验证码

前端接入文档

产品文档

腾讯云

文档目录

前端接入文档
Web 前端接入

最近更新时间：2022-04-07 14:47:06

接入步骤

前提条件

接入验证码前，进入图形验证 页完成新建验证。可在**验证列表**查看 验证码接入所需的 CaptchaAppId 以及 AppSecretKey。

快速接入

> **注意：**
> 小程序插件 CaptchaAppId 仅限小程序插件接入方式使用，请勿使用在 Web 前端接入。

以下代码为图形验证码直接绑定按钮的前端接入示例代码，如有其他接入要求（执行逻辑后调用验证码、配置验证码语言/弹出方式等），请参考定制接入章节。

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Web 前端接入示例</title>
<!-- 验证码程序依赖(必须)。请勿修改以下程序依赖，如使用本地缓存，或通过其他手段规避加载，会影响程序的正常使用。 -->
<script src="https://ssl.captcha.qq.com/TCaptcha.js"></script>
</head>
```

```
<body>
<!--点击此元素会自动激活验证码，此例使用的button元素，也可以使用div、span等-->
<!--id：(不可变) 元素的 ID，值必须是 'TencentCaptcha'-->
<!--data-appid：(必须) 验证码CaptchaAppId，从腾讯云的验证码控制台中获取，验证码控制台页面内【图形验证】>【验证列表】进行查看 。如果未新建验证，请根据业务需求选择适合的验证渠道、验证场景进行新建-->
<!--data-cbfn：(必须) 回调函数名，函数名要与 data-cbfn 相同-->
<!--data-biz-state :(可选) 业务自定义透传参数，会在回调函数内获取到 （res.bizState）-->
<button id="TencentCaptcha" data-appid="你的验证码CaptchaAppId" data-cbfn="callbackName" data-biz-state="data-biz-state"
type="button">验证</button>
</body>

<script>
// 回调函数需要放在全局对象window下
window.callbackName = function (res) {
// 返回结果
// ret Int 验证结果，0：验证成功。2：用户主动关闭验证码。
// ticket String 验证成功的票据，当且仅当 ret = 0 时 ticket 有值。
// CaptchaAppId String 验证码应用ID。
// bizState Any 自定义透传参数。
// randstr String 本次验证的随机串，请求后台接口时需带上。
console.log('callback:', res);


// res（用户主动关闭验证码）= {ret: 2, ticket: null}
// res（验证成功） = {ret: 0, ticket: "String", randstr: "String"}
// res（客户端出现异常错误 仍返回可用票据） = {ret: 0, ticket: "String", randstr: "String", errorCode: Number, errorMessage: "String"}
if (res.ret === 0) {
// 复制结果至剪切板
let str = `【randstr】->【${res.randstr}】【ticket】->【${res.ticket}】`;
let ipt = document.createElement('input');
ipt.value = str;
document.body.appendChild(ipt);
ipt.select();
document.execCommand("Copy");
```

```
document.body.removeChild(ipt);
alert('1. 返回结果（randstr、ticket）已复制到剪切板，ctrl+v 查看。\n2. 打开浏览器控制台，查看完整返回结果。');

}
}
</script>

</html>
```

> ⚠ **注意：**
> 验证码客户端接入完成后，验证码后台需二次核查验证码票据结果，请进行 后台 API 接入 操作，确保验证安全性。更多详情请参见 核查验证码票据文档 。

异常处理

以下为验证 JS 文件引入错误处理。

1. 定义错误处理函数

```
// 在脚本加载或初始化错误时 需手动绑定button元素的事件确保流程正常
// 函数定义需在script加载前
function TCaptchaLoadError(){
var CaptchaAppId=''
document.getElementById('TencentCaptcha').addEventListener('click', function () {
var CaptchaAppId = ''
/* 生成票据或自行做其它处理 */
var ticket = 'terror_1001_' + CaptchaAppId + '_' + Math.floor(new Date().getTime()/1000)
window.callback({
ret: 0,
randstr: '@'+Math.random().toString(36).substr(2),
ticket: ticket,
errorCode: 1001,
errorMessage: 'jsload_error'
})
}, false)
}
```

2. 在 html 中加入以下代码引入验证 JS 文件。

```
<!-- 如果script是在head中引入 onerror的函数需等domready再绑定元素事件 在body中加载无需等待
-->
<script src="https://ssl.captcha.qq.com/TCaptcha.js" onerror="TCaptchaLoadError()"></scri
pt>
```

3. 建议根据 ticket 和 errorCode 情况而非 ret 的值做处理，更精确和稳定。

```
window.callback = function(res){
/* res（验证成功） = {ret: 0, ticket: "String", randstr: "String"}
res（客户端出现异常错误 仍返回可用票据） = {ret: 0, ticket: "String", randstr: "String",
errorCode: Number, errorMessage: "String"}
res（用户主动关闭验证码）= {ret: 2}
*/
if (res.ticket){
// 上传票据 可根据errorCode和errorMessage做特殊处理或统计
}
}
```

定制接入

1. 如果不使用默认 id，可以通过实例化 TencentCaptcha 类，自定义参数来创建验证码组件。

> ⚠ 注意：
> 绑定单击的元素不要使用 id="TencentCaptcha" 的元素，避免重复绑定单击。

2. TencentCaptcha 支持多种参数的重载，以下3种初始化方法，可根据具体情况选择其中一种。
   - **手动初始化。**
     手动初始化的情况，一般是单击一个元素，执行一段逻辑，才调用验证码

```
new TencentCaptcha(CaptchaAppId, callback, options);
```

参数名	值类型	说明
CaptchaAppId	String	验证码应用 ID
callback	Function	回调函数

参数名	值类型	说明
options	Object	更多配置参数, 请参见 配置参数

- **绑定到一个元素。**

```
new TencentCaptcha(element);
```

参数名	值类型	说明
element	HTMLElement	验证码将绑定 click 事件到该元素上，该方式需要确保元素上有 data-appid 和 data-cbfn 属性。

- **手动初始化并绑定到一个元素。**

```
new TencentCaptcha(element, CaptchaAppId, callback, options);
```

参数名	值类型	说明
element	HTMLElement	需要绑定click事件的元素
CaptchaAppId	String	验证码应用 ID
callback	Function	回调函数
options	Object	更多配置参数, 请参见 配置参数

- **示例代码**

```
//方法1: 直接生成一个验证码对象。
try {
var captcha1 = new TencentCaptcha('CaptchaAppId', callback);
captcha1.show(); // 显示验证码
} catch (error) {
loadErrorCallback();
}
//方法2:绑定一个元素并自动识别场景id和回调。
try {
// 绑定一个元素并自动识别场景id和回调
```

```
// 验证码会读取dom上的`data-appid`和`data-cbfn`以及`data-biz-state`(可选)自动初始化
new TencentCaptcha(document.getElementById('TencentCaptcha'));
} catch (error) {
loadErrorCallback();
}
//方法3：绑定一个元素并手动传入场景Id和回调。
try {
new TencentCaptcha(
document.getElementById('TencentCaptcha'),
'CaptchaAppId',
callback,
{ bizState: '自定义透传参数' },
);
} catch (error) {
loadErrorCallback();
}

function callback(res) {
/* res（验证成功） = {ret: 0, ticket: "String", randstr: "String"}
res（客户端出现异常错误 仍返回可用票据） = {ret: 0, ticket: "String", randstr: "String", errorCod
e: Number, errorMessage: "String"}
res（用户主动关闭验证码）= {ret: 2}
*/
if (res.ticket){
// 上传票据 可根据errorCode和errorMessage做特殊处理或统计
}
}
// 验证码js加载错误处理
function loadErrorCallback() {
var CaptchaAppId = ''
/* 生成票据或自行做其它处理 */
var ticket = 'terror_1001_' + CaptchaAppId + Math.floor(new Date().getTime() / 1000);
callback({
ret: 0,
randstr: '@'+ Math.random().toString(36).substr(2),
ticket: ticket,
errorCode: 1001,
errorMessage: 'jsload_error',
```

```
});
}
```

回调内容

前端验证成功后，验证码会调用业务传入的回调函数，并在第一个参数中传入回调结果。结果字段说明如下：

字段名	值类型	说明
ret	Int	验证结果，0：验证成功。2：用户主动关闭验证码。
ticket	String	验证成功的票据，当且仅当 ret = 0 时 ticket 有值。
CaptchaAppId	String	验证码应用 ID。
bizState	Any	自定义透传参数。
randstr	String	本次验证的随机串，请求后台接口时需带上。
errorCode	Number	错误 code ，详情请参见 回调函数 errorCode 说明
errorMessage	String	错误信息

回调函数errorCode说明

errorCode	说明
1001	TCaptcha.js 加载错误
1002	调用 show 方法超时
1003	中间 js 加载超时
1004	中间 js 加载错误
1005	中间 js 运行错误
1006	拉取验证码配置错误/超时
1007	iframe 加载超时
1008	iframe 加载错误
1009	jquery 加载错误

errorCode	说明
1010	滑块 js 加载错误
1011	滑块 js 运行错误
1012	刷新连续错误3次
1013	验证网络连续错误3次

实例方法

TencentCaptcha 的实例提供一些操作验证码的常用方法：

方法名	说明	传入参数	返回内容
show	显示验证码，可以反复调用。	无	无
destroy	隐藏验证码，可以反复调用。	无	无
getTicket	获取验证码验证成功后的 ticket。	无	Object: {"CaptchaAppId":"","ticket":""}

配置参数

options 提供以下配置参数：

> ⚠ 注意：
>
> - 验证码弹窗内部不支持调整样式大小，如果需要调整，可在弹窗最外层用 class=tcaptcha-transform 的元素设置 transform:scale();。验证码更新可能会改变元素的 id，class 等属性，请勿依赖其他验证码元素属性值覆盖样式。
> - 如果手机原生端有设置左右滑动手势，需在调用验证码 show 方法前禁用，验证完成后再打开，防止与验证码滑动事件冲突。

配置名	值类型	说明
bizState	Any	自定义透传参数，业务可用该字段传递少量数据，该字段的内容会被带入 callback 回调的对象中。
enableDarkMode	Boolean,'force'	开启自适应深夜模式, 'force'将强制深夜模式。

配置名	值类型	说明
sdkOpts	Object	示例 {"width": 140, "height": 140} 仅支持移动端原生 webview 调用时传入，用来设置验证码loading加载弹窗的大小（并非验证码弹窗大小）。
ready	Function	验证码加载完成的回调，回调参数为验证码实际的宽高： {"sdkView": { "width": number, "height": number }} 该参数仅为查看验证码宽高使用，请勿使用此参数直接设定宽高。
needFeedBack	Boolean	隐藏帮助按钮。 示例 { needFeedBack: false }
userLanguage	String	指定验证码提示文案的语言，优先级高于后台配置，暂时仅支持滑块拼图验证码。支持传入值同 navigator.language 用户首选语言，大小写不敏感。详情请参见 userLanguage 配置参数。
type	String	定义验证码展示方式。 • popup（默认）弹出式，以浮层形式展示验证码。 • embed 嵌入式，以嵌入指定容器元素中的方式展示验证码。详情请参见 热点问题–验证码以嵌入式方式进行展示如何配置?。

userLanguage 配置参数

参数名	说明
zh-cn	简体中文
zh-hk	繁体中文（中国香港）
zh-tw	繁体中文（中国台湾）
en	英文
ar	阿拉伯语
my	缅甸语
fr	法语
de	德语

参数名	说明
he	希伯来语
hi	印地语
id	印尼语
it	意大利语
ja	日语
ko	朝鲜语
lo	老挝语
ms	马来语
pl	波兰语
pt	葡萄牙语
ru	俄语
es	西班牙语
th	泰语
tr	土耳其语
vi	越南语

热点问题

验证码以嵌入式方式进行展示如何配置?

- 手动初始化并绑定到一个元素，第一个参数为容器元素，将 options 提供的配置参数 type 设置为 embed 。

```
new TencentCaptcha(element, CaptchaAppId, callback, {type: 'embed'});
```

- 示例代码

```
<div id="tc"></div>
<script>
new TencentCaptcha(document.getElementById('tc'),CaptchaAppId,callbackName,{type:'em
bed'}).show()
```

```
//将验证码绑定到id为‘tc’的容器元素中
</script>
```

> **注意：**
> 如果使用嵌入式，容器元素在界面里找不到会报错。

更多信息

您可以登录 验证码控制台 ，在页面右上角单击**快速咨询**，了解更多详细信息。

App 端接入

最近更新时间：2022-04-07 14:47:25

前提条件

接入验证码前，进入图形验证 页完成新建验证。可在**验证列表**查看 验证码接入所需的 CaptchaAppId 及 AppSecretKey。

接入步骤

以下为 App 端接入流程，适用于每次都需要进行人机验证的场景（如登录、注册、下发短信、活动等），App（iOS 和 Android）皆使用 Web 前端 H5 方式进行接入。

Android 接入

Android 接入主要流程如下：

1. 在 Android 端利用 WebView 加载，需要接入滑动验证码组件的页面。
2. 通过 JS 调用代码，并把验证码 SDK 返回的参数值传到 Android App 的业务端。
3. Android 代码中获取票据后，把相关数据传入业务侧后端服务进行验证。

Android 接入的详细操作步骤如下：

1. 在项目的工程中，新建一个 Activity 并导入 WebView 组件所需的包。

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

2. 添加相关权限，如开启网络访问权限以及允许 App 进行非 HTTPS 请求等。

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

3. 在 Activity 的布局文件中，添加 WebView 组件。

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

4. 在项目的工程中，添加自定义 JavascriptInterface 文件，并定义一个方法用来获取相关数据。

```
import android.webkit.JavascriptInterface;
public class JsBridge {
@JavascriptInterface
public void getData(String data) {
System.out.println(data);
}
}
```

5. 在 Activity 文件中，加载相关 H5 业务页面。

```
public class MainActivity extends AppCompatActivity {
private WebView webview;
private WebSettings webSettings;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
initView();
}

private void initView() {
webview = (WebView) findViewById(R.id.webview);
webSettings = webview.getSettings();
webSettings.setUseWideViewPort(true);
webSettings.setLoadWithOverviewMode(true);
// 禁用缓存
webSettings.setCacheMode(WebSettings.LOAD_NO_CACHE);
```

```
webview.setWebViewClient(new WebViewClient(){
@Override
public boolean shouldOverrideUrlLoading(WebView view, String url) {
view.loadUrl(url);
return true;
}
});
// 开启js支持
webSettings.setJavaScriptEnabled(true);
webview.addJavascriptInterface(new JsBridge(), "jsBridge");
// 也可以加载本地html(webView.loadUrl("file:///android_asset/xxx.html"))
webview.loadUrl("https://x.x.x/x/");
}
}
```

6. 在 H5 业务页面中，集成验证码 SDK，并通过 JS 调用 SDK 获取验证码相关数据，最后使用 JSBridge 传回数据给具体业务端。

> ⚠ 注意：
> 如需隐藏验证码帮助按钮等功能，请参见 Web 前端接入 文档。

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Web 前端接入示例</title>
<!-- 验证码程序依赖(必须)。请勿修改以下程序依赖，如使用本地缓存，或通过其他手段规避加载，会影响程序的正常使用。 -->
<script src="https://ssl.captcha.qq.com/TCaptcha.js"></script>
</head>

<body>
<!--点击此元素会自动激活验证码，此例使用的button元素，也可以使用div、span等-->
<!--id：(不可变) 元素的 ID，值必须是 'TencentCaptcha'-->
```

```
<!--data-appid：(必须) 验证码CaptchaAppId，从腾讯云的验证码控制台中获取，验证码控制台页面内【图形验证】>【验证列表】进行查看 。如果未新建验证，请根据业务需求选择适合的验证渠道、验证场景进行新建-->
<!--data-cbfn：(必须) 回调函数名，函数名要与 data-cbfn 相同-->
<!--data-biz-state :(可选) 业务自定义透传参数，会在回调函数内获取到 （res.bizState）-->
<button id="TencentCaptcha" data-appid="你的验证码CaptchaAppId" data-cbfn="callbackName" data-biz-state="data-biz-state"
type="button">验证</button>
</body>

<script>
// 回调函数需要放在全局对象window下
window.callbackName = function (res) {
// 返回结果
// ret Int 验证结果，0：验证成功。2：用户主动关闭验证码。
// ticket String 验证成功的票据，当且仅当 ret = 0 时 ticket 有值。
// CaptchaAppId String 验证码应用ID。
// bizState Any 自定义透传参数。
// randstr String 本次验证的随机串，请求后台接口时需带上。
console.log('callback:', res);


// res（用户主动关闭验证码）= {ret: 2, ticket: null}
// res（验证成功） = {ret: 0, ticket: "String", randstr: "String"}
if (res.ret === 0) {
// 获取票据、随机数并调用App端注入的方法传入票据、随机数，进行后台票据校验
var result = { randstr:res.randstr, ticket:res.ticket };
window.jsBridge.getData(JSON.stringify(result));
}
}
</script>
</html>
```

iOS 接入

iOS 接入主要流程如下：

1. 在 iOS 中打开 WebView，通过 JSBridge 触发 HTML 页面 ，同时注入方法，供 HTML 调用传入票据结果。
2. 在 HTML 页面中，接入示例代码，滑动验证码后，需要在回调函数中判断票据，并调用 iOS 注入的方法传入票据结果。
3. 需要回调数据通过 JSBridge 返回到 iOS，需要把票据传入业务侧后端服务。

iOS 接入的详细操作步骤如下：

1. 在控制器或 view 中导入 WebKit 库。

```
#import <WebKit/WebKit.h>
```

2. 创建 WebView 并渲染。

```
-(WKWebView *)webView{
if(_webView == nil){
//创建网页配置对象
WKWebViewConfiguration *config = [[WKWebViewConfiguration alloc] init];
// 创建设置对象
WKPreferences *preference = [[WKPreferences alloc]init];
//设置是否支持 javaScript 默认是支持的
preference.javaScriptEnabled = YES;
// 在 iOS 上默认为 NO，表示是否允许不经过用户交互由 javaScript 自动打开窗口
preference.javaScriptCanOpenWindowsAutomatically = YES;
config.preferences = preference;
//这个类主要用来做 native 与 JavaScript 的交互管理
WKUserContentController * wkUController = [[WKUserContentController alloc] init];
//注册一个name为jsToOcNoPrams的js方法 设置处理接收JS方法的对象
[wkUController addScriptMessageHandler:self name:@"jsToOcNoPrams"];
[wkUController addScriptMessageHandler:self name:@"jsToOcWithPrams"];
config.userContentController = wkUController;
_webView = [[WKWebView alloc] initWithFrame:CGRectMake(0, 0, SCREEN_WIDTH, SCREEN_
HEIGHT) configuration:config];
// UI 代理
_webView.UIDelegate = self;
// 导航代理
_webView.navigationDelegate = self;
//此处即需要渲染的网页
NSString *path = [[NSBundle mainBundle] pathForResource:@"JStoOC.html" ofType:nil];
```

```
NSString *htmlString = [[NSString alloc]initWithContentsOfFile:path encoding:NSUTF8StringEncoding error:nil];
[_webView loadHTMLString:htmlString baseURL:[NSURL fileURLWithPath:[[NSBundle mainBundle] bundlePath]]];
}
return _webView;
}
[self.view addSubview:self.webView];
```

3. 代理方法，处理一些响应事件。

```
// 页面开始加载时调用
-(void)webView:(WKWebView *)webView didStartProvisionalNavigation:(WKNavigation *)navigation {
}
// 页面加载失败时调用
-(void)webView:(WKWebView *)webView didFailProvisionalNavigation:(null_unspecified WKNavigation *)navigation withError:(NSError *)error {
[self.progressView setProgress:0.0f animated:NO];
}
// 当内容开始返回时调用
-(void)webView:(WKWebView *)webView didCommitNavigation:(WKNavigation *)navigation {
}
// 页面加载完成之后调用
-(void)webView:(WKWebView *)webView didFinishNavigation:(WKNavigation *)navigation {
[self getCookie];
}
//提交发生错误时调用
-(void)webView:(WKWebView *)webView didFailNavigation:(WKNavigation *)navigation withError:(NSError *)error {
[self.progressView setProgress:0.0f animated:NO];
}
// 接收到服务器跳转请求即服务重定向时之后调用
-(void)webView:(WKWebView *)webView didReceiveServerRedirectForProvisionalNavigation:(WKNavigation *)navigation {
}
```

4. JS 将参数传给 OC。

```
<p style="text-align:center"> <button id="btn2" type = "button" onclick = "jsToOcFunction
()"> JS调用OC：带参数 </button> </p>
function jsToOcFunction()
{
window.webkit.messageHandlers.jsToOcWithPrams.postMessage({"params":"res.randstr"});
}
```

5. 将渲染好的 WebView 展示在视图上，调用验证码服务，将数据传给客户端。

> ⚠ 注意：
> 如需隐藏验证码帮助按钮等功能，请参见 Web 前端接入 文档。

```
-(void)userContentController:(WKUserContentController *)userContentController didReceiveS
criptMessage:(WKScriptMessage *)message{
此处message.body即传给客户端的json数据
//用message.body获得JS传出的参数体
NSDictionary * parameter = message.body;
//JS调用OC
if([message.name isEqualToString:@"jsToOcWithPrams"]){
//在此处客户端得到js透传数据 并对数据进行后续操作
parameter[@"params"]
}
}
```

> ⚠ 注意：
> 验证码客户端接入完成后，验证码后台需二次核查验证码票据结果，请进行 后台 API 接入 操作，确保验证安全性。更多详情请参见 核查验证码票据文档 。

热点问题

Android 使用 Web 前端 H5 方式进行接入，调试过程中先弹出空白背景，后弹出验证码页面如何调整?

- 调试过程中，正常情况下会首先调起 webview 加载网页，然后弹出验证码页面。
- 如果出现先弹出空白背景，后弹出图形验证页面的现象。形成原因如下：

- 加载验证码 js 的时间导致白屏。
- 空白层形成原因是页面没有内容时，加载的 webview 就显示出来，需要等待 ready 事件触发后再进行 webview 展示。

- 因此，Android 需要先加载页面但不进行展示，等待 ready 回调后，再通知 Android 进行展示。ready 配置说明，请参见 Web 前端接入-配置参数 文档。

```
options={ready: function(size){
// 与Android通信
}}
new TencentCaptcha(appId, callback, options);
```

App 端接入验证码显示不完整如何调整?

验证码根据容器宽高进行居中显示，验证码显示不完整可能由于容器本身设置较宽，导致展示的验证码被截断，该情况需要对客户端的弹框进行调整。此外随意加载其他 webview 都可能会出现截断的情况。

更多信息

您可以登录 验证码控制台 ，在页面右上角单击**快速咨询**，了解更多详细信息。

小程序插件接入

最近更新时间：2022-04-08 14:58:20

前提条件

接入验证码前，进入图形验证 页完成新建验证。可在**验证列表**查看 验证码接入所需的 CaptchaAppId 以及 AppSecretKey。

小程序原生语言接入

示例下载

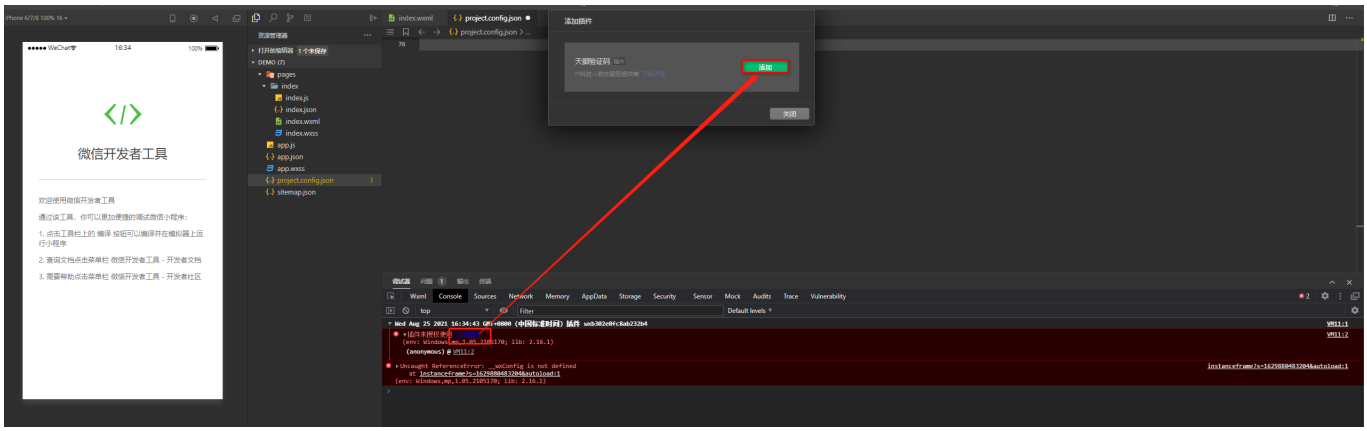
完整示例请下载：小程序验证码接入示例。

> **注意：**
> 请勿在“微信开发者工具”的“游客模式”下接入验证码。

步骤1：添加插件

1. 用管理员身份登录 微信公众平台，且需使用接入小程序的相关账号。
2. 小程序的相关账号主体账号有两种类型，分别为非个人和个人，详细接入操作如下。
   - 主体类型“非个人”（政府、媒体、企业等）小程序的账号，选择**设置** > **第三方设置** > **添加插件**，在搜索框内输入关键字“天御验证码”查找插件，并单击**添加**，如下图所示：

- 主体类型“个人”的小程序的账号，请先下载“小程序验证码接入示例”，使用“微信开发者工具”导入项目，可通过“插件未授权使用”信息，单击**添加插件 > 添加**，选择“天御验证码”小程序插件。



步骤2：集成插件

1. 引入验证码小程序插件。

   使用验证码插件前，需要在 app.json 中声明验证码小程序插件，如下：

```
{
"plugins": {
"myPlugin": {
"version": "1.0.3",
"provider": "wxb302e0fc8ab232b4"
```

```
}
}
}
```

2. 引入验证码小程序组件。

需要在页面.json文件中需要引入自定义组件，js 代码如下：

```
{
"usingComponents": {
"t-captcha": "plugin://myPlugin/t-captcha"
}
}
```

步骤3：使用小程序插件

1. 使用原生小程序语言接入时，需要在自定义的 .wxml 文件中，使用验证码插件，wxml 代码如下：

```
<!-- app-id：验证码CaptchaAppId，从腾讯云的验证码控制台中获取，在验证码控制台页面内【图形验证】>【验证列表】进行查看 -->
<t-captcha
id="captcha"
app-id="小程序插件验证码CaptchaAppId"
bindverify="handlerVerify"
bindready="handlerReady"
bindclose="handlerClose"
binderror="handlerError" />
<button bindtap='login'>登录</button>
```

- **组件参数说明：**

字段名	值类型	默认值	说明
CaptchaAppId	String	无	验证码应用ID
size	String	normal	尺寸，可选 normal、small、mini
lang	String	zh-CN	语言，可选 zh-CN、zh-TW、en
themeColor	String	#1A79FF	主题色

- **组件事件说明：**

事件名	参数	说明
ready	无	验证码准备就绪
verify	{ret, ticket}	验证码验证完成
close	{ret}	验证码弹框准备关闭
error	无	验证码配置失败

- **组件方法说明：**

方法名	说明
show	展示验证码
destroy	销毁验证码
refresh	重置验证码

2. 在自定义的 .js 文件中，监听事件，代码如下：

```
Page({
data: {},
login: function () {
this.selectComponent('#captcha').show()
// 进行业务逻辑，若出现错误需重置验证码，执行以下方法
// if (error) {
// this.selectComponent('#captcha').refresh()
// }
},
// 验证码验证结果回调
handlerVerify: function (ev) {
// 如果使用了 mpvue，ev.detail 需要换成 ev.mp.detail
if(ev.detail.ret === 0) {
// 验证成功
console.log('ticket:', ev.detail.ticket)
} else {
// 验证失败
// 请不要在验证失败中调用refresh，验证码内部会进行相应处理
}
```

```
},
// 验证码准备就绪
handlerReady: function () {
console.log('验证码准备就绪')
},
// 验证码弹框准备关闭
handlerClose: function (ev) {
// 如果使用了 mpvue，ev.detail 需要换成 ev.mp.detail,ret为0是验证完成后自动关闭验证码弹窗，ret为2是用户主动点击了关闭按钮关闭验证码弹窗
if(ev && ev.detail.ret && ev.detail.ret === 2){
console.log('点击了关闭按钮，验证码弹框准备关闭');
} else {
console.log('验证完成，验证码弹框准备关闭');
}
},
// 验证码出错
handlerError: function (ev) {
console.log(ev.detail.errMsg)
}
})
```

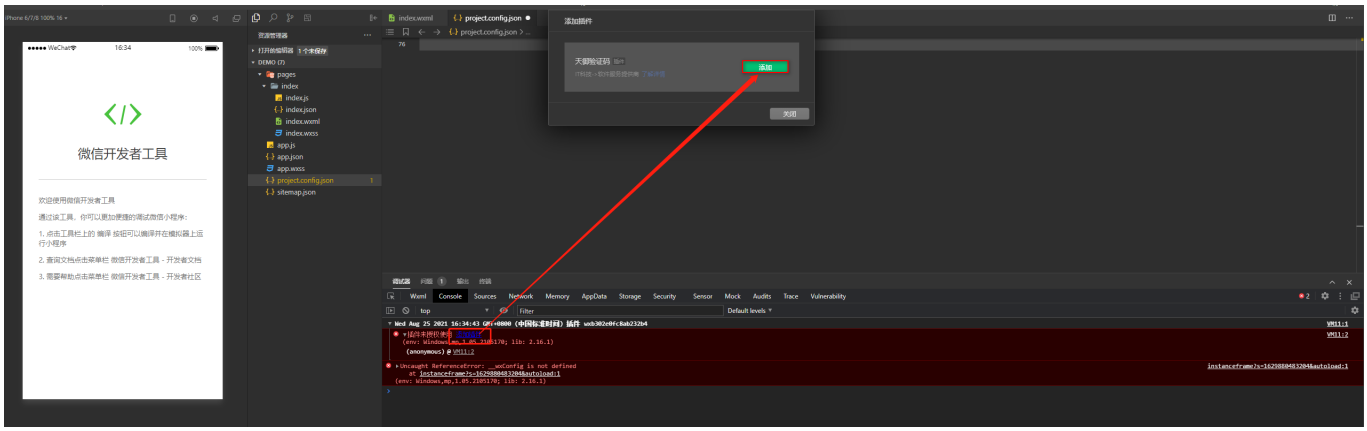
> ⚠ **注意：**
>
> 验证码客户端接入完成后，验证码后台需二次核查验证码票据结果，请进行 后台 API 接入 操作，确保验证安全性。更多详情请参见 核查验证码小程序插件票据结果 文档。

uni-app 前端框架接入

步骤1：添加插件

1. 用管理员身份登录 微信公众平台，且需使用接入小程序的相关账号。
2. 小程序的相关账号主体账号有两种类型，分别为非个人和个人，详细接入操作如下。
   - 主体类型“非个人”（政府、媒体、企业等）小程序的账号，选择**设置** > **第三方设置** > **添加插件**，在搜索框内输入关键字“天御验证码”查找插件，并单击**添加**，如下图所示：

- 主体类型“个人”的小程序的账号，请先下载“小程序验证码接入示例”，使用“微信开发者工具”导入项目，可通过“插件未授权使用”信息，单击**添加插件 > 添加**，选择“天御验证码”小程序插件。



步骤2：集成插件

1. 引入验证码小程序插件。

   使用验证码插件前，需要在 app.json 中声明验证码小程序插件，如下：

```
{
"plugins": {
"myPlugin": {
"version": "1.0.3",
"provider": "wxb302e0fc8ab232b4"
```

```
}
}
}
```

2. 引入验证码小程序组件。

需要在页面.json文件中需要引入自定义组件，js 代码如下：

```
{
"usingComponents": {
"t-captcha": "plugin://myPlugin/t-captcha"
}
}
```

步骤3：使用小程序插件

1. 使用 uni-app 框架接入时，需要在自定义的 .vue 中使用验证码插件，代码如下：

```
<!-- app-id：验证码CaptchaAppId，从腾讯云的验证码控制台中获取，在验证码控制台页面内【图形验证】>【验证列表】进行查看 -->
<t-captcha
id="captcha"
app-id="小程序插件验证码CaptchaAppId"
@verify="handlerVerify"
@ready="handlerReady"
@close="handlerClose"
@error="handlerError" />
<button @click="login">登录</button>
```

- 组件参数说明：

字段名	值类型	默认值	说明
CaptchaAppId	String	无	验证码应用 ID
size	String	normal	尺寸，可选 normal、small、mini
lang	String	zh-CN	语言，可选 zh-CN、zh-TW、en
themeColor	String	#1A79FF	主题色

- **组件事件说明：**

事件名	参数	说明
ready	无	验证码准备就绪
verify	{ret, ticket}	验证码验证完成
close	{ret}	验证码弹框准备关闭
error	无	验证码配置失败

- **组件方法说明：**

方法名	说明
show	展示验证码
destroy	销毁验证码
refresh	重置验证码

2. 在自定义的 .vue 文件中，监听事件，代码如下：

```
methods:{
login: function () {
this.selectComponent('#captcha').show()
// 进行业务逻辑，若出现错误需重置验证码，执行以下方法
// if (error) {
// this.selectComponent('#captcha').refresh()
// }
},
// 验证码验证结果回调
handlerVerify: function (ev) {
// 如果使用了 mpvue，ev.detail 需要换成 ev.mp.detail
if(ev.detail.ret === 0) {
// 验证成功
console.log('ticket:', ev.detail.ticket)
} else {
// 验证失败
// 请不要在验证失败中调用refresh，验证码内部会进行相应处理
}
```

```
},
// 验证码准备就绪
handlerReady: function () {
console.log('验证码准备就绪')
},
// 验证码弹框准备关闭
handlerClose: function (ev) {
// 如果使用了 mpvue，ev.detail 需要换成 ev.mp.detail,ret为0是验证完成后自动关闭验证码弹窗，ret为2是用户主动点击了关闭按钮关闭验证码弹窗
if(ev && ev.detail.ret && ev.detail.ret === 2){
console.log('点击了关闭按钮，验证码弹框准备关闭');
} else {
console.log('验证完成，验证码弹框准备关闭');
}
},
// 验证码出错
handlerError: function (ev) {
console.log(ev.detail.errMsg)
}
}
```

注意：

验证码客户端接入完成后，验证码后台需二次核查验证码票据结果，请进行 后台 API 接入 操作，确保验证安全性。更多详情请参见 核查验证码小程序插件票据结果 文档。

更多信息

您可以登录 验证码控制台 ，在页面右上角单击**快速咨询**，了解更多详细信息。