

Captcha Integration Guide Product Documentation





Copyright Notice

©2013-2025 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

🔗 Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.



Contents

Integration Guide

Client Integration

Web Integration

HTML5 Integration for App Client

Android Client SDK Integration

Integration of IOS Client SDK

SDK Download

Server Integration

Integration to Ticket Verification (Web and App)

Integration Guide Client Integration Web Integration

Last updated : 2025-02-21 17:04:10

Prerequisites

Before client integration, you need to create a new verification and obtain the required CaptchaAppId and AppSecretKey from the **Validation List**. The steps are as follows:

1. Log in to the Captcha Console, select Verification Management in the left sidebar, and enter the Verification Management page.

2. Click Create Captcha, and set parameters such as verification name according to business scenario needs.

3. Click **Yes** to complete the creation of a new Captcha, and then you can view the CaptchaAppId and AppSecretKey in the Validation List.

Note:

Starting from December 2024, Captcha will offer two versions of the service. The architecture of the old version's user console remains unchanged, and the user experience is not affected. Customers using the old version who wish to use the new version need to create a new CaptchaAppid to ensure the smooth use of the Captcha feature in the future. If you have any queries, feel free to contact us.

Sample Code

In the following example code, click **Verification** to activate the Captcha and display the verification result in a popup.

Note:

This example does not show the logic of calling the invoice verification API. After the business client completes the Captcha integration, the business server needs to perform a secondary check on the Captcha ticket result (without invoice verification, it will allow the black industry to easily forge verification results, losing the human robot confrontation effect of Captcha). For details, see: Integrate Invoice Verification (Web and App).

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

<head>

```
Captcha
```

```
<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 Frontend Integration Example</title>
    <!-- Captcha program dependency (required). Do not modify the following
program dependencies. If loading is evaded by other means, the captcha may not
update properly, its anti-attack capability cannot be guaranteed, and
misinterception may even occur. -->
    <script src="https://ca.turing.captcha.qcloud.com/TJNCaptcha-global.js">
</script>
</head>
<body>
    <div id="cap_iframe"></div>
    <button id="CaptchaId" type="button">Verification</button>
</body>
<script>
    // Define a callback function
    function callback(res) {
        // The first parameter is the callback result, as shown below:
        // ret
                               Verification result, 0: Verification
                      Int
successful. 2: User actively closes the Captcha.
        // ticket
                      String The invoice for successful validation, which
has a value only when ret = 0.
        // CaptchaAppId
                        String
                                      Captcha application ID.
       // bizState Any
                               Custom pass-through parameters.
        // randstr
                     String
                               Random string for this verification, which
needs to be passed during subsequent invoice verification.
       console.log('callback:', res);
       // res (user actively turns off Captcha) = {ret: 2, ticket: null}
        // res (successfully validated) = {ret: 0, ticket: "String", randstr:
"String" }
       // res (error occurred when requesting Captcha, and a disaster recovery
ticket with the prefix "terror_" is automatically returned) = {ret: 0, ticket:
"String", randstr: "String", errorCode: Number, errorMessage: "String"}
        // This code is only an example of showing the verification result. For
real business integration, it is recommended to handle different businesses
based on the ticket and errorCode situations.
       if (res.ret === 0) {
            // Copy the result to the clipboard
           var str = '[randstr]->[' + res.randstr + '] [ticket]->[' +
res.ticket + ']';
           var ipt = document.createElement('input');
```

```
ipt.value = str;
            document.body.appendChild(ipt);
            ipt.select();
            document.execCommand("Copy");
            document.body.removeChild(ipt);
            alert('1. The return result (randstr, ticket) has been copied to
the clipboard, press ctrl+v to view.\\n2. Open the browser console to view the
full return result.');
       }
    }
    // Define Captcha JS loading error handling function
    function loadErrorCallback() {
      var appid = 'CaptchaAppId';
       // Generate disaster recovery tickets or perform other processing as
needed
      var ticket = 'terror_1001_' + appid + '_' + Math.floor(new
Date().getTime() / 1000);
      callback({
        ret: 0,
        randstr: '@'+ Math.random().toString(36).substr(2),
        ticket,
        errorCode: 1001,
        errorMessage: 'jsload_error',
     });
    }
    // Define Captcha trigger event
    window.onload = function() {
        document.getElementById('CaptchaId').onclick = function() {
            try {
                  // Generate a Captcha object
                  // CaptchaAppId: Log in to the Captcha console and view it
from the [Verification management] page. If no verification has been created,
please create one first.
                  //callback: Defined callback function
                  `var captcha = new
TencentCaptcha(document.getElementById('cap_iframe'), 'Your CaptchaAppId',
callback, {});`
                  // Calling method to display the Captcha
                  captcha.show();
            } catch (error) {
            // Loading exception, call Captcha JS loading error handling
function
                  loadErrorCallback();
            }
        }
```



} </script> </html>

Connection Instructions

Step 1: Dynamically Import Captcha JS

The web page needs to dynamically import Captcha JS, and when verification is required, the Captcha is invoked for verification.

```
<!-- Example of Dynamically Importing Captcha JS --> <script src="https://ca.turing.captcha.qcloud.com/TJNCaptcha-global.js"> </script>
```

Note:

Captcha JS must be dynamically loaded. If dynamic loading is evaded by other means, the captcha may not update properly, its anti-attack capability cannot be guaranteed, and may even occur.

Step 2: Create a Captcha Object

After introducing Captcha JS, a TencentCaptcha class will be globally registered. The business side can use this class to initialize the captcha by itself and display or hide the captcha.

Note:

The element that triggers the captcha should not use <code>id="TencentCaptcha"</code>. TencentCaptcha is a system default id and cannot be used.

Constructor

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

Parameter Description

Parameter Name	Value Type	Description
DOM	Element Nodes	Container that handles the Robot checkbox.
CaptchaAppId	String	CaptchaAppld: Log in to the Captcha Console, and view it on the Verification Management page. If no verification has been created, please create one first. Note: Do not use CaptchaAppld with the client type of mini program, as it may lead to data statistics errors.



callback	Function	Captcha callback function. For details, see callback function.
options	Object	Captcha appearance configuration parameters. For details, see options appearance configuration parameters.

Callback Function

After verification, the callback function passed in by the business will be called, and the callback result will be passed in as the first parameter. The fields of the callback result are described as follows:

Field Name	Value Type	Description
ret	Int	Verification result, 0: Verification successful. 2: User actively closes the Captcha.
ticket	String	The invoice for successful validation, which has a value only when $ret = 0$.
CaptchaAppId	String	Captcha Application ID.
bizState	Any	Custom transparent transmission parameters.
randstr	String	The random string for this verification, which needs to be passed during subsequent invoice verification.
errorCode	Number	Error code. For details, see callback function error code description.
errorMessage	String	Error information.

Callback Function Error Code Description

ErrorCode	Description
1001	Loading error of TJNCaptcha - global.js.
1002	Call to the show method times out.
1003	Loading timeout of intermediate JS
1004	Loading error of intermediate JS.
1005	Execution error of intermediate JS
1006	Pull Captcha configuration error/timeout
1007	Iframe loading timeout

1008	Iframe loading error
1009	Loading error of jQuery.
1010	Loading error of slider JS.
1011	Slider JS execution error
1012	Refresh three consecutive errors
1013	Network verification failed three consecutive times.

Appearance Configuration Parameters For Options

The options parameter is used to customize the appearance settings of the Captcha, which can be set to empty by default.

Note:

The style size inside the Captcha popup cannot be adjusted. If adjustment is needed, you can set

transform:scale(); with the element of class=tcaptcha-transform at the outermost layer of the popup. The update of the Captcha may change the attributes such as the id and class of the element. Do not rely on the attribute values of other Captcha elements to override the style.

If the mobile phone native end has a left and right swipe gesture setting, it needs to be disabled before calling the Captcha show method and enabled after verification is completed to prevent conflicts with the Captcha swipe event.

Configuration Name	Value Type	Description
bizState	Any	Custom pass-through parameters. The business can use this field to transmit a small amount of data, and the content of this field will be carried into the object of the callback.
enableDarkMode	Boolean/String	Enable adaptive late-night mode or force late-night mode. Enable adaptive late-night mode: {"enableDarkMode": true} Force late-night mode: {"enableDarkMode": 'force'}
ready	Function	The callback when the Captcha loading is completed, and the callback parameter is the actual width and height of the Captcha: {"sdkView": { "width": number, "height": number }} This parameter is only used for viewing the width and height of the Captcha, please do not use this parameter to directly set the width and height.
needFeedBack	String	Custom help link: {"needFeedBack": 'URL address'}
enableAutoCheck	Boolean	After it is enabled, the verification can be automatically checked for secure requests without user operation, and it is closed by default.

		Enable: {"enableAutoCheck": true}
userLanguage	String	Specify the language of the Captcha prompt text, which has a higher priority than the console configuration. Supports input values the same as navigator.language user preferred language, case-insensitive. For details, see userLanguage configuration parameter.
type	String	Define the Captcha display method. Popup (default): The Captcha is displayed in a floating layer that pops up and centers. Embed: Display the Captcha by embedding it into a specified container element.
aidEncrypted	String	CaptchaAppId encrypted verification string, optional parameter. For details, see CaptchaAppid Encryption Verification Capability Integration Guide.
showFn	Function	Duration of rendering time + SID callback function.

userLanguage configuration parameter

The userlanguage parameter can be set to empty, which means it defaults to adapting to the browser language.

Parameter Name	Description
zh-cn	Simplified Chinese
zh-hk	Traditional Chinese (Hong Kong (China))
zh-tw	Traditional Chinese (Taiwan (China))
en	English
ar	Arabic
my	Burmese
fr	French
de	German
he	Hebrew
hi	Hindi

id	Indonesian
it	Italian
ја	Japanese
ko	Korean
lo	Lao
ms	Malay
рІ	Polish
pt	Portuguese
ru	Russian
es	Spanish
th	Thai
tr	Turkish
vi	Vietnamese

Step 3: Invoke the Captcha Instance Method

TencentCaptcha instances provide some common methods for operating Captchas:

Method Name	Description	Input Parameter	Returned Content
show	Display the Captcha, which can be called repeatedly.	No	No
destroy	Hide the Captcha, which can be called repeatedly.	No	No
getTicket	Obtain the ticket after successful verification.	No	<pre>Object: {"CaptchaAppId":"","ticket":""}</pre>
reload	Reinitialize the checkbox, which can be called repeatedly.	No	No

Step 4: Disaster Recovery

To ensure that the customer's website business process is not blocked when the Captcha server-side exception

occurs, it is recommended to integrate the Captcha in the following way.

1. Define JS loading error handling function.

```
// Function of the error handling function: Ensure the normal event process when JS
// Function definition needs to be before script loading
function loadErrorCallback() {
  var appid = ''
  // Generate disaster recovery tickets or handle other tasks as needed
  var ticket = 'terror_1001_' + appid + Math.floor(new Date().getTime() / 1000);
  callback({
    ret: 0,
    randstr: '@'+ Math.random().toString(36).substr(2),
    ticket,
    errorCode: 1001,
    errorMessage: 'jsload_error',
  });
}
```

2. When a Captcha instance catches an error, call the JS loading error handling function.

```
try {
    // Generate a Captcha object
    `var captcha = new TencentCaptcha(document.getElementById('cap_iframe'), 'Your Cap
    // Calling method to display the Captcha
    captcha.show();
} catch (error) {
    // Loading exception, call Captcha JS loading error handling function
    loadErrorCallback();
}
```

3. When defining the Captcha callback function, handle it according to the ticket and errorCode (instead of ret). For the definition of errorCode, refer to "Callback Function errorCode Description" in Step 2.

```
function callback(res) {
    // res (user actively turns off Captcha) = {ret: 2, ticket: null}
    // res (successfully validated) = {ret: 0, ticket: "String", randstr: "String"}
    // res (request error, returning a disaster recovery ticket with the prefix "terro
    if (res.ticket) {
        // Perform special handling based on the errorCode situation
        if (res.errorCode === xxxxx) {
            //Customize disaster recovery logic (for example, skip this verification)
        }
    }
}
```

Complete disaster recovery solution. For details, see Business Disaster Recovery Solution.

Note:

After the business client completes the Captcha integration, the server needs to perform a secondary check on the Captcha invoice result (without invoice verification, it will be easy for the black industry to forge the verification result, losing the effect of human robot confrontation of Captcha). For details, see Integrate Invoice Verification (Web and App).

CaptchaAppid Encryption Verification Capability Access Guide

By passing an encrypted string through the frontend (optional capability, can be selectively integrated based on security requirements), it can effectively prevent resource fraud caused by the leakage of CaptchaAppid.

Encryption Rules

The API supports passing the verification code business CaptchaAppid in encryption mode for verification through the aidEncrypted parameter (i.e., the encrypted string identifier).

When the console's forced verification is enabled, the encryption mode is triggered. The customer's server encrypts it and sends the corresponding encrypted string to the customer's frontend, which then passes the encrypted string as a parameter to the Captcha side.

The encryption procedure is as follows:

1. Log in to the Captcha Console, in the left sidebar, select Verification Management > Captcha Details, click Integration.

2. On the Manage CAPTCHA page, select the required AppSecretKey as the key. When the key is less than 32 bytes, cycle fill the same AppSecretKey to make up to 32 bytes as the key.



3. Randomly generate a 16-byte IV. Combine the obtained key Key in step 1 to perform AES256 encryption on the business data object. The encryption mode is CBC/PKCS7Padding. The encrypted business data object is the verification code business CaptchaAppid & timestamp & Encrypted Text Expiry Time. The timestamp is the current Unix timestamp at the second level and cannot be a future time. The unit of Encrypted Text Expiry Time is seconds, and the maximum value is 86400 seconds. The encrypted string obtained is CaptchaAppidEncrypted.

4. Perform Base64 encoding on the byte array CaptchaAppidEncrypted obtained by AES256 encryption of IV in step 2, that is, Base64 (IV + CaptchaAppidEncrypted), with no hyphen in the middle, and the final encrypted business parameter request string obtained is aidEncrypted.

5. Regarding the validity time and expiration time of Captcha encryption processing, customers can set them according to their own needs. It should be noted here that the timing mechanism starts when the Captcha begins to load. If the set timing duration is too short, the following situation may occur: After the Captcha is loaded, the user does not perform any operations for a period of time, and when the user performs related operations later, the encryption information corresponding to the Captcha has expired and become invalid.

Туре	Sample Value
Captchaappid	123456789
Timestamp	1710144972
Expiration Time	86,400 sec
iv	0123456789012345

Example Of Encryption Result



AppSecretKey	1234567891011121314151516
Recursively filled key	12345678910111213141515161234567
Encrypted business data object.	123456789&1710144972&86400
The final encrypted string aidEncrypted	MDEyMzQ1Njc4OTAxMjM0NWvZ11atw+1uzYmolyt5rAQVPyMK9ZDavskPw5hcayeT

Sample Code

Server-side encryption

The encryption rule is: Base64(IV + AES256(CaptchaAppid & timestamp & Encrypted Text Expiry Time, IV, Key)), that is, using a randomly generated 16-byte IV and a 32-byte encryption Key obtained by cyclic filling according to AppSecretKey, using AES256 algorithm encryption mode CBC/PKCS7Padding to encrypt the plaintext data CaptchaAppid & timestamp & Encrypted Text Expiry Time.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import base64
import time
def encrypt(plaintext, key, iv):
   cipher = AES.new(key, AES.MODE_CBC, iv) # Create a new AES cipher in CBC
mode
    ciphertext = cipher.encrypt(pad(plaintext.encode(), AES.block_size)) # Pad
the data and then encrypt it. pad(plaintext.encode(), AES.block_size) checks
whether the plaintext length is a multiple of 16 bytes. If not, it will pad the
plaintext to a multiple of 16 bytes using the PKCS7 padding method.
    ciphertextBase64 = base64.b64encode(iv + ciphertext).decode('utf-8') #
Concatenate iv with the encrypted data and perform Base64 encoding before
transmission.
    return ciphertextBase64
# Example of Encryption
AppSecretKey = b'1234567891011121314151516' # The customer obtains the
AppSecretKey under the corresponding Captcha account from the console, 25
characters.
remainder = 32 % len(AppSecretKey) # Calculate the key length to be
supplemented
```



```
key = AppSecretKey + AppSecretKey[:remainder]  # Final encryption key,
padded to 32 bits.
CaptchaAppId = "123456789" # Customer's own Captcha application ID
curTime = 1710144972  # Obtain the current timestamp; for the example, it is
temporarily set to a fixed timestamp. The customer should set it to the latest
timestamp, using int(time.time())
expireTime = 86400 # Expiration time setting, temporarily set to this value
here, the customer should set it according to their needs.
plaintext = CaptchaAppId + "&" + str(curTime) + "&" + str(expireTime)
                                                                      #
Concatenate the business data object to be encrypted, CaptchaAppId & timestamp
& Encrypted Text Expiry Time
iv = "0123456789012345".encode() # Randomly generate a 16-byte IV; for now, it
is set to this value. Customers should use randomly generated data, using
os.urandom(16)
ciphertext = encrypt(plaintext, key, iv) # Encrypt
print("Ciphertext (Base64):", ciphertext) # The sample data in this example
will output MDEyMzQ1Njc4OTAxMjM0NWvZ11atw+1uzYmoIyt5rAQVPyMK9ZDavskPw5hcayeT
```

Frontend Integration Example

```
const encryptAppid = async () => {
  /** Get the encrypted appid string from the backend */
 const { aidEncrypted } = await fetch('/api/encryptAppid');
  /** Callback function */
  const callBack = (ret) => {
   console.log('ret', ret);
  };
  /** Error Callback Function */
  const errorCb = (error) => {
   console.log('error', error);
  };
 try {
    /** Pass the obtained encrypted string to the aidEncrypted parameter */
    const captcha = new TencentCaptcha('123456789', callBack, { aidEncrypted:
aidEncrypted });
   captcha.show();
  } catch (error) {
   errorCb(error);
  }
```



};

FAQs

For details, see Integration-related Issues.

HTML5 Integration for App Client

Last updated : 2025-02-28 11:35:26

Prerequisites

Before you can integrate Captcha into a client, you need to create a CAPTCHA and obtain its CaptchaAppId and AppSecretKey from the **CAPTCHA list**. Follow the steps below:

1. Log in to the Captcha console and select **Manage CAPTCHA** in the left navigation pane to enter the CAPTCHA management page.

2. Click **Create CAPTCHA** and set parameters such as CAPTCHA name according to your business requirements.

3. Click **OK** to complete the creation. You can view its CaptchaAppId and AppSecretKey in the CAPTCHA list.

Integration steps

Note:

You can integrate Captcha into an (Android/iOS) app only by using a WebView to display H5 pages.

Integration in Android

General process

1. Use a WebView to display H5 pages in an Android app. For more information on how to integrate Captcha into H5 pages, please see Web Integration.

2. Call the CAPTCHA JS to render the verification page in an H5 page. Then pass the parameter values returned in JS to the Android app's business client.

3. The Android app's business client passes parameters such as ticket and random number to the business server for ticket verification. For more information, please see Integration to Ticket Verification (Web and App).

Detailed steps

1. In a project, create an activity and import the WebView packages required.

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

2. Add permissions, such as "Enable network access" and "Allow the app to make non-HTTPS requests".

<uses-permission android:name="android.permission.INTERNET"/>

<application android:usesCleartextTraffic="true">...</application>

3. Add the WebView in the layout file of the activity.

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

4. In the project, add a custom JavascriptInterface file and define a method to obtain data.

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

5. In the activity file, load the H5 business page.

```
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);
// Disable cache
webSettings.setCacheMode(WebSettings.LOAD_NO_CACHE);
webview.setWebViewClient(new WebViewClient() {
@Override
public boolean shouldOverrideUrlLoading(WebView view, String url) {
view.loadUrl(url);
return true;
}
});
```

S Tencent Cloud

```
// Enable js
webSettings.setJavaScriptEnabled(true);
webview.addJavascriptInterface(new JsBridge(), "jsBridge");
// Or load a local html file
(webView.loadUrl("file:///android_asset/xxx.html"))
webview.loadUrl("https://x.x.x/x/");
}
```

6. Integrate Captcha in the H5 business page (for more information, please see Web Integration), and use JSBridge to pass the verification data to the business side.

Note:

After Captcha is integrated into the business client, the business server needs to verify the CAPTCHA ticket (if ticket verification is not integrated, the black market can easily forge verification results, which defeats the purpose of human verification via CAPTCHAs). For more information, please see Integration to Ticket Verification (Web and App).

Integration in iOS

General process

1. Open a WebView in iOS, load an HTML page using JSBridge, and inject a method to be called in the HTML page to pass verification results.

2. Integrate Captcha in the HTML page, call the CAPTCHA JS to render the verification page, and then call the injected method to pass the verification result. For more information, please see Web Integration.

3. Return the verification result to iOS using JSBridge, and pass parameters such as ticket and random number to the business server for ticket verification. For more information, please see Integration to Ticket Verification (Web and App).

Detailed steps

1. In a controller or view, import the WebKit library.

#import <WebKit/WebKit.h>

2. Create a WebView and render it.

```
-(WKWebView *)webView{
if(_webView == nil){
    // Create a webpage configuration object
    WKWebViewConfiguration *config = [[WKWebViewConfiguration alloc] init];
    // Create a settings object
    WKPreferences *preference = [[WKPreferences alloc]init];
    // Set "Whether javaScript is supported". The value is YES by default.
    preference.javaScriptEnabled = YES;
    // Set "Allow javaScript to open a window without user interaction". In iOS,
    the value is set to NO by default.
```

```
preference.javaScriptCanOpenWindowsAutomatically = YES;
config.preferences = preference;
// This class is used to manage interactions between native and JavaScript.
WKUserContentController * wkUController = [[WKUserContentController alloc]
init];
// Register a js method with the name jsToOcNoPrams. Set objects to handle and
receive the JS method.
 [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 delegate
_webView.UIDelegate = self;
// Navigation delegate
_webView.navigationDelegate = self;
// Webpage to be rendered
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. Set delegate methods to process some response events.

```
// Called when the page starts to load
- (void) webView: (WKWebView *) webView didStartProvisionalNavigation: (WKNavigation
*) navigation {
}
// Called when the page fails to load
- (void) webView: (WKWebView *) webView didFailProvisionalNavigation:
(null_unspecified WKNavigation *)navigation withError:(NSError *)error {
[self.progressView setProgress:0.0f animated:NO];
l
// Called when content starts to be returned
- (void) webView: (WKWebView *) webView didCommitNavigation: (WKNavigation
*) navigation {
}
// Called when the page finishes loading
- (void) webView: (WKWebView *) webView didFinishNavigation: (WKNavigation
*)navigation {
[self getCookie];
```

🔗 Tencent Cloud

```
// Called when a submission error occurs
-(void)webView:(WKWebView *)webView didFailNavigation:(WKNavigation
*)navigation withError:(NSError *)error {
[self.progressView setProgress:0.0f animated:NO];
}
// Called after receiving the server's redirect request, i.e., when the service
is redirected
-(void)webView:(WKWebView *)webView
didReceiveServerRedirectForProvisionalNavigation:(WKNavigation *)navigation {
}
```

4. Pass the parameters to OC with JS.

```
 <button id="btn2" type = "button" onclick =
"jsToOcFunction()"> Call OC with parameters with JS </button> 
function jsToOcFunction()
{
window.webkit.messageHandlers.jsToOcWithPrams.postMessage({"params":"res.randst
r"});
}
```

5. Display the rendered WebView in the view, and then call the Captcha service and pass data to the client.

```
-(void)userContentController:(WKUserContentController *)userContentController
didReceiveScriptMessage:(WKScriptMessage *)message{
    // message.body is the json data passed to the client
    // Use message.body to obtain the parameter body passed with JS
    NSDictionary * parameter = message.body;
    // Call OC with JS
    if([message.name isEqualToString:@"jsToOcWithPrams"]){
        // The client obtains the data passed with js and performs operations on the
        data
        parameter[@"params"]
    }
    }
}
```

Note:

After Captcha is integrated into the business client, the business server needs to verify the CAPTCHA ticket (if ticket verification is not integrated, the black market can easily forge verification results, which defeats the purpose of human verification via CAPTCHAs). For more information, please see Integration to Ticket Verification (Web and App).

FAQs

For more information, please see FAQs About Integration.



Android Client SDK Integration

Last updated : 2024-12-27 11:18:42

Service Process Diagram



Preparations

Privacy Settings

Privacy Policy Link: Privacy Policy | Captcha.

Before adapting, integrating, or loading this SDK product into your product, application, or service, you should carefully read and agree to the relevant service agreements, these rules, and/or third-party developer compliance guidelines (or similar legal documents) that we have published, and conduct a compliance self-check on the collection and use of personal information by the product into which you adapt, integrate, or load this SDK product.

If you do not agree to any part of the Privacy Policy, you must immediately stop accessing and using the SDK Product and/or related services. You shall use this SDK Product and process end users' personal information only after obtaining their consent. Before obtaining end users' consent, you should not enable or initialize this SDK.

Captcha CaptchaAppid

When you access this SDK, it relies on the Captcha business ID - CaptchaAppid, which can be obtained through the Tencent Cloud Console verification management. The SDK will not function properly without it. If you do not have the corresponding CaptchaAppid, please create a new one in the Tencent Cloud Console. It will be used later in the required parameter settings for Captcha initialization, i.e., setCaptchaAppid (Captcha CaptchaAppid).

Android SDK Integration

1. Integration Methods

Add in AndroidManifest:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Add in ProGuard:

```
-keep class com.**.TNative$aa { public *; }
-keep class com.**.TNative$aa$bb { public *; }
-keep class com.**.TNative$bb { *; }
-keep class com.**.TNative$bb$I { *; }
```

2. AAR Integration

Copy the AAR file to the libs directory of the project and set the dependency. Add it in the build.gradle file.

```
dependencies {
implementation fileTree(dir: 'libs', include: ['*.aar'])
}
defaultConfig {
ndk {
abiFilters "armeabi-v7a", "arm64-v8a"
}
}
```

Choose armeabi-v7a or arm64-v8a based on actual needs. x86 and x86_64 architectures are not supported yet.

Captcha Integration

Declare to display the CAPTCHA tcaptchaWebview

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

Captcha

Captcha Initialization and Call Example

```
// Initialize
TencentCaptchaConfig.BuilderconfigBuilder =new
TencentCaptchaConfig.Builder(getApplicationContext(),
newITencentCaptchaPrivacyPolicy(){...},
newICaptchaDeviceInfoProvider(){...});
RetCoderet =TencentCaptcha.init(config.build());
...
// Required parameters
TencentCaptchaParam.BuildertencentCaptchaBuilder =new
TencentCaptchaParam.Builder
()
.setWebView(tcaptchaWebView)
.setCaptchaAppid(CaptchaAppid);
// Business start
TencentCaptcha.start(callback,tencentCaptchaBuilder.build());
```

Captcha Callback Response Example

```
TencentCaptchaCallback callback = new TencentCaptchaCallback() {
@Override
public void finish(RetCode ret, JSONObject resultObject) {...}
@Override
public void exception(Throwable t) {...}
};
```

SDK Class Description

Main Class Description

Class	Description
TencentCaptcha	Main entry class of SDK
TencentCaptchaConfig	SDK initialization configuration
TencentCaptchaCallback	SDK result callback
TencentCaptchaParam	SDK Captcha input parameters

TencentCaptcha Class (Main Entry) API Description

/***

```
Initialize with the built configuration
* @param TencentCaptchaConfig
* @return RetCode
*/
RetCode init(TencentCaptchaConfig);
/**
* Start Captcha detection
* @param TencentCaptchaCallback
* @param TencentCaptchaParam
* @return RetCode
*/
RetCode start(TencentCaptchaCallback, TencentCaptchaParam);
/**
* Interrupt detection process
*/
void stop();
```

TencentCaptchaConfig Class (Initialization Configuration) API Description

Create using Builder, supports chained calls.

```
/**
 * Create a TencentCaptcha.Builder object.
 *
 * @param Context
 * @param ITencentCaptchaPrivacyPolicy API instance, [must pass implementation],
 any API call of the SDK will use this instance to ask the user if they have
 agreed to the privacy policy
 * @param ICaptchaDeviceInfoProvider API instance, pass ANDROIDID, used for the
 validity check of the device identifier itself and to identify abnormal device
 environments to detect device risks*/
 */
Builder(Context, ITencentCaptchaPrivacyPolicy,
 ICaptchaDeviceInfoProvider);
```

TencentCaptchaCallback Class (Result Callback) API Description

```
/**
 * Callback when Captcha verification ends, notifying the host to obtain
 verification ticket and other information.
 *
 * @param RetCode, when RetCode is not OK, please refer to [retcode values]
 section
```

```
* @param JSONObject, when RetCode is OK, please refer to [JSONObject return
values] section for information
*/
void finish(RetCode, JSONObject);
/**
* Callback when an exception occurs in the SDK working thread
*
* @param Throwable
*/
void exception(Throwable);
```

TencentCaptchaParam Class (Input Parameters) API Description Parameters class for starting Captcha.

```
/**
* Create a TencentCaptchaParam.Build object.
* @return TencentCaptchaParam
*/
TencentCaptchaParam build();
/**
* Set the CaptchaAppid, obtained from the tencent cloud console. Without this,
the SDK will not work properly.
* @param String, CaptchaAppid.
* @return Build
*/
Builder setCaptchaAppid(String);
/**
* Set the WebView to display the Captcha.
* @param webView
* @return Builder
*/
Builder setWebView(WebView);
```

Note:

For more optional parameters, see TencentCaptchaParam class optional parameters configuration section.

Sample Code

Creating the SDK Configuration Builder

```
TencentCaptchaConfig.Builder configBuilder = new
TencentCaptchaConfig.Builder(
```

```
getApplicationContext(),
    new ITencentCaptchaPrivacyPolicy() {
        @Override
        public boolean userAgreement() {
            // Authorization Passed
            // true indicates authorization passed, the SDK can be used
normally
           // false indicates unauthorized, subsequent calls to the start
method will throw an exception
return false;
}
},
new ICaptchaDeviceInfoProvider() {
@Override
public String getAndroidId() {
return androidId; // Implementation to get AndroidId
}
}
);
```

Creating the SDK Configuration Builder

```
// Initialize the SDK
RetCode ret = TencentCaptcha.init(configBuilder.build());
if(ret == RetCode.OK) {
    // Initialized successfully
} else {
    // Initialization failed
    log.error("ret code: " + ret.getCode() + " msg:" + ret.getMsg()); }
```

Implementing Captcha Callback

```
TencentCaptchaCallback callback = new TencentCaptchaCallback() {
@Override
public void finish(RetCode ret, JSONObject resultObject) {
if (ret == RetCode.OK) {
// Success, execute business logic
} else {
// Failed
log.error("code:" + ret.getCode() + " msg:" + ret.getMsg());
}
@Override
public void exception(Throwable t) {
// Abnormal
```

```
t.printStackTrace();
};
```

Creating Captcha Input Parameters

```
TencentCaptchaParam.Builder paramBuilder = new
TencentCaptchaParam.Builder()
.setWebView(tcaptchaWebView) // required, WebView to display Captcha
.setCaptchaAppid("CaptchaAppid"); // required, CaptchaAppid
```

Starting Captcha

```
TencentCaptcha.start(callback, paramBuilder.build());
```

Interrupting Captcha (Typically Called When Exiting the Interface)

```
@Override
protected void onPause() {
  super.onPause();
  TencentCaptcha.stop();}
```

RetCode (Callback Return Value When Captcha Verification Ends)

Retcode Method

```
/**
* Get Error Code
*
* @return int
*/
int getCode();
/**
* Get Detailed Error Information
*
* @return String
*/
String getMsg();
```

Retcode Values

Retcode Values	code	Description
OK	0	Succeeded
INIT_FAILED	1	Initialization failed
NOT_INIT	2	Uninitialized
INVALID_PARAMS	3	Invalid Input Parameters
INVALID_WEBVIEW	4	Webview Is Empty
CAPTCHA_RETDATA_EMPTY	5	Captcha return parameter is empty
CAPTCHA_EXCEPTION	6	Captcha detection exception
POLICY_ERROR	7	Privacy terms not agreed

JSONObject (Callback Return Business Parameters When Captcha Verification Ends)

Field Name	Value Type	Description
ret	Int	Verification result, 0: Verification successful.
ticket	String	Ticket returned by Captcha, ticket has value only when ret = 0.
CaptchaAppId	String	Captcha Application ID.
bizState	Any	Custom pass-through parameters.
randstr	String	Random string for this verification, this parameter needs to be passed during subsequent ticket verification.
errorCode	Number	Error code, for details, see the callback function errorCode description.
errorMessage	String	Error message.

TencentCaptchaParam Class (Optional Parameter Configuration Description)

```
/**
* Custom pass-through parameters, this field can be used by the business to
pass a small amount of data, and the content of this field will be included in
the callback object.
* @param Object
* @return Builder
*/
Builder setBizState(Object);
/**
* true: Enable adaptive night mode.
* @param boolean
* @return Builder
*/
Builder setEnableDarkMode(boolean);
/**
* "force": Enable forced night mode.
* @param String
* @return Builder
*/
Builder setEnableDarkMode(String);
/**
* false: Hide the help button.
* @param boolean
* @return Builder
*/
Builder setNeedFeedBack(boolean);
/**
* Custom help link: "URL".
* @param String
* @return Builder
*/
Builder setNeedFeedBack(String);
/**
* Specify the language of the Captcha prompt text, which takes precedence over
console configuration.
* Supports passing the same value as navigator.language, the user's preferred
language, case-insensitive.
```

🔗 Tencent Cloud

```
* For details, see
https://intl.cloud.tencent.com/zh/document/product/1159/49680?
lang=zh&pg=#userLanguage.
* @param String
* @return Builder
*/
Builder setUserLanguage(String);
/**
* true: enable aging adaptation
* false: disable aging adaptation
* @param boolean
* @return Builder
*/
Builder setEnableAged(boolean);
/**
* Define the Captcha display method.
* popup: (default) Popup, displays the Captcha in a centered floating layer.
* full: Full Screen
* @param String
* @return Builder
*/
Builder setType(String);
/**
* Whether to show loading box during Captcha loading. If this parameter is not
specified, the loading box is shown by default.
* true: show loading box
* false: do not show loading box
* @param boolean
* @return Builder
*/
Builder setLoading(boolean);
/**
* Set CaptchaAppId encrypted verification string, see encryption verification
capability access guide for details
* Oparam String
* @return Builder
*/
Builder setAidEncrypted(String);
/**
* Callback for some internal Captcha information, see
[TencentCaptchaParam.OptionsCallback description] for details.
```

🕗 Tencent Cloud

```
* @param OptionsCallback
* @return Builder
*/
Builder setOptionsCallback(OptionsCallback);
```

TencentCaptchaParam.OptionsCallback Description

```
/**
* 1. When optFuncName is ready,
   data returns the callback when the Captcha is loaded, with the actual width
*
and height of the Captcha (unit: px):
   {"sdkView": {
*
*
        "width": number,
        "height": number
*
*
    } }
*
   This parameter is only for viewing the Captcha dimensions, do not use this
parameter to set the dimensions directly.
* 2. When optFuncName is showFn, data returns duration, rendering time + sid
information.
* @param optFuncName, String
* @param data, String
*/
void onCallback(String optFuncName, String data); result callback, return value
```



Integration of IOS Client SDK

Last updated : 2024-12-27 11:18:41

A Service Flowchart



Privacy Settings

Privacy Policy Link: Privacy Policy | Captcha.

Before adapting, integrating, or loading this SDK product into your product, application, or service, you should carefully read and agree to the related service agreement, these rules, and/or the third-party developer compliance guide (or similar legal documents) that we have published, and conduct a compliance self-check on the collection and use of personal information by the product that adapts, integrates, or loads this SDK product. If you do not agree with any part of the privacy agreement, you should immediately stop accessing and using the SDK and/or related services. You should use this SDK product and process end users' personal information only after obtaining their consent. Before obtaining end users' consent, you should not enable or initialize this SDK.

iOS SDK Integration

Integrate the SDK into your project.

Drag the TencentCaptcha directory from the SDK package into your project.

In Xcode, click your project and target, then open the "Build Settings" tab.

Locate "Library Search Paths", expand it, and add the Captcha directory if it is not automatically added. Locate "Header Search Paths", expand it, and add the Captcha directory if it is not automatically added. Ensure TencentCaptcha.a is linked with your project (Link Binary with Libraries).

Configuration Parameters

Set up WebView

The instance of WKWebView needs to be initialized by you. Due to some system limitations, the SDK cannot currently receive WKWebView generated from StoryBoard.

```
override func viewDidLoad() {
  super.viewDidLoad()
  // Do any additional setup after loading the view.
  let contentController = WKUserContentController()
  let config = WKWebViewConfiguration()
    config.userContentController = contentController
    webView = WKWebView(frame: self.view.bounds, configuration: config)
    webView.navigationDelegate = self
    webView.autoresizingMask = [.flexibleWidth, .flexibleHeight]
   self.view.addSubview(webView)
}
```

Obtain Captcha characters

```
TencentCaptcha.shared().requestCaptcha(withWebView: self.webView,
options:options) { error, result in
if let error = error {
    // If an error occurs, error is non-null
  }
else {
    // result is the return value
  }
}
```

Class Declaration

TencentCaptchaOptions

Captcha Request Parameters



See `-requestCaptchaWithWebView:options:withCompletionHandler:`TencentCaptcha

/// Captcha request parameters @interface TencentCaptchaOptions : NSObject /// Custom pass-through parameters. This field can be used by the business to pass a small amount of data, and its content will be included in the callback object. @property (nonatomic, strong, nullable) NSString *bizState; /// Night mode @property (nonatomic, assign) TencentCaptchaOptionsDarkMode darkModeEnabled; /// Help button switch @property (nonatomic, assign) BOOL needsFeedback; /// Custom help button link /// @discussion If `customFeedbackURL` is not empty, the `needsFeedback` option is invalid, and the help button is always displayed, redirecting to the URL when clicked @property (nonatomic, strong, nullable) NSString *customFeedbackURL; /// Specify the language for the Captcha prompt text, which takes precedence over console configuration @property (nonatomic, strong, nullable) NSString *userLanguage; /// Elderly-friendly mode @property (nonatomic, assign) BOOL agedEnabled; /// Define Captcha display method /// @discussion popup (default) - Pop-up style, displays the Captcha in a centered floating layer. embed - Embedded style, displays the Captcha embedded in the specified container element. @property (nonatomic, strong, nullable) NSString *type; /// Whether to show a loading box during Captcha loading @property (nonatomic, assign) BOOL loadingEnabled; /// CaptchaAppId encrypted validation string /// @discussion For details, see https://intl.cloud.tencent.com/zh/document/product/1159/49680? lang=zh&pg=#captchaAppid @property (nonatomic, assign) BOOL AppIDEncrypted; /// parameter and intermediate process callback /// @discussion Set this callback when you need to receive intermediate parameters @property (nonatomic, copy, nullable) void(^optionsCallback)(NSString *_Nonnull functionName, NSDictionary<NSString *, id> *_Nonnull data); 0end

TencentCaptchaOptionsDarkMode

Night mode options

```
/// Night mode options
typedef enum : NSUInteger {
   /// Default
```

TencentCaptchaOptionsDarkModeDefault,

/// Enabled with system

TencentCaptchaOptionsDarkModeTrue,

/// Always disabled

TencentCaptchaOptionsDarkModeFalse,

- /// Force enabled
 TencentCaptchaOptionsDarkModeForce,
- } TencentCaptchaOptionsDarkMode;

TencentCaptchaConfiguration

Captcha Service Configuration

See: `+setupWithConfigurationHandler:`TencentCaptcha

```
/// Captcha service configuration
/// @discussion
/// Used to configure the service in `+[TencentCaptcha TencentCaptcha:]`
@interface TencentCaptchaConfiguration : NSObject
- (nonnull instancetype) init NS_UNAVAILABLE;
@property (nonnull, nonatomic, copy) NSString *appID;
/// Tars server address
/// @discussion If you are unsure of its purpose, do not set it
@property (nullable, nonatomic, copy) NSURL *tarsServerURL;
/// The URL of the Captcha HTML page
/// @discussion If you are unsure of its purpose, do not set it
@property (nullable, nonatomic, copy) NSURL *webPageURL;
/// Whether to allow using cached tokens to speed up the request process
/// @discussion The default is YES, and the caching strategy can meet most
scenarios. If you are particularly concerned about real-time risks, set it to
NO
@property (nonatomic, assign) BOOL usesCacheToken;
@end
```

TencentCaptcha

```
/// The implementation class of the Captcha service
@interface TencentCaptcha : NSObject
- (nonnull instancetype)init NS_UNAVAILABLE;
/// The singleton of the Captcha service
+ (nonnull instancetype)sharedService NS_SWIFT_NAME(shared());
/// Configuration service options
/// - Parameter configurationHandler: To modify the configuration, change the
properties in the configuration provided in this callback function
- (void)setupWithConfigurationHandler:(nonnull void(^)
(TencentCaptchaConfiguration *_Nonnull))configurationHandler
```

```
NS_SWIFT_NAME(setup(configurationHandler:));
/// Request Captcha
/// - Parameters:
/// - webView: The web view used to display the Captcha
111
      - options: Captcha request parameters
     - completionHandler: Completion callback for verification
111
///
/// @discussion Due to some system limitations, the SDK cannot currently
receive WKWebView generated from StoryBoard.
- (void) requestCaptchaWithWebView: (nonnull WKWebView *) webView options:
(nullable TencentCaptchaOptions *) options withCompletionHandler: (nonnull
void(^)(NSError *_Nullable error, NSDictionary *_Nullable
result))completionHandler
NS_SWIFT_NAME (requestCaptcha (withWebView:options:withCompletionHandler:));
@end
```

Field Description

Field description of error in the completion callback (completionHandler).

When an error occurs, error is non-null and result is null. See Common mistakes.

Field description of result in the completion callback (completionHandler).

When the Captcha completes normally, error is null and result is non-null. Note that user verification failure is considered a normal completion of the Captcha.

Field Name	Value Type	Description		
ret	Int	Verification result, 0: Verification successful.		
ticket	String	Ticket returned by the Captcha, ticket has a value if and only if $ret = 0$.		
CaptchaAppId	String	Captcha Application ID.		
bizState	Any	Custom pass-through parameters.		
randstr	String	The random string for this verification, which needs to be passed during subsequent ticket verification.		
errorCode	Number	Error code. For details, see Callback function errorCode description.		
errorMessage	String	Error message.		

Common mistakes

Device Safety Layer Error (Error Domain TSLocalErrorDomain)

Error Code	Description
0	No error
1	Request protocol packaging error
2	Reply protocol unpacking error
3	Empty reply error
31	X1 protocol data compression error
4	Shark protocol shark layer unpacking error
5	Shark protocol shark layer does not contain sashimi
6	Shark protocol sashimi layer unpacking error
7	Shark protocol sashimi layer empty packet error
8	Shark protocol registration GUID error
9	Shark protocol save GUID error
10	Shark protocol unrecognized command word
20	Session expiration
21	Abnormal return of Shark Skin layer
22	Shark Skin layer unpacking error
23	Shark protocol sashimi layer return value is non-zero
11	The record is empty
12	The record content is invalid or some data is missing (error details available in DEBUG mode)
13	GUID not included when sending Shark request (currently only theoretically possible)
14	Record not included when sending Shark/WUP request
15	Current GUID does not correctly correspond to server type
16	Neither error nor server response included when generating reply content (currently only theoretically possible)
17	Client error for HTTP request (status code between 400~499)

18	Server error for HTTP request (status code between 500~599)
19	Received response does not correspond to the request
25	Temporarily unable to request
26	Illegal internal call
27	Request timeout
28	Illegal communication protocol, or incorrect communication protocol selected
29	Request queue too long
30	User/business side cancels the request
24	Internal testing reserved

SDK Download

Last updated : 2024-12-27 11:20:01

	Ċ
Download for Android	Download for iOS
Integration Guide	Integration Guide

Server Integration Integration to Ticket Verification (Web and App)

Last updated : 2024-03-01 11:20:59

The server needs to call the ticket verification API to verify the client verification result.

Note:

If ticket verification is not integrated, the black market can easily forge verification results, which defeats the purpose of human verification via CAPTCHAs.

Integration steps

Step 1. Install the SDK in the corresponding language.

Tencent Cloud provides SDKs in multiple languages. You can install the SDK in any of the languages based on your business needs. For more information, please see SDK Center.

Step 2. Obtain the TencentCloud API key.

1. Log in to the CAM console and select Access Key -> API Key Management on the left navigation pane.

2. On the API key management page, if the key has been created, you can view it on this page; if the key has not been created, click **Create Key** to generate the required key.

Create Key					
APPID	Кеу	Creation Time	Last Access Time	Status	Operation
251228349	SecretId: 🔽 🔽 SecretKey: *****Show	2022-08-19 14:28:35		On	Disable

Step 3. Call the DescribeCaptchaResult API.

It is recommended to use API Explorer to generate code online. For more information about the API, please see Verify CAPTCHA Ticket Result (Web and App).

FAQs

For more information, please see Integration FAQs.