

个验SDK API接口

本文档仅适用于个验3.0.0.0及以上版本。

3.0.0.0以前的版本、或者从老版本升级上来、或者仍旧使用老版本的一键登录接口，请查看[《升级指南》](#)

1. 结果回调

接口说明

大部分gysdk接口，执行的结果都通过对应接口中传入的gyCallBack进行回调通知。

```
public interface GyCallBack {  
    //成功回调  
    void onSuccess(GYResponse response);  
    //失败回调  
    void onFailed(GYResponse response);  
}
```

参数说明

```
public class GYResponse {  
    /* 个验用户唯一标识  
    */  
    public String getGyuid();  
  
    /* 接口请求是否成功  
    */  
    public boolean isSuccess();  
  
    /* code 返回结果，30000为成功，详见状态码说明，代码引用可参考 GyCode  
    */  
    public int getCode();  
  
    /* msg 返回结果详情，为json格式，包含errorCode、errorDesc、metadata、及其他业务自定义的字段  
    * - errorCode 错误码，0为成功，详见状态码说明，代码引用可参考 GyErrorCode  
    * - errorDesc 错误描述  
    * - metadata 原始信息，比如运营商/服务器的原始返回信息、异常出错信息等等  
    */  
    public String getMsg();  
  
    /* 请求的运营商，CM 移动、CT 电信、CU 联通  
    */  
    public String getOperator();  
}
```

调用示例

```
// 请在预登录成功之后、设置好privacyTv的内容之后再调用eAccountLogin
GYManager.getInstance().eAccountLogin(eloginActivityParam, 5000, new GyCallBack() {
    @Override
    public void onSuccess(GYResponse response) {
        Log.d(TAG, "登录成功 response:" + response);
        try {
            //业务成功，从msg中解析出特定的业务字段
            JSONObject jsonObject = new JSONObject(response.getMsg());
            JSONObject data = jsonObject.getJSONObject("data");
            String token = data.getString("token");
            long expiredTime = data.getLong("expiredTime");
            Log.d(TAG, "token:" + token + " expiredTime:" + expiredTime);
            //将token汇报服务端进行取号...
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onFailed(GYResponse response) {
        Log.e(TAG, "登录失败 response:" + response);
        try {
            //业务失败，从msg中解析出errorCode、errorDesc了解详细错误原因
            JSONObject jsonObject = new JSONObject(response.getMsg());
            int errorCode = jsonObject.getInt("errorCode");
            //...
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
```

返回示例

- 返回成功，解析对应业务的自定义字段
- 返回失败，解析 msg.errorCode、msg.errorDesc 了解详细失败原因

```
初始化成功 response:GYResponse{
    gyuid='gtc_ea0a6ae25e99358d9a99bc5ed342893838',
    success=true,
    code=30000,
    operator=,
    msg='{ "errorCode":0,"errorDesc":"gysdk success!","metadata":""}'
}
```

```
初始化失败 response:GYResponse{
    gyuid='',
```

```
success=false,
code=30004,
operator=,
msg='{ "errorCode":-20101,"errorDesc":"gysdk appid为空!","metadata":""}'
}
```

2. 初始化

接口说明

建议在Application.onCreate()中判断用户已经同意隐私政策后初始化gysdk。

```
/**
 * 初始化sdk接口。
 * @param context 上下文。
 * @param gyCallBack 成功/失败的回调，有且只有一次回调。
 */
public void init(Context context, GyCallBack gyCallBack);
```

调用示例

```
GYManager.getInstance().init(this.getApplicationContext(), new GyCallBack() {
    @Override
    public void onSuccess(GYResponse response) {
        Log.d(TAG, "初始化成功 response:" + response);
    }

    @Override
    public void onFailed(GYResponse response) {
        Log.e(TAG, "初始化失败 response:" + response);
    }
});
```

返回示例

- 返回成功，放心使用
- 返回失败，解析 msg.errorCode、msg.errorDesc 了解详细失败原因

```
初始化成功 response:GYResponse{
  gyuid='gtc_ea0a6ae25e99358d9a99bc5ed342893838',
  success=true,
  code=30000,
  operator=,
  msg='{ "errorCode":0,"errorDesc":"gysdk success!","metadata":""}'
}
```

```
初始化失败 response:GYResponse{
```

```
gyuid='',
success=false,
code=30004,
operator=,
msg='{ "errorCode":-20101,"errorDesc":"gysdk appid为空!","metadata":""}'
}
```

3. 调试模式

接口说明

默认gysdk除了少量重大错误，基本不输出日志打印。调试阶段请开启日志打印，一般发布版本中关闭，以免影响性能。

```
/**
 * 开发者日志输出。一般发布版本中关闭，以免影响性能。
 * @param isDebug 日志开关。
 */
public void setDebug(boolean isDebug);
```

调用示例

```
GYManager.getInstance().setDebug(true);
```

4. 设置渠道

接口说明

用于代码设置渠道，优先级高于build.gradle中配置的GT_INSTALL_CHANNEL。为避免没有及时生效，请在gysdk的init之前调用。

```
/**
 * 设置渠道号，需要在init之后调用才能生效
 * @param channel
 */
public void setChannel(String channel)
```

调用示例

```
GYManager.getInstance().setChannel("channel");
```

5. 版本信息

接口说明

获取gysdk版本信息。

```
/**
 * 获取版本号.
 * @return version 版本号.
 */
public String getVersion();
```

调用示例

```
GYManager.getInstance().getVersion();
```

6. 预登录

接口说明

在调用一键登录接口之前，必须先预登录，预登录成功之后才可以一键登录。

```
/**
 * 预登录，每次调用都会有且只有一次回调。连续多次调用，后面的会等第一次完成后返回相同的结果。调用
init紧接着就可以调用ePreLogin，sdk内部会自行等待init完成再预登录。
 * @param timeout 超时时间，一般5000~15000ms，<=0则使用15s，建议为5000ms以上（在不影响用户体验的地方比如Application.onCreate中调用，建议设置的更长一些比如8000ms）
 * @param gyCallback 回调.
 */
public void ePreLogin(int timeout, GyCallback gyCallback);
```

注意：

- 预登录是一个耗时操作，我们建议开发者提前进行预登录，例如在Application.onCreate初始化gysdk之后、闪屏页面或者其他任何地方，这样可以在调用一键登录的时候节省用户等待的时间；
- 预登录成功表示当前设备可以使用一键登录，失败则表示不可用，开发者在失败的时候可以进行重试、或者换成其他登录方式；
- 预登录的结果在一定时间内有效，具体的有效期会在预登录成功的结果里面返回（一般移动通信一小时有效，联通半小时）；预登录的结果只能用于一次一键登录，一键登录成功之后需要再次进行预登录；
- SDK提供了 `isPreLoginResultValid` 接口来帮助开发者判断当前的预登录结果是否有效。

调用示例

```
if (GYManager.getInstance().isPreLoginResultValid()) {
    eLogin();//预登录有效，启动登录授权页
} else {
    GYManager.getInstance().ePreLogin(5000, new GyCallback() {
        @Override
```

```

    public void onSuccess(GYResponse response) {
        Log.d(TAG, "预登录成功 response:" + response);
        eLogin();//预登录成功, 启动登录授权页
    }
    @Override
    public void onFailed(GYResponse response) {
        Log.d(TAG, "预登录失败 response:" + response);
    }
};

```

返回示例

- 返回成功, 调用登录
- 返回失败, 解析msg.errorCode、msg.errorDesc了解详细失败原因

```

预登录成功 response:GYResponse{
    gyuid='gtc_ea0a6ae25e99358d9a99bc5ed342893838',
    success=true,
    code=30000,
    operator=CM,
    msg='{
        "process_id":"1b094917c1d34bef3475e61e60d652e2",
        "operatorType":"1",
        "clienttype":"1",
        "number":"135****9774",
        "expiredTime":1631515911017,
        "errorCode":0,
        "errorDesc":"gysdk success!"
    }'
}

预登录失败 response:GYResponse{
    gyuid='gtc_ea0a6ae25e99358d9a99bc5ed342893838',
    success=false,
    code=30005,
    operator=,
    msg='{
        "errno":0,
        "data":{"msg":"应用无效, 应用不存在","result":"40004"},
        "errorCode":-20102,
        "errorDesc":"gysdk appid或者签名无效!"
    }'
}

```

7. 预登录是否有效

接口说明

在再次预登录、或者一键登录之前判断是否已经有效预登录。

```
/**
 * 判断预登录结果是否有效。
 */
public boolean isPreLoginResultValid();
```

调用示例

```
//见上面“预登录”调用示例
```

8. 预登录信息

接口说明

授权页面需要设置的运营商品牌露出、和隐私协议标题和网址。

```
/**
 * 获取预登录信息。
 */
public GyPreloginResult getPreLoginResult();
```

参数说明

```
public class GyPreloginResult {
    /* 预登录是否有效，结果同isPreLoginResultValid
    */
    public boolean isValid();

    /* 运营商隐私协议标题，比如"中国移动认证服务条款"
    */
    public String getPrivacyName();

    /* 运营商隐私协议地址
    */
    public String getPrivacyUrl();

    /* 预登录的运营商，CM 移动，CT 电信，CU 联通
    */
    public String getOperator();
}
```

调用示例

```
//见下面“一键登录”调用示例
```

9. 一键登录

接口说明

使用EloginActivityParam参数的新登录接口，完全由开发者自行设计、启动授权页activity，然后在activity的onCreate中，设置好隐私协议、调用一键登录接口。

```
/** 一键登录，app启动activity、预登录成功、设置好隐私协议之后调用
 * sdk需要对登录按钮设置onClickListener，请用activityParam.setLoginOnClickListener设置自定义的监听
 * @param activityParam 授权页activity和必要元素，以及页面相关的回调
 * @param timeout 一键登录请求运营商的超时时间，一般5000~15000ms，<=0则使用15s，建议设为5000ms
 * @param gyCallback 结果回调，请在回调中finish授权页activity。
 */
public void eAccountLogin(EloginActivityParam activityParam, int timeout, GyCallback gyCallback)
```

注意：

- 一般是在进入授权页activity前，先判断预登录有效/无效时候预登录返回成功后，再调起授权页activity，再在其onCreate中设置好隐私协议、调用一键登录接口；
- 上面启动授权页和预登录的顺序也可根据需要进行对换，先启动授权页activity，然后在activity中判断预登录有效后/进行预登录成功后，再设置隐私协议、调用一键登录接口；
- 登录成功会回调成功，登录失败或者监听到activity被销毁或者再次用同一个activity调用一键登录都会回调失。
- 请务必严格遵循《[授权页规范](#)》**，设置必要元素：号码栏(NumberTextview)，品牌露出(SloganTextview)，登录按钮(LoginButton)，隐私确认(PrivacyCheckbox)，隐私标题(PrivacyTextview)；
- 如果授权页的必要组件不符合授权页规范的，会无法一键登录，打印日志：“UI不合规不能登录”。不合规行为比如：不可见、遮挡、修改内容等；
- sdk需要对登录按钮设置onClickListener，请用activityParam.setLoginOnClickListener设置自定义的监听。

参数说明

```
public class EloginActivityParam {
    /** 授权页activity，必填项
     */
    public EloginActivityParam setActivity(Activity activity);

    /** 号码栏，必填项
     */
    public EloginActivityParam setNumberTextview(TextView numberTextview);
```



```

/* 运营商品品牌露出，必填项
 */
public EloginActivityParam setSloganTextview(TextView sloganTextview);

/* 登录按钮，必填项
 */
public EloginActivityParam setLoginButton(View loginButton);

/* 隐私确认框，必填项
 */
public EloginActivityParam setPrivacyCheckbox(CheckBox privacyCheckbox);

/* 隐私协议，必填项
 */
public EloginActivityParam setPrivacyTextview(TextView privacyTextview);

/* 按钮自定义回调，非必填。
 * loginOnClickListener中抛出异常可中断SDK内的后续登录流程
 */
public EloginActivityParam setLoginOnClickListener(View.OnClickListener
loginOnClickListener);

/* 授权页错误回调，建议项。
 * 隐私协议未打勾、界面不合规、setLoginOnClickListener抛出异常等情况下的回调
 */
public EloginActivityParam setUiErrorListener(UIErrorListener uiErrorListener);
}

```

调用示例

```

//在activity onCreate或者预登录成功后设置隐私协议内容
GyPreloginResult preLoginResult = GYManager.getInstance().getPreLoginResult();
textView.setText("");
textView.append("登录即认可");
textView.append(generateSpan(preLoginResult.getPrivacyName(),
preLoginResult.getPrivacyUrl()));
textView.append("并使用本机号码登录");

//然后调用一键登录接口，设置必要元素、回调监听器
EloginActivityParam eloginActivityParam = new EloginActivityParam()
    .setActivity(this)
    .setNumberTextview(numberTv)
    .setNumberTextview(numberTv)
    .setSloganTextview(sloganTv)
    .setLoginButton(loginBtn)
    .setPrivacyCheckbox(checkBox)
    .setPrivacyTextview(privacyTv)
    .setUiErrorListener(new EloginActivityParam.UIErrorListener() {

```

```

@Override
public void onError(String msg) {
    //隐私协议未打勾、界面不合规、setLoginOnClickListener抛出异常等情况下的回调
    Log.e(TAG, "UIErrorListener.onError:" + msg);
    hideLoadingDialog();
}
})
.setLoginOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Log.d(TAG, "一键登录按钮 onLoginClick:");
        if (!mCheckBox.isChecked()) {
            showToast("请先仔细阅读协议并勾选，然后再点击登录");
            // 抛出异常，避免sdk进行后续登录动作（否则eAccountLogin会回调onFailed错误）
            throw new IllegalStateException("请先仔细阅读协议并勾选，然后再点击登录");
        }
        //启动登录时候的转圈圈
        showLoadingDialog();
    }
});
GYManager.getInstance().eAccountLogin(eloginActivityParam, 5000, new GyCallBack() {
    @Override
    public void onSuccess(GYResponse response) {
        Log.d(TAG, "登录成功 response:" + response);
        showToast("登录成功");
        //停止登录时候的转圈圈
        hideLoadingDialog();
        //关闭一键登录界面
        finish();

        try {
            //登录成功解析msg.data.token拿出token，用于服务端gy_get_pn置换手机号
            JSONObject jsonObject = new JSONObject(response.getMsg());
            JSONObject data = jsonObject.getJSONObject("data");
            String token = data.getString("token");
            long expiredTime = data.getLong("expiredTime");
            Log.d(TAG, "token:" + token + " expiredTime:" + expiredTime);
            //将token汇报服务端进行取号
            // ....由开发者自行实现
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});

@Override
public void onFailed(GYResponse response) {
    Log.e(TAG, "登录失败 response:" + response);
    showToast("一键登录失败:" + response);
    //停止登录时候的转圈圈

```

```

        hideLoadingDialog();
        //关闭一键登录界面
        finish();
    }
});

```

返回示例

- 返回成功，解析msg.data.token用于置换手机号
- 返回失败，解析 msg.errorCode、msg.errorDesc 了解详细失败原因

```

登录成功 response:GYResponse{
    gyuid='gtc_ea0a6ae25e99358d9a99bc5ed342893838',
    success=true,
    code=30000,
    operator=CM,
    msg='{
        "process_id":"66ba3f92c9ef8640c22434fff576e895",
        "operatorType":"1",
        "clienttype":"1",
        "data": {

"token": "U1RzaWQwMDAwMDAxNjMxNTA0MzY2NTk3bTJ5TXZzOW7ubndvdHp3MHhzNFVKRzR0Q4NsSnZxNVJ8fD
F8djJ8MQ==",
            "expiredTime":1631504487552
        },
        "errorCode":0,
        "errorDesc":"gysdk success!"
    }'
}

登录失败 response:GYResponse{
    gyuid='gtc_ea0a6ae25e99358d9a99bc5ed342893838',
    success=false,
    code=30006,
    operator=CM,
    msg='{
        "metadata":{"resultCode":"200022","authType":"0","authTypeDes":"其
他","resultDesc":"未检测到网络","traceId":"0c97ca4df88b4aebadaf890c8ba86047"},
        "process_id":"312ac1fdd7ccf8ef681478e7c81b4d82",
        "operatorType":"1",
        "clienttype":"1",
        "errorCode":-20200,
        "errorDesc":"gysdk无网络可用!"
    }'
}

```

10. 本机号码校验token

接口说明

本机号码校验需要先获取通行证，然后再进行本机号码校验。

```
/**
 * 获取本机号码校验通行证。
 *
 * @param phone      手机号码。
 * @param timeout     超时时间，一般5000~15000ms，<=0则使用15s，建议设为5000ms
 * @param gyCallback 回调。
 */
public void getVerifyToken(String phone, int timeout, GyCallback gyCallback);
```

调用示例

```
GYManager.getInstance().getVerifyToken(phone, 5000, new GyCallback() {
    @Override
    public void onSuccess(final GYResponse response) {
        Log.d(TAG, "号码校验token成功 response:" + response);
        try {
            JSONObject jsonObject = new JSONObject(response.getMsg());
            String accessCode = jsonObject.getString("accessCode");
            String processId = jsonObject.getString("process_id");
            int operatorType = jsonObject.getInt("operatorType");
            String phone = jsonObject.getString("phone");
        } catch (Throwable t) {
            t.printStackTrace();
        }
    }

    @Override
    public void onFailed(GYResponse response) {
        Log.d(TAG, "号码校验token失败 response:" + response);
        showToast(response.toString());
    }
});
```

返回示例

- 返回成功，解析msg.accessCode、process_id、operatorType进行本机号码校验
- 返回失败，解析 msg.errorCode、msg.errorDesc 了解详细失败原因

```
号码校验token成功 response:GYResponse{
  gyuid='gtc_ea0a6ae25e99358d9a99bc5ed342893838',
  success=true,
  code=30000,
  operator=CM,
```

```

msg= '{
    "process_id": "824db39a7eae7668bc5294962e41248",
    "operatorType": "1", "clienttype": "1",
    "accessCode": "STsid0000001631515224169YtuuyMbK6VawRNwwA0uWn3w9VTXnRclX",
    "phone": "13577459884",
    "expiredTime": 1631515335073,
    "errorCode": 0,
    "errorDesc": "gysdk success!"
}'
}

号码校验token失败 response:GYResponse{
    gyuid='gtc_ea0a6ae25e99358d9a99bc5ed342893838',
    success=false,
    code=30007,
    operator=CM,
    msg= '{
        "metadata":{"resultCode":"200022","authType":"0","authTypeDes":"其他",
        "resultDesc":"未检测到网络","traceId":"e57a2786c61a4ba2be135dacf7eb0b05"},
        "process_id": "034217796199557790c562bfb300177c",
        "operatorType": "1",
        "clienttype": "1",
        "errorCode": -20200,
        "errorDesc": "gysdk无网络可用!"
    }'
}

```

11. 本机号码校验

接口说明

我们还提供了服务端的本机号码校验接口，如果开发者有这方面的需求，请参考服务端的文档。

```

/**
 * 验证本机号码。
 *
 * @param accessCode 本机号码校验通行证。
 * @param processId processId。
 * @param phone 待校验号码。
 * @param operatorType 校验类型。
 * @param gyCallback 回调。
 */
public void verifyPhoneNumber(String accessCode, String processId, String phone,
int operatorType, GyCallback gyCallback);

```

调用示例

```
GYManager.getInstance().verifyPhoneNumber(accessCode, processId, phone, type, new
GyCallBack() {
    @Override
    public void onSuccess(GYResponse response) {
        Log.d(TAG, "号码校验成功 response:" + response);
        verifyResult(true, phone, response);
    }

    @Override
    public void onFailed(GYResponse response) {
        Log.d(TAG, "号码校验失败 response:" + response);
        verifyResult(false, phone, response);
    }
});
```

返回示例

- 返回成功，本机号码校验通过
- 返回失败，解析 msg.errorCode、msg.errorDesc 了解详细失败原因

```
号码校验成功 response:GYResponse{
  gyuid='gtc_ea0a6ae25e99358d9a99bc5ed342893838',
  success=true,
  code=30000,
  operator=CM,
  msg='{
    "process_id":"824db39a7eae7667bc5294962e41248",
    "operatorType":"1",
    "clienttype":"1",
    "errorCode":0,
    "errorDesc":"gysdk success!"
  }'
```

```
号码校验失败 response:GYResponse{
  gyuid='gtc_ea0a6ae25e99358d9a99bc5ed342893838',
  success=false,
  code=30008,
  operator=CM,
  msg='{
    "metadata":{"errno":0,"data":{"msg":"手机号校验失败","result":"40053"}},
    "process_id":"4a0dc562853c1d5f06dd65c56695b4e8",
    "operatorType":"1",
    "clienttype":"1",
    "errorCode":-30001,
    "errorDesc":"服务器返回错误，具体见msg!"
  }'
```

