功能实现-实时音视频通话

本节介绍如何使用Zego 微信小程序SDK和即构小程序直播插件实现实时音视频通话

1. 初始化客户端实例

集成 SDK 完成后，要想使用 SDK 的功能，还需要对 SDK 进行初始化操作。

```
// 声明变量
var ZegoSDK = require("../../../js/jZego-wx-1.4.0.js");
var zg;

// 初始化实例
zg = new ZegoSDK.ZegoClient();

// 配置必要参数
zg.config({
   appid: this.data.appID,         // 必填，应用id，由即构提供
   idName: this.data.userID,       // 必填，用户自定义id，全局唯一
   nickName: this.data.userName,   // 必填，用户自定义昵称
   remoteLogLevel: 2,              // 日志上传级别，建议取值不小于 logLevel
   logLevel: 0,                    // 日志级别，debug: 0, info: 1, warn: 2, error: 3, r
eport: 99, disable: 100（数字越大，日志越少）
   server: wsServerURL,               // 必填，服务器地址，由即构提供
   logUrl: logServerURL,          // 必填，log 服务器地址，由即构提供
   audienceCreateRoom: false,      // false观众不允许创建房间
});
```

初始化 SDK 时**必须**要做的事情：

设置 idName 和 nickName。

> 请注意：
>
> idName 和 nickName 可自定义，每个 idName 必须唯一，idName只支持长度不超过 64 byte 的数字，下划线，字母；userName 只支持长度不超过 256 byte 的数字，下划线，字母。

> 如果要实现一个完善的直播功能，就需要处理 SDK 的相关回调。 **注意：回调只要在 SDK 生命周期内设置一次即可。**

2. 登录房间

2.1 设置房间相关的回调监听

登录房间之前需要设置房间相关的回调监听，以在成功登录房间后接收房间相关的事件通知，比如处理因网络中断退出房间等问题。 例如：

```
ZegoClient.onDisconnect = function(err) {
 //处理回调逻辑.....
}
```

> 其余回调接口请根据业务实际情况进行选择处理，完整的房间回调接口请查看 回调接口

2.2 获取登录 token

在开发阶段，即构提供了只用于测试环境获取token的接口，但正式上线一定要由开发者的业务服务器实现token逻辑；

登录 token 的获取详见：房间登录安全。

Demo中演示源码片段如下：

```
/* 即构提供开发阶段获取token接口:https://wsliveroom-alpha.zego.im:8282/token, 只能用于测试环境, 正式环境一定要由客户业务服务器实现token
*/

// 获取登录 token
getLoginToken: function () {
    var self = this;
    const requestTask = wx.request({
      url: 'xxxx', // 该接口由开发者后台自行实现，开发者的 Token 从各自后台获取
      data: {
        app_id: self.data.appID,
        id_name: self.data.userID,
      },
```

```
      header: {
        'content-type': 'text/plain'
      },
      success: function (res) {
        console.log(">>>[liveroom-room] get login token success. token is: " + res.dat
a);
        if (res.statusCode != 200) {
          return;
        }

        zg.setUserStateUpdate(true);
        self.loginRoom(res.data, self);
      },
      fail: function (e) {
        console.log(">>>[liveroom-room] get login token fail, error is: ")
        console.log(e);
      }
    });
},
```

```
/**调用 `login` 登录房间

**注意：需保证 `roomID` 信息的全局唯一，只支持长度不超过 128 字节 的数字，下划线，字
母。

登录房间成功是后续所有操作的前提。小程序中演示源码片段如下，仅供参考：
**/
zg.login(self.data.roomID, self.data.loginType == "anchor" ? 1 : 2, token, function (s
treamList) {
    // 登录成功处理
    console.log('>>>[liveroom-room] login succeeded');
}, function (err) {
    // 登录失败处理
    console.log('>>>[liveroom-room] login failed, error is: ', err);
});
```

3 推流

直播过程中，如果需要推送自己的画面，都需要进行推流操作。

3.1 组件说明

微信小程序中的推流功能，需要用到即构“小程序直播插件”提供的 `zego-pusher` 标签。

demo 中，创建 zego-pusher 的演示源码如下：

```
<zego-pusher
        id="zg-pusher"
        url="{{pusherInfo.url}}"
        class="push-content"
        waitingImage="{{waitingImage}}"
        enableCamera="{{enableCamera}}"
        debug="{{debug}}"
        autoFocus="{{autoFocus}}"
        aspect="{{aspect}}"
        minBitrate="{{minBitrate}}"
        maxBitrate="{{maxBitrate}}"
        zoom="{{zoom}}"
        mode="{{mode}}"
        muted="{{muted}}"
        beauty="{{beauty}}"
        whiteness="{{whiteness}}"
        orientation="{{orientation}}"
        bindstatechange="onPushStateChange"
        bindnetstatus="onPushNetStateChange">
</zego-pusher>
```

3.2 开始推流

主播登录房间成功后，根据业务逻辑准备推流。使用 SDK 推流播放需要遵循如下步骤：

1. 触发推流
2. 调用 SDK 的 `startPublishingStream` 获取 streamID 对应的推流地址
3. 在 SDK 的回调 `onStreamUrlUpdate` 中获推流地址
4. 将步骤 3 中获取的推流地址设置为 `zego-pusher` 的 `url`
5. 获取推流组件实例，然后调用实例 的 `start()` 录制视频

演示源码片段如下，仅供参考：

```
// 1/2. 主播登录房间成功后触发推流，调用 SDK 的 startPublishingStream 获取 streamID 对应的推流地址
zg.login(self.data.roomID, self.data.loginType == "anchor" ? 1 : 2, token, function (streamList) {
```

```
  // 主播登录成功即推流
  if (self.data.loginType == 'anchor') {
    console.log('>>>[liveroom-room] anchor startPublishingStream, publishStreamID: ' +
self.data.publishStreamID);

    zg.setPreferPublishSourceType(1); // 0：推流到 CDN，观众拉流延迟在 2 秒左右；1：推
流到 ZEGO 服务器，延迟在 400ms 左右
    zg.startPublishingStream(self.data.publishStreamID);
  }
}, function (err) {
  console.log('>>>[liveroom-room] login failed, error is: ', err);
});
```

```
// 3. 在 SDK 的回调 onStreamUrlUpdate 中获取推流地址
// type: {play: 0, publish: 1};
zg.onStreamUrlUpdate = function (streamid, url, type) {
    console.log(">>>[liveroom-room] zg onStreamUrlUpdate, streamId: " + streamid + ',
 type: ' + (type == 0 ? 'play' : 'publish') + ', url: ' + url);

    ...
};
```

```
zgPusher = this.selectComponent("#zg-pusher");
zgPusher.start();
```

3.3 推流事件处理

微信小程序会在 `zego-pusher` 的 `bindstatechange` 绑定的方法中通知出**推流状态事件**，开发者需要：

1. 在 `bindstatechange` 绑定的回调函数中，调用 SDK 提供的 API `updatePlayerState` 将推流事件透传给 SDK
2. 在 SDK 提供的 `publisherStateUpdate` 回调中处理推流的开始、失败状态

演示源码片段如下，仅供参考：

```
// zego-pusher 绑定推流事件
onPushStateChange(e) {
    console.log(
```

```
        `${TAG_NAME} onPushStateChange `,
        e.detail.code ,
        e.detail.message
      );
      zg.updatePlayerState(this.data.pusherInfo.streamID, e);
  },

  // 推流后，服务器主动推过来的，流状态更新；type: { start: 0, stop: 1 }，主动停止推流没有回调，其他情况均回调
  zg.onPublishStateUpdate = (type, streamID, error) => {
        console.warn(TAG_NAME, 'onPublishStateUpdate', type, streamID, error);
        this.setData({
          publishing: type === 0 ? true : false,
          beginToPush: false
        })
        ...
  }
```

另外，小程序会在 `zego-pusher` 的 `bindnetstatus` 绑定的方法中通知出**推流网络事件**，开发者也需要在对应的小程序回调中，调用 `updatePlayerNetStatus` 将推流事件透传给 SDK。

演示源码片段如下，仅供参考：

```
  // zego-pusher 绑定网络状态事件
  onPushNetStateChange(e) {
      console.log(
        `${TAG_NAME} onPushNetStateChange `,
        e.detail.code ,
        e.detail.message
      );
      zg.updatePlayerNetStatus(this.data.pusherInfo.streamID, e);
  },

  // SDK 获取推流网络质量
  zg.onPublishQualityUpdate = (streamID, streamQuality) => {
        console.log(`${TAG_NAME}  onPublishQualityUpdate ${streamID}`, streamQuality);
  };
```

3.4 停止推流

停止推流，开发者需要：

1. 调用 SDK 提供的 `stopPublishingStream(streamID)` 清空推流状态

2. 调用 `zego-pusher` 实例提供的 `stop()` 停止推流

> 请注意，上述第 1 点一定要处理，否则可能导致 SDK 状态异常！

演示源码片段如下，仅供参考：

```
// 停止推流
zg.stopPublishingStream (this.data.pushStreamID);
zgPusher.stop();
```

4 拉流

直播过程中，如果想观看房间内其他成员的推流画面，都需要进行拉流操作。

4.1 组件说明

微信小程序中的拉流流功能，需要用到插件提供的 `zego-player` 标签。

demo中，创建 zego-player 的演示源码如下：

```
<zego-player
        id="zg-player"
        sid="{{playerInfo.streamID}}"
        url="{{playerInfo.url}}"
        orientation="{{orientation}}"
        objectFit="{{objectFit}}"
        minCache="{{minCache}}"
        maxCache="{{maxCache}}"
        mode="{{mode}}"
        muted="{{muted}}"
        debug="{{debug}}"
        pictureInPictureMode="{{pictureInPictureMode}}"
        objectFit="{{objectFit}}"
        class="play-content"
        bindstatechange="onPlayStateChange"
        bindnetstatus="onPlayNetStateChange">
</zego-player>
```

4.2 开始拉流

观众登录房间成功后，根据业务逻辑准备拉流。使用 SDK 拉流播放需要遵循如下步骤：

1. 触发拉流
2. 调用 SDK 的 `startPlayingStream` 获取 streamID 对应的播放地址
3. 在 SDK 的回调 `onStreamUrlUpdate` 中获取拉流地址
4. 将步骤 3 中获取的推流地址设置为 `zego-player` 的 `url`，流ID设置为`sid`
5. 获取拉流组件实例，然后调用实例 的 `play()` 播放视频 或者设置拉流组件的autoplay属性为true，实现自动拉流。

演示源码片段如下，仅供参考：

```
// 1/2. 通过 SDK 获取 streamID 对应的播放地址
zg.startPlayingStream(streamList[0].stream_id);
```

```
// 3. 在 SDK 的回调 onStreamUrlUpdate 中获取拉流地址
// type: {play: 0, publish: 1};
zg.onStreamUrlUpdate = function (streamid, url, type) {
    console.log(`${TAG_NAME} onStreamUrlUpdate ${streamID} ${type === 0 ? 'play' : 'pu
blish'} ${url}`);
    ...
};
```

```
zgPlayer = this.selectComponent("#zg-player");
```

```
zgPlayer.play();
```

4.3 拉流事件处理

微信小程序会在 `zego-player` 的 `bindstatechange` 绑定的方法中通知出**拉流状态事件**，开发者需要：

1. 在 `bindstatechange` 绑定的回调函数中，调用 SDK 提供的 API `updatePlayerState` 将推流事件透传给 SDK
2. 在 SDK 提供的 `onPlayStateUpdate` 回调中处理播推、拉流的开始、失败状态

演示源码片段如下，仅供参考：

```
// zego-player 绑定的拉流事件
onPlayStateChange(e) {
    // 透传拉流事件给 SDK, type 0 拉流
    zg.updatePlayerState(e.detail.streamID, e);
},

// 服务端主动推过来的 流的播放状态, 视频播放状态通知；type: { start:0, stop:1};
zg.onPlayStateUpdate = function (updatedType, streamID) {
    console.log(`${TAG_NAME} onPlayStateUpdate ${updatedType === 0 ? 'start ' : 'stop
 '} ${streamID}`);
};
```

另外，小程序会在 `zego-player` 的 `bindnetstatus` 绑定的方法中通知出**拉流网络事件**，开发者也需要在对应的小程序回调中，调用 `updatePlayerNetStatus` 将推流事件透传给 SDK。

演示源码片段如下，仅供参考：

```
// zego-player 绑定网络状态事件
onPlayNetStateChange(e) {
    console.log(
      `${TAG_NAME} onPlayNetStateChange `,
      e.detail.info
    );
    zg.updatePlayerNetStatus(e.detail.streamID, e);
},

// SDK 拉流网络质量回调
zg.onPlayQualityUpdate = (streamID, streamQuality) => {
      console.log(`${TAG_NAME} onPlayStateUpdate ${streamID}`, streamQuality);
};
```

4.4 停止拉流

停止拉流，开发者需要：

1. 调用 SDK 提供的 `stopPlayingStream(streamid)` 清空拉流状态
2. 调用 `zego-player` 提供的 `stop()` 停止推流

> 请注意，上述第 1 点一定要处理，否则可能导致 SDK 状态异常！

演示源码片段如下，仅供参考：

```
// 停止拉流
zg.stopPlayingStream(this.data.playInfo.streamID);
zgPlayer.stop();
```

5 退出房间

调用如下 API 退出房间。**请开发者在退出房间前，确保停止推拉流，并清理相关状态。**

示例代码如下：

```
zg.logout();
```

6 微信公众平台域名配置

ZEGO 分配给开发者的 URL（包含 HTTPS、WSS 协议），需要在微信公众平台进行“合法域名”配置后，小程序才能正常访问。

微信后台配置地址：`微信公众平台 -> 设置 -> 开发设置 -> 服务器域名`。

请开发者将 ZEGO 分配的请求域名，按照协议分类，填到指定的 `request合法域名` 或者 `socket合法域名` 中。例如：

服务器域名

服务器配置		说明	操作
request合法域名	https://[redacted] https://[redacted] https://[redacted] https://[redacted] https://[redacted] https://[redacted] https://[redacted]	一个月内可申请5次修改 本月还可修改5次	修改
socket合法域名	wss://[redacted] wss://[redacted] wss://[redacted]		
uploadFile合法域名			
downloadFile合法域名			