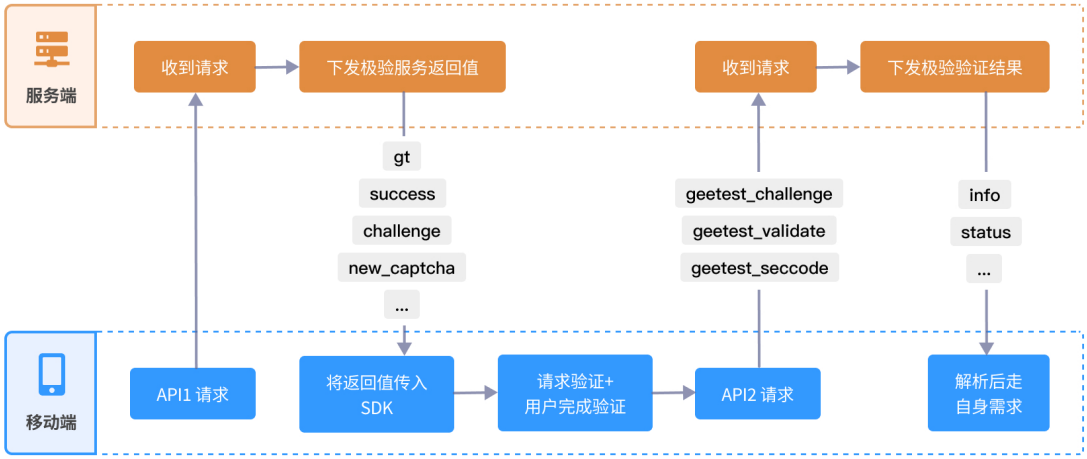
自定义 API 集成方式说明

什么是自定义 API

自定义 API 指客户移动端自行完成与客户服务端交互的过程，此请求可在返回极验服务字段的同时封装客户需求字段，完成客户两端交互需求。

流程图



注：API1 下发的四个字段为极验服务返回，若无需求请勿修改。API2 的请求参数 SDK 已封装，无需自行定义。

字段说明

API1 请求常规下发值如下 (SDK唯一识别以下字段对应类型，若有修改请解析后按以下格式还原再传入 SDK)

字段名	字段类型	字段说明
success	整型	宕机字段标识，0 为宕机，1 为正常
challenge	字符串	验证事件流水号
gt	字符串	客户唯一标识
new_captcha	布尔值	是否为新版验证，true 为是，false 为否

API2 请求必传参数如下 (默认 SDK 已封装，也可以根据需求自己构造)

字段名	字段类型
geetest_challenge	字符串
geetest_validate	字符串
geetest_seccode	字符串

API2 响应示例中的参数

字段名	字段类型	字段说明
status	字符串	验证结果，success 为成功
info	字符串	附加信息

代码示例

Android 端

自定义 API1，请求后返回值通过 `setApi1Json` 方法传入 SDK，通过 `getGeetest` 方法继续调起验证，示例如下，更多详情可参考demo。

```
/**
 * 自定义 API1 回调
 */
@Override
public void onButtonClick() {
    // 开启自定义 API1 逻辑
    new RequestAPI1().execute();
}
```

```
/**
 * 请求api1
 */
class RequestAPI1 extends AsyncTask<Void, Void, JSONObject> {

    @Override
    protected JSONObject doInBackground(Void... params) {
        String string = HttpUtils.requestGet(captchaURL + "?t=" +
System.currentTimeMillis());
        JSONObject jsonObject = null;
        try {
            jsonObject = new JSONObject(string);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return jsonObject;
```

```
    }

    @Override
    protected void onPostExecute(JSONObject parmas) {
        // 继续验证
        Log.i(TAG, "RequestAPI1-->onPostExecute: " + parmas);
        // SDK可识别格式为
        // {"success":1,"challenge":"...","gt":"...","new_captcha":true}
        // TODO 设置返回api1数据，即使为null也要设置，SDK内部已处理
        gt3ConfigBean.setApi1Json(parmas);
        // 继续调起验证
        gt3GeetestUtils.getGeetest();
    }
}
```

自定义 API2，根据验证结果设置成功失败弹框，示例如下，更多详情可参考demo。

```
/**
 * 自定义 API2 回调
 * @param result API2请求需上传的参数
 */
@Override
public void onDialogResult(String result) {
    // 开启自定义 API2 逻辑
    new RequestAPI2().execute(result);
}
```

```
/**
 * 请求api2
 */
class RequestAPI2 extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... params) {
        if (!TextUtils.isEmpty(params[0])) {
            return HttpUtils.requestPost(validateURL+ "?t=" +
System.currentTimeMillis(), params[0]);
        } else {
            return null;
        }
    }

    @Override
    protected void onPostExecute(String result) {
        Log.i(TAG, "RequestAPI2-->onPostExecute: " + result);
        if (!TextUtils.isEmpty(result)) {
            try {
                JSONObject jsonObject = new JSONObject(result);
```

```
                String status = jsonObject.getString("status");
                if ("success".equals(status)) {
                    gt3GeetestUtils.showSuccessDialog();
                } else {
                    gt3GeetestUtils.showFailedDialog();
                }
            } catch (Exception e) {
                gt3GeetestUtils.showFailedDialog();
                e.printStackTrace();
            }
        } else {
            gt3GeetestUtils.showFailedDialog();
        }
    }
}
```

iOS 端

0.13.2 级 0.13.2 之后的版本

1. 给 `UIViewController` 实现 `GT3AsyncTaskProtocol` 协议

```
// 实现 API2 请求，以完成 validation 过程
- (void)executeValidationTaskWithValidateParam:(GT3ValidationParam *)param
completion:(void (^)(BOOL, GT3Error * _Nullable))completion {
    // 1. 使用 GT3ValidationParam 构造 API2 请求参数，以下以表单格式为示例（实际情
况开发者后端为准）
    __block NSMutableArray *postArray = [[NSMutableArray alloc] init];
    [param.result enumerateKeysAndObjectsUsingBlock:^(id key, id obj, BOOL
* stop) {
        NSString *paramItem = [NSString stringWithFormat:@"%@=%@", key,
obj];
        [postArray addObject:paramItem];
    }];

    NSString *postForm = [postArray componentsJoinedByString:@"&"];

    // 2. 使用网络库发送 API2 请求，并处理响应数据和可能存在的网络错误

    ...

    // 3. 将结果并通过 completion 返回给验证管理器，以便管理器处理收尾的处理

    if (completion) {
      if (!error) {
      completion(YES, nil);
    } else {
      completion(NO, error);
```

```
    }
    }
}

// 实现 API1 请求，以完成 register 过程
- (void)executeRegisterTaskWithCompletion:(void (^)(GT3RegisterParameter *
_Nullable, GT3Error * _Nullable))completion {
    // 1. 使用网络库请求 API1，以获得验证参数
    ...

    // 2. 将 API1 返回的数据 gt、challenge、success 转化为
GT3RegisterParameter，并通过 completion 返回给管理器
    GT3RegisterParameter *param = [[GT3RegisterParameter alloc] init];
    param.gt        = gt;
    param.challenge = challenge;
    param.success   = success;

    if (completion) {
        completion(param, nil);
    }
}
```

2. 在发起验证前，将`GT3AsyncTaskProtocol`的实现注册给管理器

```
- (GT3CaptchaManager *)manager {
    if (!_manager) {
        _manager = [[GT3CaptchaManager alloc] initWithAPI1:nil API2:nil
timeout:5.0];
        _manager.delegate = self;

        [_manager registerCaptchaWithCustomAsyncTask:self completion:nil];
    }
    return _manager;
}
```

0.13.1 及 0.13.1 之前的版本

考虑到部分公司的服务是需要鉴权或者自定义参数的, 可以通过下面的代理方法来对API1请求实现自定义。

```
- (void)gtCaptcha:(GT3CaptchaManager *)manager willSendRequestAPI1:
(NSURLRequest *)originalRequest withReplacedHandler:(void (^)(NSURLRequest
*))replacedHandler {
    NSMutableURLRequest *mRequest = [originalRequest mutableCopy];

    /**
     TO-DO, 处理mRequest, 进行自定义
     */

    replacedHandler(mRequest);
}
```

考虑到部分公司的服务是需要鉴权或者自定义参数的, 可以通过下面的代理方法来对API1请求实现自定义。

```
- (void)gtCaptcha:(GT3CaptchaManager *)manager
willSendSecondaryCaptchaRequest:(NSURLRequest *)originalRequest
withReplacedRequest:(void (^)(NSMutableURLRequest *))replacedRequest {
    /// 如果需要可以修改二次验证向服务器发送的请求
    NSMutableURLRequest *mReuqest = [originalRequest mutableCopy];
    NSData *secodaryData = mReuqest.HTTPBody;
    NSLog(@"data: %@", [[NSString alloc] initWithData:secodaryData
encoding:NSUTF8StringEncoding]);

    /**
     TO-DO, 处理secodaryData, 添加业务数据
     */

//    NSData *newData = [secodaryData handlerYourData]
//    mReuqest.HTTPMethod = newData;
    replacedRequest(mReuqest);
}
```