 时表示网络未连接。
> - 该网络状态与玩家信息中的网络状态（Player.commonNetworkState/Player.relayNetworkState）概念不同，前者表示本地 socket 状态，后者表示玩家在 MGOBE 后台中的状态。
> - 本地 socket 网络状态变化时，onUpdate 将被触发。

返回值说明

无

使用示例

```
const room = new Room();

// 示例1：不传 roomInfo 参数
// 该 Room 实例房间信息被清除，room.getRoomDetail 将查询玩家所在的房间
room.initRoom();

// 示例2：指定房间 ID
// 该 Room 实例代表 ID 为 xxx 的房间，room.getRoomDetail 将查询 xxx 房间信息
```

```
const roomInfo = { id: "xxx" };
room.initRoom(roomInfo);
```

initRoom

接口描述

初始化 Room 实例的房间信息，即更新 roomInfo 属性。

参数描述

参数名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo 或 { id: string }	初始化参数，id表示房间id

说明：

- initRoom 会更新 Room 实例的 roomInfo，接受 MGOBE.types.RoomInfo 或 { id: string; } 类型的参数。
- 如果不传参数，该方法将清空 Room 实例的 roomInfo 属性，此时调用 getRoomDetail 方法将查询玩家所在的房间。
- 当玩家需要加入指定 id 房间时，需要使用该接口初始化 Room 实例的 roomInfo 属性，然后才能通过调用 joinRoom 方法加入该 Room 实例所代表的房间。

返回值说明

无

使用示例

```
const room = new Room();

// 示例1：不传 roomInfo 参数
// 该 Room 实例房间信息被清除，room.getRoomDetail 将查询玩家所在的房间
room.initRoom();

// 示例2：指定房间 ID
// 该 Room 实例代表 ID 为 xxx 的房间，room.getRoomDetail 将查询 xxx 房间信息
const roomInfo = { id: "xxx" };
room.initRoom(roomInfo);
```

onUpdate

接口描述

房间信息更新回调接口。

参数描述

参数名	类型/值	描述	可选
room	Room	更新的 Room 实例	是

说明：

- onUpdate 表明 Room 实例的 roomInfo 信息发生变化，这种变化原因包括各种房间操作、房间广播、本地网络状态变化等。
- 开发者可以在该接口中更新游戏画面，或者使用 networkState 属性判断网络状态。

返回值说明

无

使用示例

```
room.onUpdate = (_) => {
console.log(_ === room); // true, 参数 _ 等于 room
console.log("房间信息更新", room.roomInfo);
};
```

createRoom

接口描述

创建房间。

参数描述

参数名	类型/值	描述
createRoomPara	MGOBE.types.CreateRoomPara	创建房间参数
callback	MGOBE.types.ReqCallback<MGOBE.types.CreateRoomRsp>	响应回调函数

说明：

- createRoom 调用结果将在 callback 中异步返回。操作成功后，roomInfo 属性将更新。
- 创建房间成功后，玩家自动进入该房间，因此无法继续调用 joinRoom、matchPlayers 等方法，可以利用房间 ID 邀请其他玩家进入该房间。

返回值说明

无

使用示例

```
const playerInfo = {
name: "Tom",
customPlayerStatus: 1,
customProfile: "https://xxx.com/icon.png",
};
```

```
const createRoomPara = {
roomName: "房间名",
maxPlayers: 4,
roomType: "2V2",
isPrivate: false,
customProperties: "WAIT",
playerInfo,
};

room.createRoom(createRoomPara, event => console.log(event));
```

createTeamRoom

接口描述

创建团队房间。

参数描述

参数名	类型/值	描述
createTeamRoomPara	MGOBE.types.CreateTeamRoomPara	创建团队房间参数
callback	MGOBE.types.ReqCallback<MGOBE.types.CreateRoomRsp>	响应回调函数

说明：

- createTeamRoom 调用结果将在 callback 中异步返回。操作成功后，roomInfo 属性将更新。
- 创建房间成功后，玩家自动进入该房间，因此无法继续调用 joinRoom、matchPlayers 等方法。
- 参数中的"房间最大玩家数量"要求能被"队伍数量"整除，创建成功后每个队伍的"队伍最小人数"为1，"队伍最大人数"为整除结果。

返回值说明

无

使用示例

```
const playerInfo = {
name: "Tom",
customPlayerStatus: 1,
customProfile: "https://xxx.com/icon.png",
};

const createTeamRoomPara = {
roomName: "房间名",
maxPlayers: 4,
roomType: "2V2",
isPrivate: false,
customProperties: "WAIT",
```

```
playerInfo,
teamNumber: 2,
};

room.createTeamRoom(createTeamRoomPara, event => console.log(event));
```

joinRoom

接口描述

加入房间。

参数描述

参数名	类型/值	描述
joinRoomPara	MGOBE.types.JoinRoomPara	加入房间参数
callback	MGOBE.types.ReqCallback<MGOBE.types.JoinRoomRsp>	响应回调函数

说明：

- joinRoom 调用结果将在 callback 中异步返回。
- 该接口加入的房间是 Room 实例所代表的房间，如果该 Room 实例的 roomInfo 不存在 roomId，则需要使用 roomId 通过 init 方法初始化 Room 实例。
- 加房成功后，房间内全部成员都会收到一条玩家加入房间广播 onJoinRoom，roomInfo 属性将更新。

返回值说明

无

使用示例

```
const playerInfo = {
name: "Tom",
customPlayerStatus: 1,
customProfile: "https://xxx.com/icon.png",
};

const joinRoomPara = {
playerInfo,
};

// 示例1：加入指定 ID 的房间
const room1 = new MGOBE.Room();
room1.initRoom({ id: "xxx" });
room1.joinRoom(joinRoomPara, event => console.log(event));

// 示例2：加入没有房间信息的房间
const room2 = new MGOBE.Room();
```

```
// 加房失败，找不到房间信息
room2.joinRoom(joinRoomPara, event => console.log(event));
```

joinTeamRoom

接口描述

加入团队房间。

参数描述

参数名	类型/值	描述
joinTeamRoomPara	MGOBE.types.JoinTeamRoomPara	加入团队房间参数
callback	MGOBE.types.ReqCallback<MGOBE.types.JoinRoomRsp>	响应回调函数

说明：

- joinTeamRoom 调用结果将在 callback 中异步返回。
- 与 joinRoom 类似，该接口加入的房间是 Room 实例所代表的房间。teamId 为 roomInfo.teamList 中定义的队伍 ID。

返回值说明

无

使用示例

```
const playerInfo = {
name: "Tom",
customPlayerStatus: 1,
customProfile: "https://xxx.com/icon.png",
};

const joinTeamRoomPara = {
playerInfo,
teamId: "1",
};

room.joinTeamRoom(joinTeamRoomPara, event => console.log(event));
```

leaveRoom

接口描述

退出房间。

参数描述

参数名	类型/值	描述

参数名	类型/值	描述
para	object	预留参数，传{}即可
callback	MGOBE.types.ReqCallback<MGOBE.types.LeaveRoomRsp>	响应回调函数

说明：

- leaveRoom 调用结果将在 callback 中异步返回。退出成功后，房间内剩余成员都会收到一条玩家退出房间广播 onLeaveRoom，roomInfo 属性将更新，roomInfo.playerList 中将没有该玩家信息。
- 退房后，如果房间内还剩下其他玩家，则该 room 实例仍然代表退房前的房间，可以继续调用 room.initRoom() 清除房间信息。

返回值说明

无

使用示例

```
room.leaveRoom({}, event => {
if (event.code === 0) {
// 退房成功
console.log("退房成功", room.roomInfo.id);
// 可以使用 initRoom 清除 roomInfo
room.initRoom();
}
});
```

dismissRoom

接口描述

解散房间。

参数描述

参数名	类型/值	描述
para	object	预留参数，传{}即可
callback	MGOBE.types.ReqCallback<MGOBE.types.DismissRoomRsp>	响应回调函数

说明：

- dismissRoom 调用结果将在 callback 中异步返回。解散成功后，房间内全部成员都会收到一条
- 广播 onDismissRoom，roomInfo 属性将更新。
- 只有房主有权限解散房间

返回值说明

无

使用示例

```
room.dismissRoom({}, event => {
if (event.code === 0) {
console.log("解散成功");
}
});
```

changeRoom

接口描述

修改房间信息。

参数描述

参数名	类型/值	描述
changeRoomPara	MGOBE.types.ChangeRoomPara	修改房间参数
callback	MGOBE.types.ReqCallback<MGOBE.types.ChangeRoomRsp>	响应回调函数

说明：

- changeRoom 调用结果将在 callback 中异步返回。修改成功后，房间内全部成员都会收到一条修改房间广播 onChangeRoom，roomInfo 属性将更新。
- 只有房主有权限修改房间

返回值说明

无

使用示例

```
const changeRoomPara = {
roomName: "房间名",
owner: "xxxxxx",
isPrivate: false,
customProperties: "xxxxxx",
};

room.changeRoom(changeRoomPara, event => console.log(event));
```

changeCustomPlayerStatus

接口描述

修改玩家自定义状态。

参数描述

参数名	类型/值	描述
changeCustomPlayerStatusPara	MGOBE.types.ChangeCustomPlayerStatusPara	修改玩家状态参数
callback	MGOBE.types.ReqCallback<MGOBE.types.ChangeCustomPlayerStatusRsp>	响应回调函数

说明：

- 修改玩家状态是修改 PlayerInfo 中的 customPlayerStatus 字段，玩家的状态由开发者自定义。
- 修改成功后，房间内全部成员都会收到一条修改玩家状态广播 onChangeCustomPlayerStatus，roomInfo 属性将更新。
- 每个玩家只能修改自己的状态，调用结果将在 callback 中异步返回。

返回值说明

无

使用示例

```
const changeCustomPlayerStatusPara = {
customPlayerStatus: 2,
};

room.changeCustomPlayerStatus(changeCustomPlayerStatusPara, event => console.log(event));
```

removePlayer

接口描述

移除房间内玩家。

参数描述

参数名	类型/值	描述
removePlayerPara	MGOBE.types.RemovePlayerPara	移除房间内玩家参数
callback	MGOBE.types.ReqCallback<MGOBE.types.RemovePlayerRsp>	响应回调函数

说明：

- 调用结果将在 callback 中异步返回。移除玩家成功后，房间内全部成员都会收到一条移除玩家广播 onRemovePlayer，roomInfo 属性将更新。
- 只有房主有权限移除其他玩家

返回值说明

无

使用示例

```
const removePlayerPara = {
removePlayerId: "xxxxxx",
};

room.removePlayer(removePlayerPara, event => console.log(event));
```

getRoomDetail

接口描述

获取 Room 实例的房间信息。

参数描述

参数名	类型/值	描述
callback	MGOBE.types.ReqCallback<MGOBE.types.GetRoomByRoomIdRsp>	响应回调函数

说明：

- 该接口获取的是 Room 实例的房间信息，调用结果将在 callback 中异步返回。
- 如果该 Room 实例中的 roomInfo 属性没有 ID，该接口将查询玩家所在的房间。
- 如果 roomInfo 属性含有 ID，则查询该 ID 对应的房间信息。
- 操作成功后，roomInfo 属性将更新。
- 如果需要获取指定 ID 的房间信息，可以使用 getRoomByRoomId 方法。

返回值说明

无

使用示例

```
// 示例1：查询room实例的信息
room.getRoomDetail(event => {
if (event.code === 0) {
console.log("房间名", event.data.roomInfo.name);
}
});
```

```
// 示例2：查询玩家所在房间信息
room.initRoom();// 或者 room.initRoom({ id: "" });
room.getRoomDetail(event => {
if (event.code === 0) {
console.log("房间名", event.data.roomInfo.name);
}
});
```

getRoomByRoomId

接口描述

根据房间 ID 获取房间。

参数描述

参数名	类型/值	描述
getRoomByRoomIdPara	MGOBE.types.GetRoomByRoomIdPara	获取房间参数
callback	MGOBE.types.ReqCallback<MGOBE.types.GetRoomByRoomIdRsp>	响应回调函数

说明：

- 调用结果将在 callback 中异步返回。
- 该接口为 Room 的静态方法，只能通过 Room.getRoomByRoomId 方式调用，Room 实例无法直接访问该方法。
- 如果参数中的 roomId 为空字符串，将查询玩家所在的房间。

返回值说明

无

使用示例

```
// 示例1：查询指定房间id的信息
const getRoomByRoomIdPara1 = {
roomId: "800000",
};
MGOBE.Room.getRoomByRoomId(getRoomByRoomIdPara1, event => console.log(event));

// 示例2：查询玩家所在房间信息
const getRoomByRoomIdPara2 = {
roomId: "",
};
MGOBE.Room.getRoomByRoomId(getRoomByRoomIdPara2, event => console.log(event));
```

getMyRoom

接口描述

查询玩家所在的房间信息。

参数描述

参数名	类型/值	描述
callback	MGOBE.types.ReqCallback<MGOBE.types.GetRoomByRoomIdRsp>	响应回调函数

说明：

- 调用结果将在 callback 中异步返回。
- 该接口为 Room 的静态方法，只能通过 Room.getMyRoom 方式调用，Room 实例无法直接访问该方法。

返回值说明

无

使用示例

```
MGOBE.Room.getMyRoom(event => {
if (event.code === 0) {
console.log("玩家在房间内", event.data.roomInfo);
const room = new MGOBE.Room(event.data.roomInfo);
}
});
```

getRoomList

接口描述

获取房间列表。

参数描述

参数名	类型/值	描述
getRoomListPara	MGOBE.types.GetRoomListPara	获取房间列表参数
callback	MGOBE.types.ReqCallback<MGOBE.types.GetRoomListRsp>	响应回调函数

说明：

- 调用结果将在 callback 中异步返回。
- 该接口为 Room 的静态方法，只能通过 Room.getRoomList 方式调用，Room 实例无法直接访问该方法。

返回值说明

无

使用示例

```
const getRoomListPara = {
pageNo: 1,
pageSize: 10,
};

// 不要使用 room.getRoomList
// 直接使用 Room 对象
MGOBE.Room.getRoomList(getRoomListPara, event => console.log(event));
```

onJoinRoom

接口描述

新玩家加入房间广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.JoinRoomBst>	回调参数

> 说明：
> onJoinRoom 广播表示该房间有新玩家加入。房间内全部成员都会收到该广播。

返回值说明

无

使用示例

```
room.onJoinRoom = event => console.log("新玩家加入", event.data.joinPlayerId);
```

onLeaveRoom

接口描述

玩家退出房间广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.LeaveRoomBst>	回调参数

> 说明：
> onLeaveRoom 广播表示该房间有玩家退出。房间内全部成员都会收到该广播。

返回值说明

无

使用示例

```
room.onLeaveRoom = event => console.log("玩家退出", event.data.leavePlayerId);
```

onDismissRoom

接口描述

房间被解散广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.DismissRoomBst>	回调参数

> 说明：
> onDismissRoom 广播表示房主解散了该房间。房间内全部成员都会收到该广播。

返回值说明

无

使用示例

```
room.onDismissRoom = event => console.log("房间已被房主解散");
```

onChangeRoom

接口描述

房主修改房间信息广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.ChangeRoomBst>	回调参数

> 说明：
> onChangeRoom 广播表示房主修改了该房间属性。房间内全部成员都会收到该广播。

返回值说明

无

使用示例

```
room.onChangeRoom = event => console.log("房间属性变更", event.data.roomInfo);
```

onRemovePlayer

接口描述

房间内玩家被移除广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.RemovePlayerBst>	回调参数

说明：
onRemovePlayer 广播表示有玩家被房主移除。房间内全部成员都会收到该广播。

返回值说明

无

使用示例

```
room.onRemovePlayer = event => console.log("玩家被移除", event.data.removePlayerId);
```

onChangePlayerNetworkState

接口描述

房间内玩家网络状态变化广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.ChangePlayerNetworkStateBst>	回调参数

说明：

- onChangePlayerNetworkState 广播表示 ID 为 changePlayerId 的玩家网络状态发生变化。
- 玩家在房间中、帧同步中的网络变化都会触发该广播，因此 networkState 将有四中情况，分别表示房间中上下线、帧同步中上下线。

返回值说明

无

使用示例

```
room.onChangePlayerNetworkState = event => {
if (event.data.networkState === MGOBE.ENUM.NetworkState.COMMON_OFFLINE)
console.log("玩家下线");
};
```

onChangeCustomPlayerStatus

接口描述

玩家自定义状态变化广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.ChangeCustomPlayerStatusBst>	回调参数

> 说明：
> onChangeCustomPlayerStatus 广播表示房间内 ID 为 changePlayerId 的玩家状态发生变化。玩家状态由开发者自定义。

返回值说明

无

使用示例

```
room.onChangeCustomPlayerStatus = event => {
console.log("玩家状态变化", event.data.changePlayerId);
};
```

匹配相关接口

最近更新时间：2019-12-11 10:36:26

matchPlayers

接口描述

多人在线匹配。

参数描述

参数名	类型/值	描述
matchPlayersPara	MGOBE.types.MatchPlayersPara	多人匹配参数
callback	MGOBE.types.ReqCallback<MGOBE.types.MatchPlayersRsp>	响应回调函数

说明：

- 调用该接口后将发起多人在线匹配，callback 将异步返回调用结果。返回码为0表示匹配成功，Room 对象内部 roomInfo 属性将自动更新。
- 该接口需要与匹配规则配合使用，因此，匹配超时时间由开发者在匹配规则中定义。开发者需要在控制台上创建匹配，得到匹配 Code 作为该方法的参数 matchCode。
- matchPlayersPara.playerInfo 中的 matchAttributes 数组对应匹配规则中定义的 playerAttributes，playerAttributes 的每一种属性都要填入 matchAttributes 中，name 表示属性名，value 表示玩家该属性的值。

返回值说明

无

使用示例

```
const playerInfo = {
name: "Tom",
customPlayerStatus: 1,
customProfile: "https://xxx.com/icon.png",
matchAttributes: [{
name: "skill1",
value: 99,
}]
};

const matchPlayersPara = {
playerInfo,
matchCode: "play-xxx",
};

// 发起匹配
room.matchPlayers(matchPlayersPara, event => {
```

```
if (event.code === 0) {
console.log("匹配成功", room.roomInfo);
} else {
console.log("匹配失败", event.code);
}
});
```

matchRoom

接口描述

房间匹配。

参数描述

参数名	类型/值	描述
matchRoomPara	MGOBE.types.MatchRoomPara	房间匹配参数
callback	MGOBE.types.ReqCallback<MGOBE.types.MatchRoomSimpleRsp>	响应回调函数

说明：

- 调用该接口后将发起房间匹配，匹配结果将在 callback 中异步返回。操作成功后，Room 对象内部 roomInfo 属性将更新。
- 房间匹配是指按照传入的参数搜索现存的房间，如果存在，则将玩家加入该房间；如果不存在，则为玩家创建并加入一个新房间。

返回值说明

无

使用示例

```
const playerInfo = {
name: "Tom",
customPlayerStatus: 1,
customProfile: "https://xxx.com/icon.png",
};

const matchRoomPara = {
playerInfo,
maxPlayers: 5,
roomType: "1",
};

room.matchRoom(matchRoomPara, event => {
if (event.code !== 0) {
console.log("匹配失败", event.code);
```

```
}
});
```

cancelPlayerMatch

接口描述

取消玩家匹配。

参数描述

参数名	类型/值	描述
cancelMatchPara	MGOBE.types.CancelPlayerMatchPara	取消匹配参数
callback	MGOBE.types.ReqCallback<MGOBE.types.CancelPlayerMatchRsp>	响应回调函数

说明：

- 该接口作用是取消多人匹配请求，即 matchPlayers 请求。调用结果将在 callback 中异步返回。如果玩家已经在房间中，回调函数将返回 roomInfo。
- cancelMatchPara.matchType 需要设置为 MGOBE.ENUM.MatchType.PLAYER_COMPLEX。

返回值说明

无

使用示例

```
const cancelMatchPara = {
matchType: MGOBE.ENUM.MatchType.PLAYER_COMPLEX,
};

room.cancelPlayerMatch(cancelMatchPara, event => console.log(event));
```

帧同步相关接口

最近更新时间：2019-12-05 20:54:08

startFrameSync

接口描述

开始帧同步。

参数描述

参数名	类型/值	描述
para	object	预留参数，传{}即可
callback	MGOBE.types.ReqCallback<MGOBE.types.StartFrameSyncRsp>	响应回调函数

说明：

- 调用结果将在 callback 中异步返回。调用成功后房间内全部成员将收到 onStartFrameSync 广播。该接口会修改房间帧同步状态为“已开始帧同步”。
- 房间内任意一个玩家成功调用该接口将导致全部玩家开始接收帧广播。

返回值说明

无

使用示例

```
room.startFrameSync({}, event => {
if (event.code === 0) {
console.log("开始帧同步成功");
}
});
```

stopFrameSync

接口描述

停止帧同步。

参数描述

参数名	类型/值	描述
para	object	预留参数，传{}即可
callback	MGOBE.types.ReqCallback<MGOBE.types.StoptFrameSyncRsp>	响应回调函数

说明：

- 调用结果将在 callback 中异步返回。调用成功后房间内全部成员将收到 onStopFrameSync 广播。该接口会修改房间帧同步状态为“已停止帧同步”。
- 房间内任意一个玩家成功调用该接口将导致全部玩家停止接收帧广播。

返回值说明

无

使用示例

```
room.stopFrameSync({}, event => console.log(event));
```

sendFrame

接口描述

发送帧同步数据。

参数描述

参数名	类型/值	描述
sendFramePara	MGOBE.types.SendFramePara	发送帧同步数据参数
callback	MGOBE.types.ReqCallback<MGOBE.types.SendFrameRsp>	响应回调函数

说明：

- 帧数据内容 data 类型为普通 object，由开发者自定义，目前支持最大长度不超过1k。
- 后台将集合全部玩家的帧数据，并以一定时间间隔（由房间帧率定义）通过 onRecvFrame 广播给各客户端。调用结果将在 callback 中异步返回。
- 只有房间处于“已开始帧同步”状态才能调用该接口。

返回值说明

无

使用示例

```
const frame = {cmd: "xxxxxxxx", id: "xxxxxxxx" };
const sendFramePara = { data: frame };
room.sendFrame(sendFramePara, event => console.log(event));
```

requestFrame

接口描述

请求补帧。

参数描述

参数名	类型/值	描述
requestFramePara	MGOBE.types.RequestFramePara	请求补帧参数
callback	MGOBE.types.ReqCallback<MGOBE.types.RequestFrameRsp>	响应回调函数

说明：

- 调用结果将在 callback 中异步返回。

返回值说明

无

使用示例

```
const requestFramePara = {
beginFrameId: 100,
endFrameId: 120,
};

room.requestFrame(requestFramePara, event => console.log(event));
```

onRecvFrame

接口描述

房间帧消息广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.RecvFrameBst>	回调参数

说明：

- onRecvFrame 广播表示收到一个帧 frame，frame 的内容由多个 MGOBE.types.FrameItem 组成，即一帧时间内房间内所有玩家向服务器发送帧消息的集合。

返回值说明

无

使用示例

```
room.onRecvFrame = event => {
console.log("帧广播", event.data.frame);
};
```

onStartFrameSync

接口描述

开始帧同步广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.StartFrameSyncBst>	回调参数

说明：

- onStartFrameSync 广播表示房间开始帧同步。收到该广播后将持续收到 onRecvFrame 广播。

返回值说明

无

使用示例

```
room.onStartFrameSync = event => console.log("开始帧同步");
```

onStopFrameSync

接口描述

停止帧同步广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.StopFrameSyncBst>	回调参数

说明：

- onStopFrameSync 广播表示房间停止帧同步。收到该广播后将不再收到 onRecvFrame 广播。

返回值说明

无

使用示例

```
room.onStopFrameSync = event => console.log("停止帧同步");
```

onAutoRequestFrameError

接口描述

自动补帧失败回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.ResponseEvent<MGOBE.types.RequestFrameRsp>>	回调参数

说明：

- onAutoRequestFrameError 表示自动补帧失败，在初始化 Listener 时开启自动补帧后才能触发。
- 发生补帧失败后，将不能收到帧广播，开发者可以使用 retryAutoRequestFrame 方法重试自动补帧。

返回值说明

无

使用示例

```
room.onAutoRequestFrameError = event => {
console.log("自动补帧失败", event.data.code);
// 重试
room.retryAutoRequestFrame();
};
```

retryAutoRequestFrame

接口描述

重试自动补帧。

参数描述

无

说明：

- 当收到 onAutoRequestFrameError 回调时，表示自动补帧失败，可以使用该方法重新触发自动补帧。

返回值说明

无

使用示例

```
room.onAutoRequestFrameError = event => {
console.log("自动补帧失败", event.data.code);
// 重试
room.retryAutoRequestFrame();
};
```

消息发送相关接口

最近更新时间：2019-08-08 15:35:47

sendToClient

接口描述

发送消息给房间内玩家。

参数描述

参数名	类型/值	描述
sendToClientPara	MGOBE.types.SendToClientPara	发送消息参数
callback	MGOBE.types.ReqCallback<MGOBE.types.SendToClientRsp>	响应回调函数

说明：

- 调用结果将在 callback 中异步返回。调用成功后所指定的接收消息的玩家将收到 onRecvFromClient 广播。
- 当 recvType 值为1（即 ROOM_ALL ）时，房间内全部玩家将收到消息。
- 当 recvType 值为2（即 ROOM_OTHERS ）时，房间内除消息发送者外的其他玩家将收到消息。
- 当 recvType 值为3（即 ROOM_SOME ）时，接收消息玩家才由 recvPlayerList 决定。

返回值说明

无

使用示例

```
const sendToClientPara = {
recvType: MGOBE.ENUM.RecvType.ROOM_SOME,
recvPlayerList: ["xxxxxxxx1", "xxxxxxxx2"],
msg: "hello",
};

room.sendToClient(sendToClientPara, event => console.log(event));
```

onRecvFromClient

接口描述

收到房间内其他玩家消息广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.RecvFromClientBst>	回调参数

sendToGameSvrPara	MGOBE.types.SendToGameSvrPara	发送消息参数
callback	MGOBE.types.ReqCallback<MGOBE.types.SendToGameSvrRsp>	响应回调函数

说明：

该接口只能在玩家进入房间后调用，调用结果将在 callback 中异步返回。

返回值说明

无

使用示例

```
const sendToGameServerPara = {
data: {
cmd: 1,
},
};

room.sendToGameSvr(sendToGameServerPara, event => console.log(event));
```

onRecvFromGameSvr

接口描述

收到自定义服务消息广播回调接口。

参数描述

参数名	类型/值	描述
event	MGOBE.types.BroadcastEvent<MGOBE.types.RecvFromGameSvrBst>	回调参数

说明：

onRecvFromGameSvr 广播表示收到来自自定义服务的消息。

返回值说明

无

使用示例

```
room.onRecvFromGameSvr = event => console.log("新自定义服务消息", event);
```

ErrCode 错误码对象

最近更新时间：2019-05-15 19:43:10

该属性是一个 object，记录了 SDK 全部错误码，具体的 key 和 value 与错误码文档相对应。详情请参考 错误码。该对象结构示例如下：

```
ErrCode = {
//返回成功
EC_OK: 0,
//请求包格式错误
EC_REQ_BAD_PKG: 1,
// 其他错误码
// ...
}
```

ENUM 枚举对象

最近更新时间：2019-06-21 20:07:45

ENUM 对象为 MGOBE 的子属性，记录了 SDK 常用的几种枚举数据，且均由枚举名称（key）和值（value）组成的 object。
ENUM 对象全部属性如下：

ENUM

对象描述

操作类型枚举。

参数描述

属性名	类型/值	描述
CreateRoomType	MGOBE.types.CreateRoomType	创建房间方式
MatchType	MGOBE.types.MatchType	匹配类型
NetworkState	MGOBE.types.NetworkState	网络状态
FrameSyncState	MGOBE.types.FrameSyncState	房间帧同步状态
RecvType	MGOBE.types.RecvType	消息接收者范围

DebuggerLog 日志打印

最近更新时间：2019-02-28 17:22:29

接口描述

SDK 内部日志打印工具，记录接口关键调用步骤。

参数说明

名称	类型	描述
enable	boolean	是否开启控制台日志打印
callback	Function	日志回调函数

说明：

设置回调函数后，每一条日志将作为该函数的参数。

使用示例

```
MGOBE.DebuggerLog.enable = true;
MGOBE.DebuggerLog.callback = (log) => console.log(...log); // 默认值;
```

RandomUtil 随机数工具

最近更新时间：2019-06-21 20:08:07

MGOBE SDK 内部实现了一个基于“线性同余算法”的随机数生成方法。RandomUtil 对象方法如下：

init

接口描述

初始化随机数。

参数描述

参数名	类型/值	描述
seed	number	随机数种子

> **说明：**
>
> init 方法接受一个 seed 为参数，RandomUtil 在后续生成随机数的过程中将以 seed 为种子。使用相同的 seed 初始化，调用 random 方法生成的随机数序列相同。

返回值说明

无

使用示例

```
MGOBE.RandomUtil.init(12345678);
```

random

接口描述

生成随机数。

参数描述

无

> **说明：**
>
> 如果种子相同、初始化后调用次数相同，生成的随机数将相同。

返回值说明

返回值类型为 number，表示随机数。

使用示例

```
const num = MGOBE.RandomUtil.random();
```

对象类型定义

最近更新时间：2019-11-22 15:15:00

PlayerInfoPara

对象描述

玩家信息参数。

参数描述

属性名	类型/值	描述
name	string	玩家昵称
customPlayerStatus	number	自定义玩家状态
customProfile	string	自定义玩家信息

MatchPlayerInfoPara

对象描述

玩家信息参数。

参数描述

属性名	类型/值	描述
name	string	玩家昵称
customPlayerStatus	number	自定义玩家状态
customProfile	string	自定义玩家信息
matchAttributes	MGOBE.types.MatchAttribute[]	匹配属性

GameInfoPara

对象描述

初始化参数：游戏信息。

参数描述

属性名	类型/值	描述
gameId	string	游戏 ID
openId	string	玩家 openId
secretKey	string	游戏密钥
createSignature	MGOBE.types.CreateSignature	签名函数

说明：

- 游戏密钥指控制台上的“游戏 key”。在初始化 SDK 时，secretKey、CreateSignature 两个参数传其中一个即可。如果实现了 CreateSignature 方法，则忽略 secretKey 参数。
- CreateSignature 用于计算签名 signature，优点在于避免客户端泄露游戏密钥。

ConfigPara

对象描述

初始化参数：配置参数。

参数描述

属性名	类型/值	描述
reconnectMaxTimes	number	重连接次数
reconnectInterval	number	重连接时间间隔
resendInterval	number	消息重发时间间隔
resendTimeout	number	消息重发超时时间
url	string	服务地址
isAutoRequestFrame	boolean	是否自动补帧
cacertNativeUrl	string	本地 CA 根证书路径（CocosNative 环境需要该参数）

说明：
服务地址指控制台上的“域名”。

Signature

对象描述

初始化签名。

参数描述

属性名	类型/值	描述
sign	string	签名
nonce	number	随机正整数（uint64 类型）
timestamp	number	时间戳，秒（uint64 类型）

说明：
可以使用签名的方式初始化 SDK，避免客户端泄露游戏密钥。

CreateSignature

对象描述

签名函数。

参数描述

属性名	类型/值	描述
callback	(signature: MGOBE.types.Signature) => any	回调函数，在该函数返回 Signature 对象

说明：
开发者如果使用签名方式初始化 SDK，需要实现该方法，并在 callback 中回调 Signature 对象。

ChangeCustomPlayerStatusPara

对象描述

修改玩家状态参数。

参数描述

属性名	类型/值	描述
customPlayerStatus	number	自定义玩家状态

CreateRoomPara

对象描述

创建房间参数。

参数描述

属性名	类型/值	描述
roomName	string	房间名称
roomType	string	房间类型
maxPlayers	number	房间最大玩家数量
isPrivate	boolean	是否私有
customProperties	string	自定义房间属性

属性名	类型/值	描述
playerInfo	MGOBE.types.PlayerInfoPara	玩家信息

CreateTeamRoomPara

对象描述

创建团队房间参数。

参数描述

属性名	类型/值	描述
roomName	string	房间名称
roomType	string	房间类型
maxPlayers	number	房间最大玩家数量
isPrivate	boolean	是否私有
customProperties	string	自定义房间属性
playerInfo	MGOBE.types.PlayerInfoPara	玩家信息
teamNumber	number	队伍数量

JoinRoomPara

对象描述

加入房间参数。

参数描述

属性名	类型/值	描述
playerInfo	MGOBE.types.PlayerInfoPara	玩家信息

JoinTeamRoomPara

对象描述

加入团队房间参数。

参数描述

属性名	类型/值	描述
playerInfo	MGOBE.types.PlayerInfoPara	玩家信息
teamId	string	队伍 ID

ChangeRoomPara

对象描述

房间变更参数。

参数描述

属性名	类型/值	描述	可选
roomName	string	房间名称	是
owner	string	房主 ID	是
isPrivate	boolean	是否私有	是
customProperties	string	自定义房间属性	是
isForbidJoin	boolean	是否禁止加入房间	是

RemovePlayerPara

对象描述

移除房间内玩家参数。

参数描述

属性名	类型/值	描述
removePlayerId	string	被移除玩家 ID

GetRoomListPara

对象描述

获取房间列表参数。

参数描述

属性名	类型/值	描述	可选
pageNo	number	页号，从1开始	否
pageSize	number	每页数量，最大为10	否
roomType	string	房间类型	是
isDesc	boolean	是否按照房间创建时间倒序	是

GetRoomByRoomIdPara

对象描述

获取房间参数。

参数描述

属性名	类型/值	描述
roomId	string	房间 ID

MatchPlayersPara

对象描述

多人匹配参数。

参数描述

属性名	类型/值	描述
matchCode	string	匹配 Code
playerInfo	MGOBE.types.MatchPlayerInfoPara	玩家信息

> 说明：
> 匹配 Code 需要在控制台创建匹配获得。

MatchRoomPara

对象描述

房间匹配参数。

参数描述

属性名	类型/值	描述
playerInfo	MGOBE.types.PlayerInfoPara	玩家信息
maxPlayers	number	房间最大玩家数量
roomType	string	房间的类型

CancelPlayerMatchPara

对象描述

取消匹配参数。

参数描述

属性名	类型/值	描述
matchType	MGOBE.types.MatchType	匹配类型

SendFramePara

对象描述

发送帧数据参数。

参数描述

属性名	类型/值	描述
data	object	帧数据

RequestFramePara

对象描述

请求补帧参数。

参数描述

属性名	类型/值	描述
beginFrameId	number	起始帧号
endFrameId	number	结束帧号

说明：
补帧范围大于等于 beginFrameId，小于等于 endFrameId。

RecvType

对象描述

消息接收者类型。

参数描述

属性名	类型/值	描述
ROOM_ALL	1	全部玩家
ROOM_OTHERS	2	除自己外的其他玩家
ROOM_SOME	3	房间中部分玩家

SendToClientPara

对象描述

发送房间内消息参数。

参数描述

属性名	类型/值	描述

属性名	类型/值	描述
recvPlayerList	string[]	接收消息玩家 ID 列表
msg	string	消息内容
recvType	MGOBE.types.RecvType	消息接收者类型

SendToGameSvrPara

对象描述

发自定义服务消息参数。

参数描述

属性名	类型/值	描述
data	object	消息内容

RecvFromGameSvrBst

对象描述

自定义服务消息广播回调参数。

参数描述

属性名	类型/值	描述
roomId	number	房间 ID
recvPlayerIdList	string[]	接收消息玩家 ID 列表
data	object	消息内容

FrameItem

对象描述

帧内容。

参数描述

属性名	类型/值	描述
playerId	string	玩家 ID
data	object	玩家帧内容
timestamp	number	时间戳，各玩家本地发送帧的时间

Frame

对象描述

帧数据。

参数描述

属性名	类型/值	描述
frameId	number	帧 ID
items	MGOBE.types.FrameItem[]	帧内容
ext	MGOBE.types.FrameExtInfo	附加信息
roomId	number	房间 ID
time	number	该帧到达客户端时间
isReplay	boolean	是否为补帧

说明：

- 附加信息包含一个 number 类型随机种子，开发者可以使用帧 ID 与随机种子组合成一个值来初始化 RandomUtil 工具。
- time 为 SDK 拟合出来的时间，目的是使每一帧到达客户端的时间尽量均匀分布，并且时间间隔尽量接近帧率的倒数。
- isReplay 表示该帧是否为自动补帧产生的帧，自动补帧需要在初始化 Listener 时设置。

RecvFrameBst

对象描述

帧广播回调参数。

参数描述

属性名	类型/值	描述
frame	MGOBE.types.Frame	帧数据

RequestFrameRsp

对象描述

请求补帧回调参数。

参数描述

属性名	类型/值	描述
frames	MGOBE.types.Frame[]	帧数据数组

JoinRoomBst

对象描述

玩家加入房间广播回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息
joinPlayerId	string	加房玩家 ID

LeaveRoomBst

对象描述

玩家退出房间广播回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息
leavePlayerId	string	退房玩家 ID

DismissRoomBst

对象描述

房间被解散广播回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间解散前的信息

ChangeRoomBst

对象描述

房间属性变更广播回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

RemovePlayerBst

对象描述

房间内玩家被移除广播回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息
removePlayerId	string	被移除玩家 ID

RecvFromClientBst

对象描述

房间消息广播回调参数。

参数描述

属性名	类型/值	描述
roomId	number	房间 ID
sendPlayerId	string	发送者 ID
msg	string	消息内容

ChangePlayerNetworkStateBst

对象描述

房间内玩家网络状态变化广播回调参数。

参数描述

属性名	类型/值	描述
changePlayerId	string	玩家 ID
networkState	MGOBE.types.NetworkState	网络状态
roomInfo	MGOBE.types.RoomInfo	房间信息

ChangeCustomPlayerStatusBst

对象描述

玩家自定义状态变化广播回调参数。

参数描述

属性名	类型/值	描述
changePlayerId	string	玩家 ID
customPlayerStatus	number	自定义玩家信息
roomInfo	MGOBE.types.RoomInfo	房间信息

StartFrameSyncBst

对象描述

开始帧同步广播回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

StopFrameSyncBst

对象描述

停止帧同步广播回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

RoomInfo

对象描述

房间属性。

参数描述

属性名	类型/值	描述
id	string	房间 ID
name	string	房间名称
type	string	房间类型
createType	MGOBE.types.CreateRoomType	创建房间方式
maxPlayers	number	房间最大玩家数量
owner	string	房主 ID
isPrivate	boolean	是否私有
customProperties	string	房间自定义属性
playerList	MGOBE.types.PlayerInfo[]	玩家列表
teamList	MGOBE.types.TeamInfo[]	团队属性
frameSyncState	MGOBE.types.FrameSyncState	房间帧同步状态

属性名	类型/值	描述
frameRate	number	帧率
routeId	string	路由 ID
createTime	number	房间创建时的时间戳（单位：秒）
startGameTime	number	开始帧同步时的时间戳（单位：秒）
isForbidJoin	number	是否禁止加入房间

说明：
isPrivate 属性为 true 表示该房间为私有房间，不能被 matchRoom 接口匹配到。

ChangeCustomPlayerStatusRsp

对象描述

修改玩家状态回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

CreateRoomRsp

对象描述

创建房间回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

JoinRoomRsp

对象描述

加入房间回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

LeaveRoomRsp

对象描述

退出房间回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

DismissRoomRsp

对象描述

解散房间回调参数。

参数描述

无

ChangeRoomRsp

对象描述

修改房间回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

RemovePlayerRsp

对象描述

移除房间内玩家回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

GetRoomByRoomIdRsp

对象描述

获取房间信息回调参数。

参数描述

属性名	类型/值	描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

GetRoomListRsp

对象描述

获取房间列表回调参数。

参数描述

属性名	类型/值	描述
roomList	MGOBE.types.RoomInfo[]	房间列表
total	number	房间总数

MatchPlayersRsp

对象描述

多人匹配回调参数。

参数描述

属性名	类型/值	描述
matchType	MGOBE.types.MatchType	匹配类型
roomInfo	MGOBE.types.RoomInfo	房间信息

MatchRoomSimpleRsp

对象描述

房间匹配回调参数。

参数描述

属性名	类型/值	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

CancelPlayerMatchRsp

对象描述

取消匹配回调参数。

参数描述

无

StartFrameSyncRsp

对象描述

开始帧同步回调参数。

参数描述

无

StopFrameSyncRsp

对象描述

停止帧同步回调参数。

参数描述

无

SendFrameRsp

对象描述

发送帧同步数据回调参数。

参数描述

无

SendToClientRsp

对象描述

房间内发送消息回调参数。

参数描述

无

SendToGameSvrRsp

对象描述

发送自定义服务消息回调参数。

参数描述

无

MatchAttribute

对象描述

匹配属性。

参数描述

属性名	类型/值	描述

属性名	类型/值	描述
name	string	属性名称
value	number	属性值

MatchType

对象描述

匹配类型。

参数描述

属性名	类型/值	描述
ROOM_SIMPLE	1	房间匹配
PLAYER_COMPLEX	2	玩家匹配

FrameExtInfo

对象描述

帧数据附加信息。

参数描述

属性名	类型/值	描述
seed	number	随机数种子

CreateRoomType

对象描述

创建房间方式。

参数描述

属性名	类型/值	描述
COMMON_CREATE	0	普通创建
MATCH_CREATE	1	匹配创建

NetworkState

对象描述

网络状态。

参数描述

属性名	类型/值	描述
COMMON_OFFLINE	0	房间中玩家掉线
COMMON_ONLINE	1	房间中玩家在线
RELAY_OFFLINE	2	帧同步中玩家掉线
RELAY_ONLINE	3	帧同步中玩家在线

FrameSyncState

对象描述

房间帧同步状态。

参数描述

属性名	类型/值	描述
STOP	0	未开始帧同步
START	1	已开始帧同步

PlayerInfo

对象描述

玩家信息。

参数描述

属性名	类型/值	描述
id	string	玩家 ID
name	string	玩家昵称
teamId	string	队伍 ID
customPlayerStatus	number	自定义玩家状态
customProfile	string	自定义玩家信息
commonNetworkState	MGOBE.types.NetworkState	玩家在房间的网络状态
relayNetworkState	MGOBE.types.NetworkState	玩家在游戏中的网络状态
isRobot	boolean	玩家是否为机器人
matchAttributes	MGOBE.types.MatchAttribute[]	玩家匹配属性列表（isRobot 为 true 时生效）

TeamInfo

对象描述

队伍信息。

参数描述

属性名	类型/值	描述
id	string	队伍 ID
name	string	队伍名称
minPlayers	number	队伍最小人数
maxPlayers	number	队伍最大人数

实时服务器
框架下载

最近更新时间：2019-10-11 14:17:11

为方便开发者创建自定义服务逻辑，在下载 示例代码之前，请查阅 更新历史。

平台	更新时间	版本	示例代码	文档
Nodejs	2019/9/24	V1.1.0	下载	操作指南 开发指南

使用简介

最近更新时间：2019-08-13 14:41:22

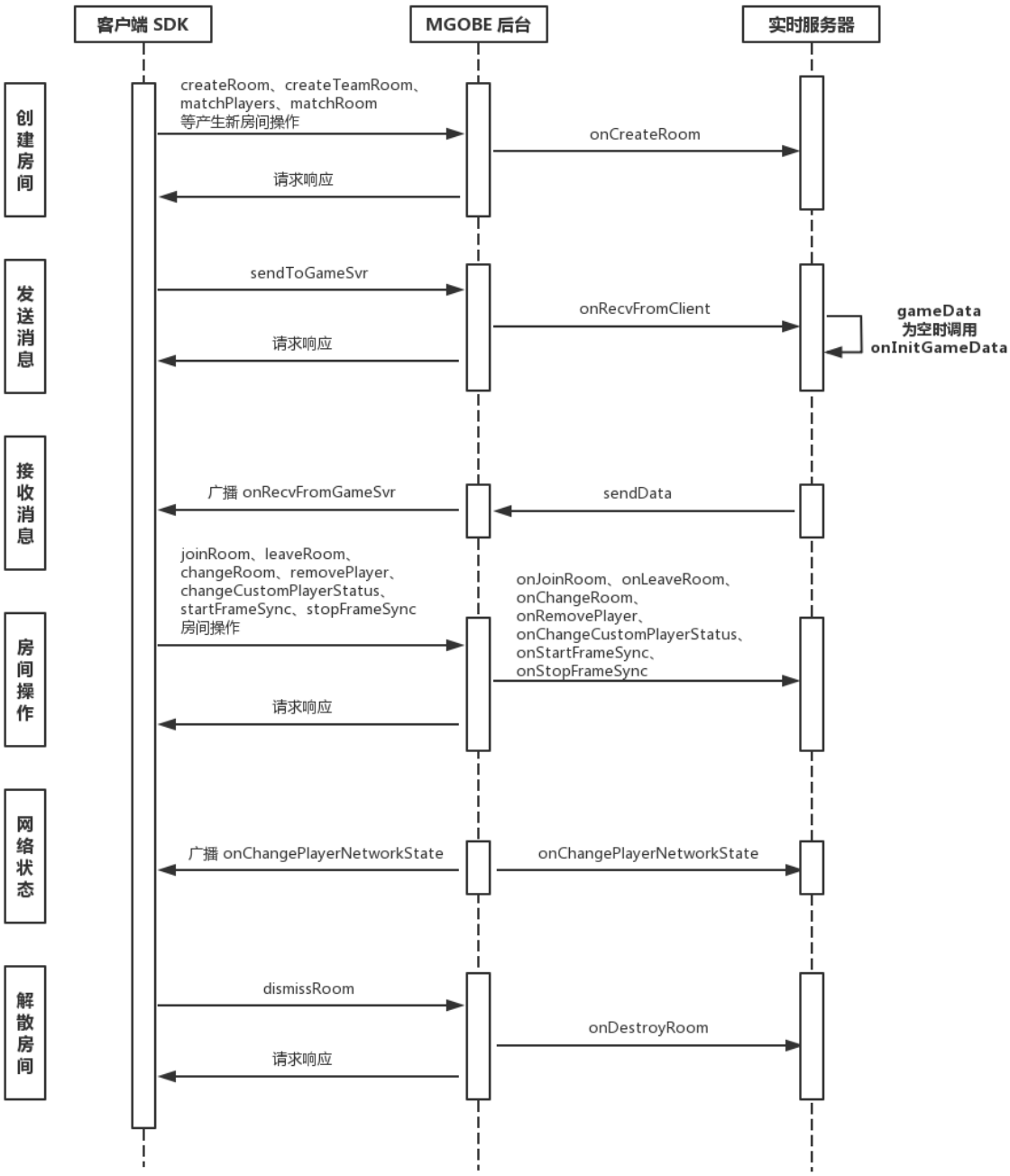
自定义服务逻辑的 实时服务器 部分包含两部分内容：

1. 房间的系列操作会自动触发到自定义服务逻辑。
2. 前端请求到自定义服务逻辑。当玩家加房成功后，可以使用客户端 SDK 中的 sendToGameServer 方法直接与游戏服务器通信，实现游戏服务端拓展逻辑，如保存玩家数据，游戏状态同步等。

实时服务器时序图

> 注意：
> onInitGameData 方法是在 onCreateRoom 之后，收到 onRecvFromClient 广播时检查 gameData。
>
> - 如果 gameData 为空，先执行 onInitGameData 再执行 onRecvFromClient。
> - 如果 gameData 不为空，只执行 onRecvFromClient。

开通自定义实时服务器后，客户端发起的房间相关操作，均被 MGOBE 后台广播到自定义实时服务器。客户端、 MGOBE 后台、自定义实时服务器三者交互时序图如下：



上传脚本

开发者只需要提供一个 index.js 脚本，然后按照以下文件夹结构压缩成 zip 文件，在 MGOBE 控制台上传即可。

```
├── mgobexs
│   └── index.js
```

注意：
zip 文件的根目录内只能有一个文件夹，建议命名为 mgobexs。mgobexs 目录下的 index.js 文件是实时服务器入口。

示例代码

开发者在接入实时服务器时，需要在 index.js 中导出一个 mgobexsCode 对象，该对象拥有一个 gameServer 属性。Node.js 示例代码如下：

```
exports.mgobexsCode = {
gameServer
};
```

在 gameServer 中需要实现 onInitGameData、onRecvFromClient、onCreateRoom、onJoinRoom 等接口。NodeJS 示例代码如下：

```
const gameServer = {
// 消息模式
mode: 'sync',
// 初始化游戏数据
onInitGameData: function () {
return {};
},
// 监听客户端数据
onRecvFromClient: function onClientData(args) {
args.SDK.logger.debug("onRecvFromClient");
args.SDK.exitAction();
},
// 监听加房广播
onJoinRoom: function (args) {
args.SDK.logger.debug("onJoinRoom");
},
// 监听创建房间广播
onCreateRoom: function (args) {
args.SDK.logger.debug("onCreateRoom");
},
// 监听退房广播
onLeaveRoom: function (args) {
args.SDK.logger.debug("onLeaveRoom");
},
// 监听玩家被移除广播
onRemovePlayer: function (args) {
args.SDK.logger.debug("onRemovePlayer");
},
// 监听房间销毁广播
onDestroyRoom: function (args) {
args.SDK.logger.debug("onDestroyRoom");
},
// 监听修改房间属性广播
onChangeRoom: function (args) {
```

```
args.SDK.logger.debug("onChangeRoom");
},
// 监听修改玩家自定义状态广播
onChangeCustomPlayerStatus: function (args) {
args.SDK.logger.debug("onChangeCustomPlayerStatus");
},
// 监听玩家网络状态变化广播
onChangePlayerNetworkState: function (args) {
args.SDK.logger.debug("onChangePlayerNetworkState");
},
// 监听开始帧同步广播
onStartFrameSync: function (args) {
args.SDK.logger.debug("onStartFrameSync");
},
// 监听停止帧同步广播
onStopFrameSync: function (args) {
args.SDK.logger.debug("onStopFrameSync");
}
};
```

查看日志

开发者使用 SDK.logger 输出的日志，您可前往控制台选择【实时服务器】，单击【查看日志】链接进行查看。

API
mgobexsCode 对象

最近更新时间：2019-09-26 14:23:23

mgobexsCode 对象是自定义服务的入口，开发者需要在代码中导出该对象。

gameServer 属性

描述

gameServer 是 mgobexsCode 对象的一个属性，类型为 GameServer.IGameServer 。开发者需要实现一个 GameServer.IGameServer 对象，并赋值给 gameServer。

使用示例

```
const gameServer = {
// 消息模式
mode: 'sync',
// 初始化游戏数据
onInitGameData: function () {
return {};
},
// 监听客户端数据
onRecvFromClient: function onRecvFromClient(args) {
args.SDK.logger.debug("onRecvFromClient");
// 发送消息给客户端
args.SDK.sendData({ playerIdList: [], data: { msg: "hello" } });
args.SDK.exitAction();
}
};

exports.mgobexsCode = {
gameServer
};
```

logLevelSDK 属性

描述

logLevelSDK 是 mgobexsCode 对象的一个属性，类型为字符串，表示实时服务器内部日志的打印级别。只能填写以下值：

值	含义
debug+	打印 debug、info、error
info+	打印 info、error
error+	打印 error

使用示例

```
exports.mgobexsCode = {
logLevelSDK: "debug+",
gameServer
};
```

logLevel 属性

描述

logLevel 是 mgobexsCode 对象的一个属性，类型为字符串，表示开发者代码内使用 ActionArgs.SDK.logger 时的日志打印级别。只能填写以下值：

值	含义
debug+	打印 debug、info、error
info+	打印 info、error
error+	打印 error

使用示例

```
exports.mgobexsCode = {
logLevel: "debug+",
gameServer
};
```

onInitGameServer 属性

描述

onInitGameServer 是 mgobexsCode 对象的一个属性，类型为 function。该函数在实时服务器初始化后会被调用，开发者可以在该函数内初始化 TCB 实例。

参数说明

参数名	类型	描述
tcb	object	腾讯云云开发模块

返回值说明

无。

使用示例

```
exports.mgobexsCode = {
onInitGameServer: (tcb) => {
// 可以在此初始化 TCB
const tcbApp = tcb.init({
secretId: "请填写腾讯云API密钥ID",
```

```
secretKey: "请填写腾讯云API密钥KEY",
env: "请填写云开发环境ID",
serviceUrl: 'http://tcb-admin.tencentyun.com/admin',
timeout: 5000,
});
},
gameServer
};
```

gameInfo 属性

描述

gameInfo 是 mgobexsCode 对象的一个属性，类型为 object。开发者如果需要在实时服务器调用 getRoomByRoomId、changeRoom、changeCustomPlayerStatus、removePlayer 方法需要实现该对象。该对象属性如下：

属性名	类型	描述
gameId	string	游戏 ID，从控制台获取
serverKey	string	后端密钥，从控制台获取

使用示例

```
exports.mgobexsCode = {
gameInfo: {
gameId: "请填写游戏ID，从控制台获取",
serverKey: "请填写后端密钥，从控制台获取",
},
gameServer
};
```

GameServer.IGameServer 对象

最近更新时间：2019-08-08 15:34:51

GameServer.IGameServer 对象即实时服务器接口。提供了接收客户端消息、监听房间广播相关接口。

mode 属性

描述

mode 是实时服务器处理客户端消息的模式。可以取值为 "sync" 或 "async"。

- 当 mode 为 "sync" 时，实时服务器将使用同步模式处理客户端消息。开发者在 onRecvFromClient 回调中必须显式调用 SDK.exitAction 方法，实时服务器才能处理下一条 onRecvFromClient 广播。
- 当 mode 为 "async" 时，实时服务器将使用异步模式处理客户端消息。每次监听到 onRecvFromClient 广播时都将执行回调函数。

使用示例

```
const gameServer = {};
gameServer.mode = "async";
```

onInitGameData 接口

描述

初始化游戏数据时的回调接口。

参数说明

参数名	类型	描述
args	{ room: IRoomInfo; }	回调参数

IRoomInfo 定义如下：

字段名	类型	描述
id	number	房间 ID
name	string	房间名称
type	string	房间类型
createType	CreateType	创建房间方式，参考 枚举类型 一节
maxPlayers	number	房间最大玩家数量
owner	string	房主 ID
isPrivate	boolean	是否私有
customProperties	string	房间自定义属性

字段名	类型	描述
playerList	IPlayerInfo[]	玩家列表
teamList	ITeamInfo[]	团队属性
frameSyncState	FrameSyncState	房间状态，参考 枚举类型 一节
frameRate	number	帧率
routeId	string	路由ID
createTime	number	房间创建时的时间戳，秒
startGameTime	number	开始帧同步时的时间戳，秒

ITeamInfo 定义如下：

字段名	类型	描述
id	string	队伍 ID
name	string	队伍名称
minPlayers	number	队伍最小人数
maxPlayers	number	队伍最大人数

IPlayerInfo 定义如下：

字段名	类型	描述
id	string	玩家 ID
name	string	玩家昵称
teamId	string	队伍 ID
customPlayerStatus	number	自定义玩家状态
customProfile	string	自定义玩家信息
commonNetworkState	NetworkState	玩家在房间的网络状态，参考 枚举类型 一节，只取房间中的两种状态
relayNetworkState	NetworkState	玩家在游戏中的网络状态，参考 枚举类型 一节，只取游戏中的两种状态

返回值说明

在该接口需要返回一个 GameData 类型数据，该数据会作为游戏初始化数据，在整个房间被销毁之前都能使用。

GameData 默认为 object 类型，开发者可以根据需要进行自定义。

说明：

onInitGameData 方法是在 onCreateRoom 之后，收到 onRecvFromClient 广播时检查 gameData：如果 gameData 为空，先执行 onInitGameData 再执行 onRecvFromClient；如果 gameData 不为空，只执行 onRecvFromClient。

使用示例

```
const gameServer = {};
gameServer.onInitGameData = args => {
return { room: args.room };
};
```

onRecvFromClient 接口

描述

接收玩家消息回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<UserDefinedData>	回调参数

ActionArgs 定义请参考 ActionArgs 对象。

UserDefinedData 即玩家的消息类型，类型为 object。开发者可以根据需要进行自定义。

返回值说明

无。

说明：
mode 为 "sync" 时需要在该回调里面显式调用 args.SDK.exitAction 方法才能继续处理下一条 onRecvFromClient 广播消息。

使用示例

```
const gameServer = {};
gameServer.mode = "sync";
gameServer.onRecvFromClient = args => {
args.SDK.sendData({ playerIdList: [], data: { msg: "hello" } });
args.SDK.exitAction();
};
```

onCreateRoom 接口

描述

创建房间广播回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<ICreateRoomBst>	回调参数

ICreateRoomBst 定义如下：

字段名	类型	描述
roomInfo	IRoomInfo	房间信息

返回值说明

无。

使用示例

```
const gameServer = {};
gameServer.onCreateRoom = args => {
args.SDK.logger.debug("onCreateRoom");
};
```

onJoinRoom 接口

描述

玩家加房广播回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<IJoinRoomBst>	回调参数

IJoinRoomBst 定义如下：

字段名	类型	描述
roomInfo	IRoomInfo	房间信息
joinPlayerId	string	加房玩家 ID

返回值说明

无。

使用示例

```
const gameServer = {};
gameServer.onJoinRoom = args => {
```

```
args.SDK.logger.debug("onJoinRoom");
};
```

onLeaveRoom 接口

描述

玩家退房广播回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<ILeaveRoomBst>	回调参数

ILeaveRoomBst 定义如下：

字段名	类型	描述
roomInfo	IRoomInfo	房间信息
leavePlayerId	string	退房玩家 ID

返回值说明

无。

使用示例

```
const gameServer = {};
gameServer.onLeaveRoom = args => {
args.SDK.logger.debug("onLeaveRoom");
};
```

onRemovePlayer 接口

描述

移除玩家广播回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<IRemovePlayerBst>	回调参数

IRemovePlayerBst 定义如下：

字段名	类型	描述
roomInfo	IRoomInfo	房间信息

字段名	类型	描述
removePlayerId	string	被移除玩家 ID

返回值说明

无。

使用示例

```
const gameServer = {};
gameServer.onRemovePlayer = args => {
args.SDK.logger.debug("onRemovePlayer");
};
```

onChangeRoom 接口

描述

修改房间属性广播回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<IChangeRoomBst>	回调参数

IChangeRoomBst 定义如下：

字段名	类型	描述
roomInfo	IRoomInfo	房间信息

返回值说明

无。

使用示例

```
const gameServer = {};
gameServer.onChangeRoom = args => {
args.SDK.logger.debug("onChangeRoom");
};
```

onChangeCustomPlayerStatus 接口

描述

修改玩家自定义状态广播回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<IChangeCustomPlayerStatusBst>	回调参数

IChangeCustomPlayerStatusBst 定义如下：

字段名	类型	描述
changePlayerId	string	玩家 ID
customPlayerStatus	number	玩家状态
roomInfo	IRoomInfo	房间信息

返回值说明

无。

使用示例

```
const gameServer = {};
gameServer.onChangeCustomPlayerStatus = args => {
args.SDK.logger.debug("onChangeCustomPlayerStatus");
};
```

onChangePlayerNetworkState 接口

描述

玩家网络状态变化广播回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<IChangePlayerNetworkStateBst>	回调参数

IChangePlayerNetworkStateBst 定义如下：

字段名	类型	描述
changePlayerId	string	玩家 ID
networkState	NetworkState	网络状态，有四种情况，参考 枚举类型 一节
roomInfo	IRoomInfo	房间信息

返回值说明

无。

使用示例

```
const gameServer = {};
gameServer.onChangePlayerNetworkState = args => {
args.SDK.logger.debug("onChangePlayerNetworkState");
};
```

onDestroyRoom 接口

描述

销毁房间广播回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<IDestroyRoomBst>	回调参数

IDestroyRoomBst 定义如下：

字段名	类型	描述
roomInfo	IRoomInfo	房间信息

返回值说明

无。

使用示例

```
const gameServer = {};
gameServer.onDestroyRoom = args => {
args.SDK.logger.debug("onDestroyRoom");
};
```

onStartFrameSync 接口

描述

开始帧同步广播回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<IStartFrameSyncBst>	回调参数

IStartFrameSyncBst 定义如下：

字段名	类型	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

返回值说明

无。

使用示例

```
const gameServer = {};
gameServer.onStartFrameSync = args => {
args.SDK.logger.debug("onStartFrameSync");
};
```

onStopFrameSync 接口

描述

停止帧同步广播回调接口。

参数说明

参数名	类型	描述
args	ActionArgs<IStopFrameSyncBst>	回调参数

IStopFrameSyncBst 定义如下：

字段名	类型	描述
roomInfo	MGOBE.types.RoomInfo	房间信息

返回值说明

无。

使用示例

```
const gameServer = {};
gameServer.onStopFrameSync = args => {
args.SDK.logger.debug("onStopFrameSync");
};
```

ActionArgs 类型

最近更新时间：2019-09-26 14:24:40

ActionArgs 是一个模板类型，其 TypeScript 定义如下：

```
interface ActionArgs<T> {
sender: string;
actionData: T;
gameData: GameData;
room: IRoomInfo;
exports: { data: GameData; };
SDK: {
sendData: (data: { playerIdList: string[]; data: UserDefinedData; }) => void;
dispatchAction: (actionData: UserDefinedData) => void;
clearAction: () => void;
exitAction: () => void;
logger: {
debug: (...args: any[]) => void;
info: (...args: any[]) => void;
error: (...args: any[]) => void;
};
};
}
```

因此，模板类型指定了 actionData 的类型。例如，在 gameServer.onRecvFromClient 接口中，入参是 ActionArgs<UserDefinedData> 类型，表明 actionData 类型为 UserDefinedData。

sender 属性

描述

该属性在 gameServer.onRecvFromClient 中有效，其类型为 string，表示消息发送者的玩家 ID。

actionData 属性

描述

该属性在 gameServer 不同回调中的类型不同，表示该回调的响应数据。例如，在 gameServer.onRecvFromClient 中表示玩家发送给实时服务器的数据；在 onJoinRoom 表示加房广播数据；在 onLeaveRoom 中表示玩家退房广播数据。

gameData 属性

描述

该属性类型为 GameData，表示游戏数据，开发者可以用来实现游戏状态同步等功能。在第一次执行 gameServer.onRecvFromClient 时会被初始化，在执行 gameServer.onDestroyRoom 时会被销毁。

room 属性

描述

该属性类型为 IRoomInfo，表示当前房间信息。

exports 属性

描述

该属性类型为 object，包含了一个类型为 GameData 的子属性 data，用于更新游戏数据 gameData。

如果开发者需要重新给 gameData 赋值，可以参考以下代码：

```
exports.data = {};
```

SDK 属性

描述

该属性类型为 object，包含了一系列实时服务器提供的方法。

SDK.sendData 方法

描述

实时服务器向客户端推送消息。

参数说明

参数名	类型	描述
data	{ playerIdList: string[]; data: UserDefinedData; }	消息内容

说明：

- data.playerIdList 表示接收消息的玩家列表。数组为空表示发给房间内全部玩家。
- data.data 为具体消息，类型为 UserDefinedData，即 object。

返回值说明

无。

使用示例

```
let data = { playerIdList: [], data: { msg: "hello" } };
SDK.sendData(data);
```

SDK.dispatchAction 方法

描述

模拟客户端给实时服务器发送数据。

参数说明

参数名	类型	描述
actionData	UserDefinedData	消息内容

返回值说明

无。

> 说明：
> 使用该方法后，下次 gameServer.onRecvFromClient 接口回调将处理该方法发送的消息。

使用示例

```
let actionData = { data: "hello" };
SDK.dispatchAction(actionData);
```

SDK.clearAction 方法

描述

清空 onRecvFromClient 队列。

参数说明

无。

返回值说明

无。

> 说明：
> 当 gameServer.mode 为 "sync" 时，gameServer.onRecvFromClient 广播会保存在一个队列里面，在 gameServer.onRecvFromClient 回调函数中通过调用 SDK.exitAction 才能处理下一条 gameServer.onRecvFromClient 广播。SDK.clearAction 作用就是清空 gameServer.onRecvFromClient 队列，可用于游戏结束后实时服务器忽略客户端消息的场景。

使用示例

```
SDK.clearAction();
```

SDK.exitAction 方法

描述

结束 gameServer.onRecvFromClient 方法。

参数说明

无。

返回值说明

无。

说明：
当 gameServer.mode 为 "sync" 时，需要在 gameServer.onRecvFromClient 回调里面显式调用 SDK.exitAction 方法才能继续处理下一条 gameServer.onRecvFromClient 广播消息。

使用示例

```
SDK.exitAction();
```

SDK.logger 属性

描述

logger 是 SDK 提供的日志记录能力，可以使用 logger.debug、logger.info、logger.error 三种日志级别进行记录。记录的日志可以在 MGOBE 控制台的实时服务器页面查看。

SDK.getRoomByRoomId 方法

描述

根据房间 ID 查询房间信息。

参数说明

参数名	类型	描述
getRoomByRoomIdPara	IGetRoomByRoomIdPara	请求参数
callback	ReqCallback<IGetRoomByRoomIdRsp>	回调函数

IGetRoomByRoomIdPara 定义如下：

属性名	类型/值	描述
roomId	string	房间 ID

IGetRoomByRoomIdRsp 定义如下：

属性名	类型/值	描述
roomInfo	IRoomInfo	房间信息

返回值说明

无。

说明：
调用该接口需要在 mgobexsCode 配置正确的游戏 ID 和后端密钥。

使用示例

```
const getRoomByRoomIdPara = { roomId: "xxx", };
SDK.getRoomByRoomId(getRoomByRoomIdPara, event => {
console.log(event.code, event.data);
});
```

SDK.changeRoom 方法

描述

修改指定房间的房间信息。

参数说明

参数名	类型	描述
changeRoomPara	IChangeRoomPara	请求参数
callback	ReqCallback<IChangeRoomRsp>	回调函数

IChangeRoomPara 定义如下：

属性名	类型/值	描述	可选
roomId	string	房间 ID	
roomName	string	房间名称	是
owner	string	房主ID	是
isPrivate	boolean	是否私有	是
isForbidJoin	boolean	是否禁止加入房间	是
customProperties	string	自定义房间属性	是

IChangeRoomRsp 定义如下：

属性名	类型/值	描述
roomInfo	IRoomInfo	房间信息

返回值说明

无。

> 说明：
> 调用该接口需要在 mgobexsCode 配置正确的游戏 ID 和后端密钥。

使用示例

```
const changeRoomPara = { roomId: "xxx", roomName: "xxx" };
SDK.changeRoom(changeRoomPara, event => {
console.log(event.code, event.data);
});
```

SDK.changeCustomPlayerStatus 方法

描述

修改指定房间的玩家自定义状态。

参数说明

参数名	类型	描述
changeCustomPlayerStatusPara	IChangeCustomPlayerStatusPara	请求参数
callback	ReqCallback<IChangeCustomPlayerStatusRsp>	回调函数

IChangeCustomPlayerStatusPara 定义如下：

属性名	类型/值	描述
roomId	string	房间 ID
playerId	string	玩家 ID
customPlayerStatus	number	玩家自定义状态

IChangeCustomPlayerStatusRsp 定义如下：

属性名	类型/值	描述
roomInfo	IRoomInfo	房间信息

返回值说明

无。

> 说明：
> 调用该接口需要在 mgobexsCode 配置正确的游戏 ID 和后端密钥。

使用示例

```
const changeCustomPlayerStatusPara = { roomId: "xxx", playerId: "xxx", customPlayerStatus: 1 };
SDK.changeCustomPlayerStatus(changeCustomPlayerStatusPara, event => {
console.log(event.code, event.data);
});
```

SDK.removePlayer 方法

描述

在指定房间踢除玩家。

参数说明

参数名	类型	描述
removePlayerPara	IRemovePlayerPara	请求参数
callback	ReqCallback<IRemovePlayerRsp>	回调函数

IRemovePlayerPara 定义如下：

属性名	类型/值	描述
roomId	string	房间 ID
removePlayerId	string	玩家 ID

IRemovePlayerRsp 定义如下：

属性名	类型/值	描述
roomInfo	IRoomInfo	房间信息

返回值说明

无。

> 说明：
> 调用该接口需要在 mgobexsCode 配置正确的游戏 ID 和后端密钥。

使用示例

```
const removePlayerPara = { roomId: "xxx", removePlayerId: "xxx" };
SDK.removePlayer(removePlayerPara, event => {
console.log(event.code, event.data);
});
```

错误码

最近更新时间：2019-10-31 16:50:16

客户端 SDK 错误码

错误码 KEY	值	描述
EC_SDK_SEND_FAIL	90001	消息发送失败
EC_SDK_UNINIT	90002	SDK 未初始化
EC_SDK_RES_TIMEOUT	90003	消息响应超时
EC_SDK_NO_LOGIN	90004	登录态错误
EC_SDK_NO_CHECK_LOGIN	90005	帧同步鉴权错误
EC_SDK_SOCKET_ERROR	90006	网络错误
EC_SDK_SOCKET_CLOSE	90007	Socket 断开
EC_SDK_NO_ROOM	90008	无房间

系统逻辑错误码

错误码 KEY	值	描述
EC_OK	0	返回成功
EC_REQ_BAD_PKG	1	请求包格式错误
EC_CMD_INVALID	2	非法命令字
EC_PARAMS_INVALID	3	参数错误
EC_INNER_ERROR	4	服务器内部错误
EC_TIME_OUT	5	后端超时错误
EC_SERVER_BUSY	6	服务器繁忙
EC_NO_RIGHT	7	没有权限请求
EC_ACCESS_CMD_INVALID_ERR	200	命令字无效错误
EC_ACCESS_CMD_GET_TOKEN_ERR	201	获取 Token 失败
EC_ACCESS_CMD_TOKEN_PRE_EXPIRE	202	Token 即将过期
EC_ACCESS_CMD_INVALID_TOKEN	203	Token 无效或过期
EC_ACCESS_PUSH_SERIALIZE_ERR	204	PUSH 序列化包失败
EC_ACCESS_LOGIN_BODY_PARSE_ERR	205	登录用户中心回包解析出错

错误码 KEY	值	描述
EC_ACCESS_CONN_ERR	206	查找连接信息出错
EC_ACCESS_GET_RS_IP_ERR	207	获取 Relay 的 RS_IP 或 RS_PORT 出错
EC_ACCESS_ADD_COMM_CONN_ERR	208	添加 COMM 连接信息失败
EC_ACCESS_ADD_HEART_CONN_ERR	209	添加心跳连接信息失败
EC_ACCESS_ADD_RELAY_CONN_ERR	210	添加 Relay 连接信息失败
EC_ACCESS_HEART_BODY_PARSE_ERR	211	心跳包解析出错

用户中心错误

错误码 KEY	值	描述
EC_PLAYER_GAME_NOT_EXIST	10000	game 不存在
EC_PLAYER_SECRET_KEY_FAIL	10001	查询 secret_key 失败
EC_PLAYER_SIGN_ERR	10002	sign校验失败
EC_PLAYER_DUPLICATE_REQ	10003	重复请求
EC_PLAYER_TIMESTAMP_INVALID	10004	timestamp 非法
EC_PLAYER_QUERY_PLAYER_FAIL	10005	查询用户信息失败
EC_PLAYER_ADD_PLAYER_FAIL	10006	新增用户信息失败
EC_PLAYER_QUERY_GAME_FAIL	10007	查询 game 信息失败
EC_PLAYER_RECORD_NUM_ERR	10008	用户记录数不正确
EC_PLAYER_GET_TOKEN_FAIL	10009	查询 Token 失败
EC_PLAYER_TOKEN_NOT_EXIST	10010	Token 不存在
EC_PLAYER_TOKEN_INVALID	10011	Token 非法
EC_PLAYER_CLEAR_TOKEN_FAIL	10012	清除 Token 缓存失败
EC_PLAYER_LOCK_FAIL	10013	获取分布式锁失败
EC_PLAYER_UNLOCK_FAIL	10014	释放分布式锁失败
EC_PLAYER_SAVE_TOKEN_FAIL	10015	保存 Token 缓存失败

房间管理类错误

错误码 KEY	值	描述
EC_ROOM_CREATE_NO_PERMISSION	20000	创建房间无权限

错误码 KEY	值	描述
EC_ROOM_DESTORY_NO_PERMISSION	20001	销毁房间无权限
EC_ROOM_JOIN_NO_PERMISSION	20002	无权限加入房间
EC_ROOM_REMOVE_PLAYER_NO_PERMISSION	20003	无踢人权限
EC_ROOM_MODIFY_PROPERTIES_NO_PEMISSION	20004	无修改房间属性权限
EC_ROOM_DISSMISS_NO_PERMISSION	20005	无解散房间权限
EC_ROOM_REMOVE_SELF_NO_PERMISSION	20006	无踢出自己权限
EC_ROOM_CHECK_LOGIN_SESSION_ERR	20007	检查登录失败
EC_ROOM_PLAYER_ALREADY_IN_ROOM	20010	用户已经在房间内，不能操作创建房间、加房等操作
EC_ROOM_PLAYER_NOT_IN_ROOM	20011	用户目前不在房间内，不能操作更改房间属性、踢人等操作
EC_ROOM_PLAYERS_EXCEED_LIMIT	20012	房间内用户数已经达到最大人数不能再加入了
EC_ROOM_JOIN_NOT_ALLOW	20013	房间不允许加入用户
EC_ROOM_MAX_PLAYERS_INVALID	20014	最大用户数值设置非法
EC_ROOM_CREATE_FAIL	20015	创建房间失败
EC_ROOM_PLAYER_OFFLINE	20016	用户在房间中掉线，不能开始游戏等操作
EC_ROOM_PARAM_PAGE_INVALID	20017	页号、页数大小参数不合法，可能实际大小没这么大
EC_ROOM_GET_PLAYER_INFO_ERR	20050	查询用户信息失败
EC_ROOM_GET_ROOM_INFO_ERR	20051	获取房间信息失败
EC_ROOM_MODIFY_OWNER_ERR	20052	修改房主失败
EC_ROOM_MAX_ROOM_NUMBER_EXCEED_LIMIT	20053	房间数量超过限制
EC_ROOM_QUERY_PLAYER_ERR	20060	查询用户信息失败
EC_ROOM_QUERY_GAME_ERR	20061	游戏信息失败
EC_ROOM_PLAYER_INFO_NOT_EXIST	20062	用户信息不存在
EC_ROOM_GAME_INFO_NOT_EXIST	20063	游戏信息不存在
EC_ROOM_REGION_INFO_NOT_EXIST	20065	查询不到 accessRegion 信息
EC_ROOM_QUERY_REGION_ERR	20066	查询地域信息失败
EC_ROOM_INFO_UNEXIST	20080	房间信息不存在
EC_ROOM_ALLOCATE_RELAYSVR_IP_PORT_ERR	20090	ctrlsvr 分配 relaysvr 失败
EC_ROOM_INVALID_PARAMS_TEAM_ID	20100	房间 teamId 无效

错误码 KEY	值	描述
EC_ROOM_TEAM_MEMBER_LIMIT_EXCEED	20101	房间团队人员已满

匹配服务类错误

错误码 KEY	值	描述
EC_MATCH_NO_ROOM	30000	匹配失败，无任何房间
EC_MATCH_TIMEOUT	30001	匹配超时
EC_MATCH_LOGIC_ERR	30002	匹配逻辑错误
EC_MATCH_ERR	30010	匹配失败
EC_MATCH_PLAYER_IS_IN_MATCH	30011	用户已经在匹配中
EC_MATCH_PLAYER_NOT_IN_MATCH	30012	用户不在匹配状态
EC_MATCH_GET_MATCH_INFO_ERR	30013	获取匹配信息失败
EC_MATCH_UPDATE_MATCH_INFO_ERR	30014	更新匹配信息失败
EC_MATCH_CANCEL_FAILED	30015	取消匹配失败
EC_MATCH_GET_PLAYER_LIST_INFO_ERR	30016	查询匹配队列信息失败
EC_MATCH_CREATE_ROOM_ERR	30041	匹配创建房间失败
EC_MATCH_JOIN_ROOM_ERR	30042	匹配加入房间失败
EC_MATCH_QUERY_PLAYER_ERR	30100	查询用户信息失败
EC_MATCH_PLAYER_INFO_NOT_EXIST	30101	用户信息不存在
EC_MATCH_QUERY_GAME_ERR	30102	查询游戏信息失败
EC_MATCH_GAME_INFO_NOT_EXIST	30103	游戏信息不存在
EC_MATCH_QUERY_REGION_ERR	30104	查询大区信息失败
EC_MATCH_REGION_INFO_NOT_EXIST	30105	无大区信息
EC_MATCH_TEAM_FAIL	30106	团队匹配失败
EC_MATCH_PLAY_RULE_NOT_RUNNING	30107	匹配规则不可用
EC_MATCH_PLAY_ATTR_NOT_FOUND	30108	匹配参数不完整
EC_MATCH_PLAY_RULE_NOT_FOUND	30109	匹配规则不存在
EC_MATCH_PLAY_RULE_ATTR_SEGMENT_NOT_FOUND	30110	匹配规则获取属性匹配区间失败
EC_MATCH_PLAY_RULE_FUNC_ERR	30111	匹配规则算法错误

错误码 KEY	值	描述
EC_MATCH_GET_PLAYER_ATTR_FAIL	30112	匹配获取玩家属性失败
EC_MATCH_GET_TEAM_ATTR_FAIL	30113	匹配获取队伍属性失败

帧同步服务类错误

错误码 KEY	值	描述
EC_RELAY_ALREADY_EXISTS	40000	重复创建
EC_RELAY_NOT_EXISTS	40001	服务不存在
EC_RELAY_DATA_EXCEED_LIMITED	40002	data 长度超限制
EC_RELAY_MEMBER_ALREADY_EXISTS	40003	成员已存在
EC_RELAY_MEMBER_NOT_EXISTS	40004	成员不存在
EC_RELAY_STATE_INVALID	40005	状态异常
EC_RELAY_INVALID_FRAME_RATE	40006	帧率非法
EC_RELAY_SET_FRAME_RATE_FORBIDDEN	40007	开局状态下，G 不允许修改帧率
EC_RELAY_NO_MEMBERS	40008	没任何成员
EC_RELAY_GAMESVR_SERVICE_NOT_OPEN	40009	自定义扩展服务（gamesvr）未开通
EC_RELAY_REQ_POD_FAIL	40010	请求分配 pod 失败
EC_RELAY_NO_AVAILABLE_POD	40011	无可用的 pod
EC_RELAY_GET_FRAME_CACHE_FAIL	40012	查询帧缓存失败
EC_RELAY_HKV_CACHE_ERROR	40015	共享内存缓存错误
EC_RELAY_REDIS_CACHE_ERROR	40016	redis 缓存错误
EC_RELAY_NOTIFY_RELAYWORKER_FAIL	40018	通知 relayworker 失败
EC_RELAY_RESET_RELAY_ROOM_FAIL	40019	重置房间对局失败
EC_RELAY_CLEAN_RELAY_ROOM_FAIL	40020	清理房间对局数据失败
EC_RELAY_NO_PERMISSION	40100	没权限，401开头是权限相关错误
EC_RELAY_NOTIFY_GAMESVR_FAIL	40200	通知自定义服务 gamesvr 失败， 402开头，是自定义 gamesvr 相关的错误
EC_RELAY_FORWARD_TO_GAMESVR_FAIL	40201	转发到自定义逻辑 svr 失败
EC_RELAY_FORWARD_TO_CLIENT_FAIL	40202	转发到 client-sdk 失败

参数错误

错误码 KEY	值	描述
EC_INVALID_PARAMS	60000	业务参数错误
EC_INVALID_PARAMS_PLAY_MODE_VERSION	60001	玩法协议版本号错误
EC_INVALID_PARAMS_PLAY_MODE_RULETYPE	60002	玩法协议规则类型错误
EC_INVALID_PARAMS_PLAY_MODE_EXPRESSION	60003	玩法协议规则表达式错误
EC_INVALID_PARAMS_PLAY_MODE_TEAM	60004	玩法协议规则团队表达式错误
EC_INVALID_PARAMS_GAME_ID	61000	参数错误 game_id
EC_INVALID_PARAMS_PLAYER_INFO	61001	参数错误 player_info
EC_INVALID_PARAMS_MAX_PLAYERS	61002	参数错误 max_players
EC_INVALID_PARAMS_ROOM_TYPE	61003	参数错误 room_type
EC_INVALID_PARAMS_PLAYER_ID	61004	参数错误 player_id
EC_INVALID_PARAMS_MATCH_TYPE	61005	参数错误 match_type
EC_INVALID_PARAMS_MATCH_CODE	61006	参数错误 match_code
EC_INVALID_PARAMS_OPEN_ID	61007	参数错误 open_id
EC_INVALID_PARAMS_PLATFORM	61008	参数错误 platform
EC_INVALID_PARAMS_TIMESTAMP	61009	参数错误 timestamp
EC_INVALID_PARAMS_SIGN	61010	参数错误 sign
EC_INVALID_PARAMS_NONCE	61011	参数错误 nonce
EC_INVALID_PARAMS_TOKEN	61012	参数错误 token
EC_INVALID_PARAMS_NETWORK_STATE	61013	参数错误 network_state
EC_INVALID_PARAMS_ROOM_NAME	61014	参数错误 room_name
EC_INVALID_PARAMS_CREATE_ROOM_TYPE	61015	参数错误 create_room_type
EC_INVALID_PARAMS_DEVICE_ID	61016	参数错误 device_id
EC_INVALID_PARAMS_PAGE_NO	61017	参数错误 page_no
EC_INVALID_PARAMS_PAGE_SIZE	61018	参数错误 page_size