ALIBABA CLOUD

阿里云

数据风控
最佳实践

文档版本：20210402

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.通过RAM配置阿里云验证码权限控制

阿里云验证码支持阿里云访问控制（RAM）服务，您可以通过RAM的用户、用户组、角色、权限策略实现对阿里云验证码的权限控制。

控制台登录与编程访问分离

在业务中集成验证码能力，通常您需要在先控制台创建业务配置，然后在您的业务前端和服务端分别集成验证码提供的功能接入代码。

在服务端集成验证码功能接入代码时，需要您填写访问密钥（AccessKey）参数信息，供阿里云验证码服务端验证您的身份。

通过创建不同的RAM用户，可以实现控制台登录与编程访问权限分离，即为创建业务配置与服务端调用设置不同的用户，更好地实现人员权限管控。您只需在创建RAM用户时为两个用户分别单独选择**控制台密码登录**和**编程访问**访问方式即可：

- 拥有**编程访问**方式的用户账号仅用于业务服务端的阿里云验证码调用。
- 拥有**控制台密码登录**访问方式的用户在阿里云验证码控制台中管理业务配置。

> **说明**
>
> 您还可以为该用户开启多因素认证（MFA），进一步提升用户访问安全性。

关于创建RAM用户的具体操作步骤，请参见创建RAM用户。

功能权限控制

RAM提供系统默认权限策略 `AliyunYundunAFSFullAccess` 用于管理用户通过控制台访问或通过编程调用阿里云验证码服务接口，您只需为相关的RAM用户授予该权限，该RAM用户即可正常使用阿里云验证码。

> **说明**
>
> 系统默认权限策略 `AliyunYundunAFSFullAccess` 的授权粒度为服务级别，即拥有该权限的用户可以使用阿里云验证码产品提供的所有功能。

如果您期望设置更细粒度的权限控制，您可以通过创建自定义策略的方式实现。例如，您可以创建包含以下策略内容的自定义策略，实现验证码业务配置只读权限：

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": "yundun-afs:Describe*",
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

创建该自定义权限策略并为指定RAM用户授予该权限后，该用户在控制台中将只能查看业务配置，无法创建配置、修改配置、设置自定义样式、设置预警，也无法使用该账号的AccessKey调用阿里云验证码接口。

关于创建自定义权限策略和用户授权的具体操作步骤，请参见创建自定义策略和为RAM用户授权。

2.在Android App中接入HTML5滑块验证

随着混合模式移动应用（Hybrid App）开发技术的日益成熟，您可以通过在App业务中启用WebView组件的方式直接接入移动端HTML5业务类型的滑动验证组件，实现App业务中的人机对抗。用户只需通过类似滑动解锁的方式实现验证码，对正常用户来说无需思考即可通过人机识别（图灵测试）的挑战。

同时，移动端HTML5业务类型的验证码组件具备快速迭代、强兼容性等优势。通过这种方式在App业务中接入滑动验证服务，您无需再为Native App组件中的各种依赖、引用导致的兼容性问题而感到困扰，而移动端HTML5业务类型的滑动验证组件的快速迭代也将帮助您的App业务更好地应对“强对抗”场景。

在Android App中接入HTML5应用类型的滑块验证组件主要包含以下步骤：

1. **Android App端**：利用WebView组件在App应用中开启并部署需要接入滑动验证组件的业务页面。
2. **HTML5页面前端**：通过JavaScript函数调用Native Java代码并将记录的用户行为参数值传至阿里云验证码服务端。
3. **HTML5页面对应的服务端**：集成滑动验证服务端SDK。

步骤 1：在Android App端开启并部署业务页面

1. 在您的Android App工程的Activity文件中，导入WebView组件的依赖库。

```
import android.webkit.WebView;
import android.webkit.WebSettings;
import android.webkit.WebViewClient;
import android.webkit.WebChromeClient;
```

2. 在AndroidManifest.xml配置文件中，设置网页加载的权限。

   说明：如果存在其它HTTP资源调用，也需要增加相应的配置。

```
<uses-permission android:name="android.permission.INTERNET"/>
<application
...
android:usesCleartextTraffic="true"
...>
```

3. 在activity_main.xml布局文件中，添加WebView组件。

```
<WebView android:id="@+id/webview"
  android:layout_height="match_parent"
  android:layout_width="match_parent" />
```

4. 在Activity文件中，加载HTML5业务页面。

```
public class MainActivity extends AppCompatActivity {
  private WebView testWebview;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    initView();
  }
  private void initView() {
    // 页面布局。
    testWebview = (WebView) findViewById(R.id.webview);
    // 设置屏幕自适应。
    testWebview.getSettings().setUseWideViewPort(true);
    testWebview.getSettings().setLoadWithOverviewMode(true);
    // 建议禁止缓存加载，以确保在攻击发生时可快速获取最新的滑动验证组件进行对抗。
  testWebview.getSettings().setCacheMode(WebSettings.LOAD_NO_CACHE);
    // 设置不使用默认浏览器，而直接使用WebView组件加载页面。
    testWebview.setWebViewClient(new WebViewClient(){
      @Override
      public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return true;
      }
    });
    // 设置WebView组件支持加载JavaScript。
    testWebview.getSettings().setJavaScriptEnabled(true);
      // 建立JavaScript调用Java接口的桥梁。
    testWebview.addJavascriptInterface(new testJsInterface(), "testInterface");
    // 加载业务页面。
    testWebview.loadUrl("http://39.x.x.x/demo/");
  }
}
```

5. 在Activity文件中，添加自定义Java接口（testJsInterface），并定义getSlideData方法获取滑块数据。

```
import android.webkit.JavascriptInterface;
public class testJsInterface {
  @JavascriptInterface
  public void getSlideData(String callData) {
    System.out.println(callData);
  }
}
```

6. 在Activity的initView()方法中，将所添加的自定义Java接口与JavaScript函数绑定。

```
// 设置WebView组件支持JavaScript。
testWebview.getSettings().setJavaScriptEnabled(true);
// 建立JavaScript调用Java接口的桥梁。
testWebview.addJavascriptInterface(new testJsInterface(), "testInterface");
```

步骤 2：在HTML5页面前端代码中添加JavaScript函数调用App业务中的JavaScript函数

确认已在HTML5业务页面前端代码中集成滑动验证提供的前端接入代码。关于HTML5页面的滑动验证集成方式，请参见前端接入代码集成。

在已集成滑动验证前端接入代码的HTML5业务页面中，添加对应的JavaScript函数，实现在用户将滑块滑动至末端时的回调参数中返回自定义Java接口。

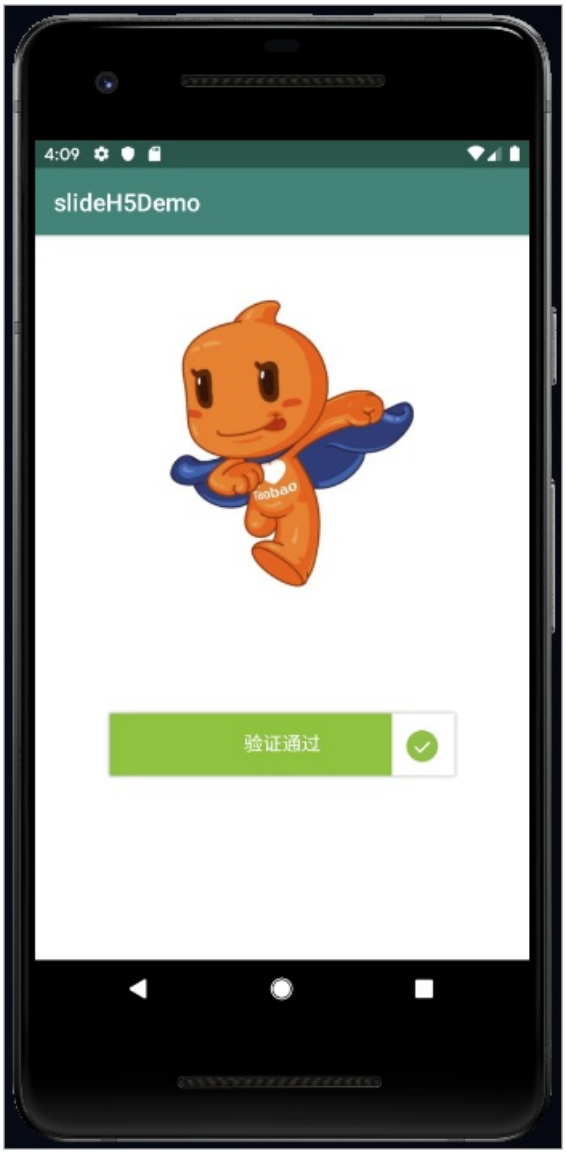
```
function slideCallBack(data) {
  var result = {
    'nc_token':nc_token,
    'sessionid':data.csessionid,
    'sig':data.sig
  };
  // 绑定Java接口与JavaScript函数。
  window.testInterface.getSlideData(JSON.stringify(result));
}
```

步骤 3：在HTML5页面对应的服务端集成滑动验证服务端SDK

在HTML5业务页面对应的服务端中，集成滑动验证提供的服务端功能SDK。关于HTML5页面的前端接入代码集成方式，请参见滑动验证服务端代码集成。

示例效果

完成上述步骤后，您的Andriod App业务中即接入HTML5应用类型的滑块验证组件实现验证码。



当用户在App中将滑块滑动至末端时，HTML5业务页面将调用testInterface.getSlideData接口获取s所记录的用户行为参数值，将其传至阿里云验证码服务端识别机器风险，根据判决结果和所设定的业务逻辑返回相应结果。

```
D/eglCodecCommon: setVertexArrayObject: set vao to 0 (0) 7 9
I/System.out: {"nc_token":"CF_APP_1:1561535675592:0.09196837861932572","sessionid":"016wK9fOQuR_XVKtFywYu65fLe5JDdW-FU4TeeviysEN8qxD
D/eglCodecCommon: setVertexArrayObject: set vao to 1 (1) 7 4
```

3.使用UIWebView在iOS App中接入HTML5滑块验证

随着混合模式移动应用（Hybrid App）开发技术的日益成熟，您可以通过在App业务中启用WebView组件的方式直接接入移动端HTML5业务类型的滑动验证组件，实现App业务中的人机对抗。用户只需通过类似滑动解锁的方式来实现验证码，对正常用户来说无需思考即可通过人机识别（图灵测试）的挑战。

同时，移动端HTML5业务类型的验证码组件具备快速迭代、强兼容性等优势。通过这种方式在App业务中接入滑动验证服务，您无需再为Native App组件中的各种依赖、引用导致的兼容性问题而感到困扰，而移动端HTML5业务类型的滑动验证组件的快速迭代也将帮助您的App业务更好地应对强对抗场景。

本示例使用iOS原生框架JavaScript Core.FrameWork实现在iOS App中接入HTML5应用类型的滑块验证组件。您可以根据App业务的开发环境选择其它合适的第三方框架。

在iOS App中接入HTML5应用类型的滑块验证组件主要包含以下步骤：

1. **iOS App端**：利用WebView组件在App应用中开启并部署需要接入滑动验证组件的业务页面。
2. **HTML5页面前端**：通过JavaScript函数调用Object-C方法并将记录的用户行为参数值传至阿里云验证码服务端。
3. **HTML5页面对应的服务端**：集成滑动验证服务端SDK。

步骤一：在iOS App端开启并部署业务页面

1. 在您的iOS App工程的.h文件中，导入框架的依赖库。

```
#import <UIKit/UIKit.h>
#import <JavaScriptCore/JavaScriptCore.h>
```

2. 在您的iOS App工程的.m文件中，加载HTML5业务页面。

```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.webView.delegate = self;
    // 配置页面的自适应缩放。
    self.webView.scalesPageToFit = YES;

    // 加载远程HTML5业务页面。
    [self.webView loadRequest:[[NSURLRequest alloc] initWithURL:[NSURL URLWithString:@"http://xxx.com/demo/"]]];

    // 您也可以直接加载本地HTML文件。但使用这种加载方式将无法方便地升级该HTML5页面，因此不推荐使用。
    //  NSURL *baseURL = [NSURL fileURLWithPath:[[NSBundle mainBundle] bundlePath]];
    //  NSString *path = [[NSBundle mainBundle] pathForResource:@"demo" ofType:@"html"];
    //  NSString *html = [NSString stringWithContentsOfFile:path encoding:NSUTF8StringEncoding error:nil];
    //  [self.webView loadHTMLString:html baseURL:baseURL];
}
```

3. 设置JavaScript对象，作为Native App引用与HTML5业务页面间的桥梁。

> 说明
>
> 本示例中将该对象命名为testWebView。

```
- (void)webViewDidFinishLoad:(UIWebView *)webView
{
  // 获取context对象。
  self.context = [self.webView valueForKeyPath:@"documentView.webView.mainFrame.javaScriptC
ontext"];

  // 设置testWebView JS对象，并将该对象指向其自身。
  self.context[@"testWebView"] = self;
  self.context.exceptionHandler = ^(JSContext *context, JSValue *exceptionValue) {
    context.exception = exceptionValue;
    NSLog(@"异常信息：%@", exceptionValue);
  };
}
```

4. 在该对象的Object-C中，设置相应的方法获取滑动验证服务的返回值和您自身的业务逻辑代码。

```
- (void)getSlideData:(NSString *)callData {
  NSLog(@"Get:%@", callData);
}
```

5. 在.h文件中，声明所设置的作为Native App应用与HTML5业务页面间桥梁的JavaScript对象名和方法。

```
#import <UIKit/UIKit.h>
#import <JavaScriptCore/JavaScriptCore.h>

@protocol JSObjcDelegate

// 声明testWebView对象调用的JS方法。
(void)getSlideData:(NSString*)callData;

@end
@interface ViewController : UIViewController <UIWebViewDelegate,JSObjcDelegate>
@property (nonatomic, strong) JSContext *context;
@property (weak, nonatomic) IBOutlet UIWebView *webView;

@end
```

步骤二：在HTML5页面前端代码中添加JavaScript函数调用App业务中的Object-C

确认已在HTML5业务页面前端代码中集成滑动验证提供的前端接入代码。关于HTML5页面的前端接入代码集成方式，请参见滑动验证集成方式。

在已集成滑动验证前端接入代码的HTML5业务页面中，添加对应的JavaScript函数，实现在用户将滑块滑动至末端时的回调参数中返回Object-C。

```
function slideCallBack(data) {
  var result = {
    'nc_token':nc_token,
    'sessionid':data.csessionid,
    'sig':data.sig
  };
  window.testWebView.getSlideData(JSON.stringify(result));
}
```

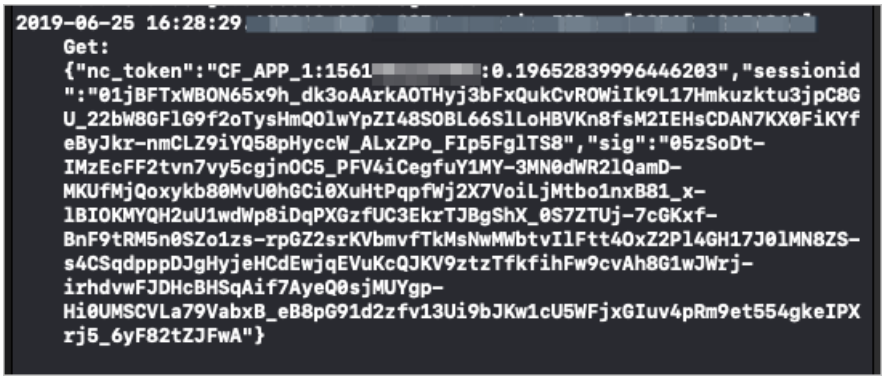
步骤三：在HTML5页面对应的服务端集成滑动验证服务端SDK

在HTML5业务页面对应的服务端中，集成滑动验证提供的服务端功能SDK。关于HTML5页面的前端接入代码集成方式，请参见滑动验证服务端代码集成。

示例效果

完成上述步骤后，您的iOS App业务中即接入HTML5应用类型的滑块验证组件实现验证码。

当用户在App中将滑块滑动至末端时，HTML5业务页面将调用testWebView.getSlideData对象获取所记录的用户行为参数值，将其传至阿里云验证码服务端识别机器风险，根据判决结果和所设定的业务逻辑返回相应结果。



4.使用WKWebView在iOS App中接入滑动验证（OC）

随着混合模式移动应用（Hybrid App）开发技术的日益成熟，您可以通过在App业务中启用WebView组件的方式直接接入移动端HTML5应用类型的滑动验证组件，实现App业务中的人机对抗。使用者只需通过滑动解锁的方式来实现验证码，即可轻松通过人机识别（图灵测试）的挑战。

背景信息

移动端HTML5应用类型的验证码组件具备快速迭代、强兼容性等优势。使用该方式在App业务中接入滑动验证服务，您无需再为Native App组件中的各种依赖、引用导致的兼容性问题而感到困扰，而移动端HTML5应用业务类型的滑动验证组件的快速迭代也将帮助您的App业务更好地应对强对抗场景。

接入流程

在iOS App中接入HTML5应用类型的滑块验证组件主要包含以下步骤：

1. **iOS App端**：利用WebView组件在App应用中开启并部署需要接入滑动验证组件的业务页面。
2. **HTML5页面前端**：通过JavaScript函数调用Object-C方法并将记录的用户行为参数值传至阿里云验证码服务端。
3. **HTML5页面对应的服务端**：集成滑动验证服务端SDK。

步骤一：在iOS App端开启并部署业务页面

1. 在您的iOS App工程中后缀为H的文件中，导入框架的依赖库及相关声明。

```
#import <UIKit/UIKit.h>
#import <WebKit/WebKit.h>
@interface ViewController : UIViewController;
@property (nonatomic, strong) WKWebView *  webView;
```

2. 在您的iOS App工程中后缀为M的文件中，加载HTML5应用页面。

```
- (void)viewDidLoad {
   [super viewDidLoad];
   NSURL *url = [NSURL URLWithString:@"https://xxx.com/demo/"];
   NSURLRequest *urlRequest = [NSURLRequest requestWithURL:url];
   [self.webView loadRequest:urlRequest]; // 加载页面
}
```

3. 设置WKWebView对象，通过WKScriptMessageHandler协议来实现JS与WKWebView的交互。

```
- (WKWebView *)webView {
  // 配置页面自适应缩放
  NSString *jscript = @"var meta = document.createElement('meta'); meta.setAttribute('name', 'viewport'); meta.setAttribute('content', 'width=device-width'); document.getElementsByTagName('head')[0].appendChild(meta)";
  WKUserScript *userScript = [[WKUserScript alloc] initWithSource:jscript injectionTime:WKUserScriptInjectionTimeAtDocumentEnd forMainFrameOnly:YES];
  WKUserContentController *userContentController = [[WKUserContentController alloc] init];
  [userContentController addUserScript:userScript];
  // 添加HTML页面js的调用方法，这里默认添加的方法名称为getSlideData，可自行按需更改
  [userContentController addScriptMessageHandler:self name:@"getSlideData"];
  // 配置WKWebView
  WKWebViewConfiguration *configuration = [[WKWebViewConfiguration alloc] init];
  configuration.userContentController = userContentController;
  // 显示WKWebView
  _webView = [[WKWebView alloc] initWithFrame:self.view.frame configuration:configuration];
  [self.view addSubview:_webView];
  return _webView;
}
```

4. 设置WKScriptMessageHandler协议相应的方法获取滑动验证服务的返回值和您自身的业务逻辑代码。

```
- (void)userContentController:(WKUserContentController *)userContentController didReceiveScriptMessage:(WKScriptMessage *)message {
  // 获取HTML页面js调用相应方法获取的验证码服务返回值，方法名应与添加在HTML页面js调用方法名保持一致
  if([message.name isEqualToString:@"getSlideData"]){
    NSLog(@"参数%@", message.body);
  }
}
```

步骤二：在HTML5页面前端代码中添加JavaScript函数调用WKScriptMessageHandler协议实现的JavaScript方法

> **注意**
>
> 在执行本步骤前，您需要完成在HTML5业务页面前端代码中集成滑动验证提供的前端接入代码。关于HTML5页面的前端接入代码集成的详细步骤，请参见滑动验证集成方式。

在已集成滑动验证前端接入代码的HTML5业务页面中，添加以下的JavaScript函数，实现在用户将滑块滑动至末端时，可以调用相关JavaScript方法获取回调参数。

```
function (data) {
 // 调用WkScriptMessageHandler协议实现的JavaScript方法获取参数sessionID及sig
 window.console && console.log(data.sessionId)
 window.console && console.log(data.sig)
 window.webkit.messageHandlers.getSlideData.postMessage(data)
}
```

步骤三：在HTML5页面对应的服务端集成滑动验证服务端SDK

在HTML5业务页面对应的服务端中，集成滑动验证提供的服务端功能SDK。关于HTML5页面的前端接入代码集成的详细步骤，请参见服务端代码集成的概述。

接入效果示例

完成上述步骤后，您的iOS App业务将接入HTML5应用类型的滑块验证组件，并实现验证码。

当App业务使用者在App中将滑块滑动至末端时，HTML5业务页面将调用window.webkit.messageHandlers.getSlideData.postMessage方法获取相关参数。

```
2020-10-16 17:42:49.434146+0800
    wkwebviewtest[5187:2216648] 参数{
    sessionId =
        "013QHwVNRchjUK-1yFT9-mbEoMz5es3WaT5ou1tO-LbH5a0vL
        kxZrFq6Tg07sLIUA1mMTrZ5alg5MTG_-316HWnzyTMPeJ_H9wh
        UO3L3bXcJ-zHWWytiSlkvSVd1WRPv-4ReCqElj9t4iqYG_iFPc
        cf6sfowVJ9q0BbSqQAk22cXD4v_ywJilUOodZmj6abtUW";
    sig =
        "050ryIkFc2GgsVD4cjdQb2DyBXmchzP0_tv_kdOZ90zSphmBE
        5Dkr72cdwJlb3J8nnszW2Es4pPUzT-I84s0fJQhrHYcQacaCX4
        yjs1sgwiERx4pAgI4rhZDJUFp7bAaI9sOw7tvbhMHdJxMFV07x
        e8iyAo-RkA3V452b7DUiKZynczHR2bNsJsnJ2JWyaOXhgDB3G2
        3w6f5v0-RqfCYrmJDz1m_EM3HXA6JCbrS32_OeMQQKew1woMM3
        Ydzc-GGw7QgfJP-kmGBFA4Ev1VvSeGe7mBx5_c6bR2M89rvtdp
        3pHJTPH1w6aCxvMMU4YevY_1LY8JXoh1PmDK3N1HatDuzvR8bz
        i-EcymWCS8ITiSioM3O39Vctt7nStngHFUQ-O2ohrrXANDYC73
        cjVpDJY0Akc5geGQaaHFk3xbn7UZFQb0AGevj66GlL_qDpWjgT
        5";
}
```

5.使用WKWebView在IOS App中接入滑动验证（Swift）

随着混合模式移动应用（Hybrid App）开发技术的日益成熟，您可以通过在App业务中启用WebView组件的方式直接接入移动端HTML5业务类型的滑动验证组件，实现App业务中的人机对抗。用户只需通过类似滑动解锁的方式实现验证码，对正常用户来说即可轻松通过人机识别（图灵测试）的挑战。

背景信息

移动端HTML5业务类型的验证码组件具备快速迭代、强兼容性等优势。通过这种方式在App业务中接入滑动验证服务，您无需再为Native App组件中的各种依赖、引用导致的兼容性问题而感到困扰，而移动端HTML5业务类型的滑动验证组件的快速迭代也将帮助您的App业务更好地应对强对抗场景。

接入流程

在iOS App中接入HTML5应用类型的滑块验证组件主要包含以下步骤：

1. **iOS App端**：利用WebView组件在App应用中开启并部署需要接入滑动验证组件的业务页面。
2. **HTML5页面前端**：通过JavaScript函数调用Swift方法并将记录的用户行为参数值传至阿里云验证码服务端。
3. **HTML5页面对应的服务端**：集成滑动验证服务端SDK。

步骤一：在iOS App端开启并部署业务页面

1. 在您的iOS App工程的Controller文件中，导入相关依赖。

```
import UIKit
import WebKit
```

2. 在您的iOS App工程的Conteroller文件中，加载HTML5业务页面。

```
override func viewDidLoad() {
  super.viewDidLoad()
  view.addSubview(webView)
  let url = URL(string: "https://xxx.com/demo/")
  let request = URLRequest(url: url!);
  webView.load(request); // 加载页面
}
```

3. 设置WKWebView对象，通过WKScriptMessageHandler协议来实现JS与WKWebView的交互。

```
lazy var webView: WKWebView = {
  // 配置页面自适应缩放
  let javascript = "var meta = document.createElement('meta'); meta.setAttribute('name', 'viewport');
meta.setAttribute('content', 'width=device-width'); document.getElementsByTagName('head')[0].app
endChild(meta)";
  let configuration = WKWebViewConfiguration();
  let userScript = WKUserScript(source: javascript, injectionTime: .atDocumentEnd, forMainFrameOnly:
true);
  let usercontroller = WKUserContentController();
  usercontroller.addUserScript(userScript);
  configuration.userContentController = usercontroller;
  // 添加HTML页面js的调用方法，这里默认添加的方法名称为getSlideData，可自行按需更改
  configuration.userContentController.add(self, name: "getSlideData");
  // 配置WKWebView
  let webView = WKWebView(frame: self.view.frame, configuration: configuration);
  webView.navigationDelegate = self;
  return webView;
}()
```

4. 设置WKScriptMessageHandler协议相应的方法获取滑动验证服务的返回值和您自身的业务逻辑代码。

```
func userContentController(_ userContentController: WKUserContentController, didReceive message:
WKScriptMessage) {
  if(messsage.name == "getSlideData"){
    print(message.body)
  }
}
```

步骤二：在HTML5页面前端代码中添加JavaScript函数调用WKScriptMessageHandler协议实现的JavaScript方法

> **注意**
>
> 在执行本步骤前，您需要完成在HTML5业务页面前端代码中集成滑动验证提供的前端接入代码。关于HTML5页面的前端接入代码集成的详细步骤，请参见滑动验证集成方式。

在已集成滑动验证前端接入代码的HTML5业务页面中，添加对应的JavaScript函数，实现在用户将滑块滑动至末端时的回调参数中调用相关JavaScript方法进行参数获取。

```
function (data) {
 // 调用WkScriptMessageHandler协议实现的JavaScript方法获取参数sessionID及sig
 window.console && console.log(data.sessionId)
 window.console && console.log(data.sig)
 window.webkit.messageHandlers.getSlideData.postMessage(data)
}
```

步骤三：在HTML5页面对应的服务端集成滑动验证服务端SDK

在HTML5业务页面对应的服务端中，集成滑动验证提供的服务端功能SDK。关于HTML5页面的前端接入代码集成的详细步骤，请参见服务端代码集成的概述。

接入效果示例

完成上述步骤后，您的iOS App业务将接入HTML5应用类型的滑块验证组件，并实现验证码。

当App业务使用者在App中将滑块滑动至末端时，HTML5业务页面将调用window.webkit.messageHandlers.getSlideData.postMessage方法获取相关参数。

```
2020-10-16 17:42:49.434146+0800
    wkwebviewtest[5187:2216648] 参数{
    sessionId =
        "013QHwVNRchjUK-1yFT9-mbEoMz5es3WaT5ou1tO-LbH5a0vL
        kxZrFq6Tg07sLIUA1mMTrZ5alg5MTG_-316HWnzyTMPeJ_H9wh
        UO3L3bXcJ-zHWWytiSlkvSVd1WRPv-4ReCqElj9t4iqYG_iFPc
        cf6sfowVJ9q0BbSqQAk22cXD4v_ywJilUOodZmj6abtUW";
    sig =
        "050ryIkFc2GgsVD4cjdQb2DyBXmchzP0_tv_kdOZ90zSphmBE
        5Dkr72cdwJlb3J8nnszW2Es4pPUzT-I84s0fJQhrHYcQacaCX4
        yjs1sgwiERx4pAgI4rhZDJUFp7bAaI9sOw7tvbhMHdJxMFV07x
        e8iyAo-RkA3V452b7DUiKZynczHR2bNsJsnJ2JWyaOXhgDB3G2
        3w6f5v0-RqfCYrmJDz1m_EM3HXA6JCbrS32_OeMQQKew1woMM3
        Ydzc-GGw7QgfJP-kmGBFA4Ev1VvSeGe7mBx5_c6bR2M89rvtdp
        3pHJTPH1w6aCxvMMU4YevY_1LY8JXoh1PmDK3N1HatDuzvR8bz
        i-EcymWCS8ITiSioM3O39Vctt7nStngHFUQ-O2ohrrXANDYC73
        cjVpDJY0Akc5geGQaaHFk3xbn7UZFQb0AGevj66GlL_qDpWjgT
        5";
}
```