埋点对接

1. web端埋点

可以在运营管理——页面埋点中绑定，配合url拼接方式可以回传未登录用户的点击事件

谷歌平台：

gtag

请将辨别id（例：G-PFSS5SP9ND）加在推广地址上

示例：https://web2023web.cg.vin/home/game?id=25660923&gtagId=xxxxxxxxx

gtagId 这个代表著 gtag 用此id载入并且初始化，接下来操作的上报都会反馈到这个id所属的后台

gtm：请提供辨别id、例：GTM-N3ZMGKH

请将辨别id（例：GTM-N3ZMGKH）加在推广地址上

示例：https://web2023web.cg.vin/home/game?id=25660923&gtmId=xxxxxxxxx

gtmId 这个代表著 GTM 用此id载入并且初始化，接下来操作的上报都会反馈到这个id所属的后台

firebase：请提供辨别埋点代码内涵埋点参数、例：{  
        apiKey: 'AIzaSyDDHonDGjU0zyFHV89178\_1ggDvxxxxxxx',  
        authDomain: 'ssceshi-xxxx.firebaseapp.com',  
        projectId: 'ssceshi-xxxx',  
        storageBucket: 'ssceshi-xxxx.appspot.com',  
        messagingSenderId: '74244900000',  
        appId: '1:742449233486:web:8a1ac1d8c79xxxxxxx',  
        measurementId: 'G-QZVVxxxxxx'  
      }

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Facebook/Meta：

meta pixel：请将辨别id（例：1253619738841000）加在推广地址上

示例：https://web2023web.cg.vin/home/game?id=25660923&fb\_dynamic\_pixel=xxxxxxxxx

fb\_dynamic\_pixel这个代表著meta pixel用此id载入并且初始化，接下来操作的上报都会反馈到这个id所属的后台

标准事件：

register:  'CompleteRegistration', 完成注册

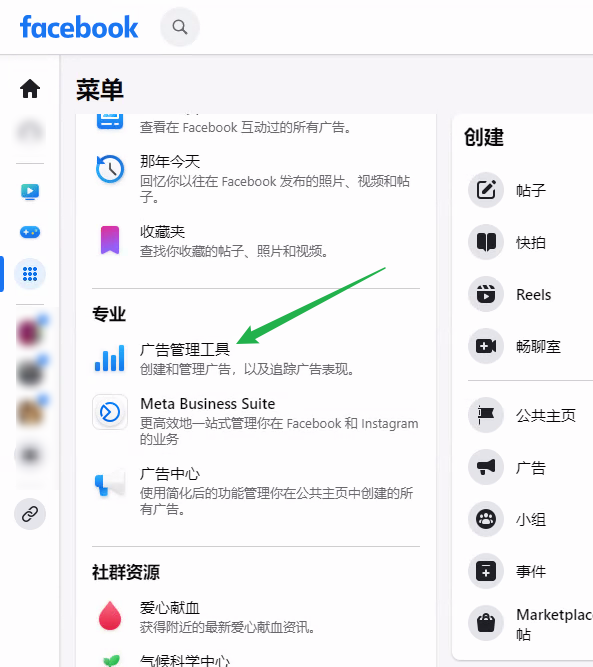
 fbq('track', 'InitiateCheckout'  发起结账 ——送出存款订单  
 firstrecharge: 'Purchase',购物   ——首充  
 recharge: 'Purchase'购物   ——复充

Facebook操作步骤

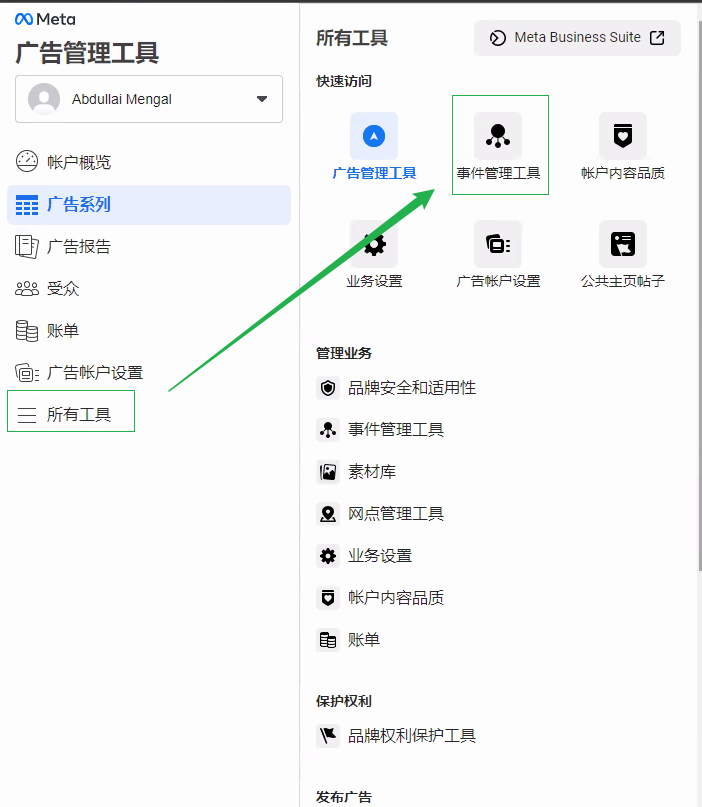
1.进入Facebook  pixel/meta后台

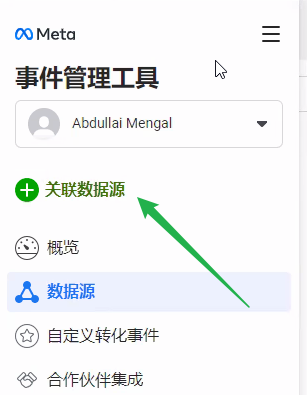


2、pixel/meta后台菜单页面点击专业类别下的广告管理工具，进入广告管理工具

[](https://www.cg.xxx/file-read-417509.png)

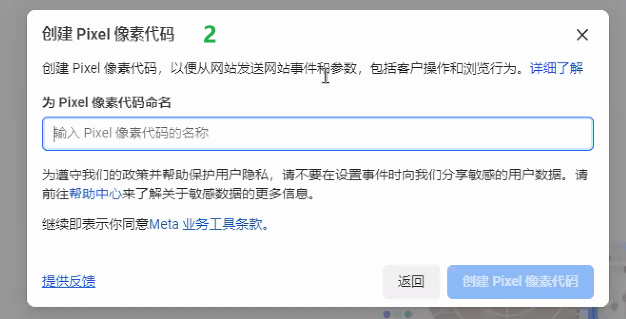
3、进入广告管理工具后，点击菜单下的所有工具，选择事件管理工具，事件管理器

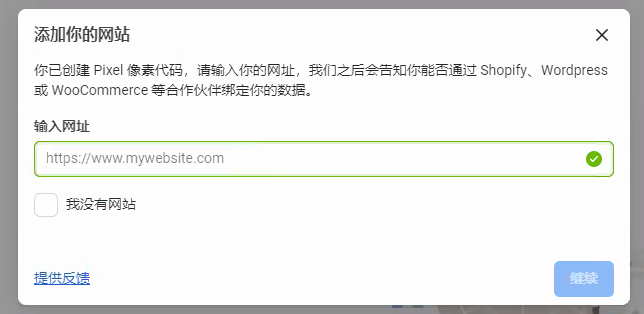
[](https://www.cg.xxx/file-read-417510.png)[展开全部](https://www.cg.xxx/doc-objectLibs-custom-0-274-24669.html###)  
4、进入事件管理器，点击连接数据源

**[](https://www.cg.xxx/file-read-417511.png)**

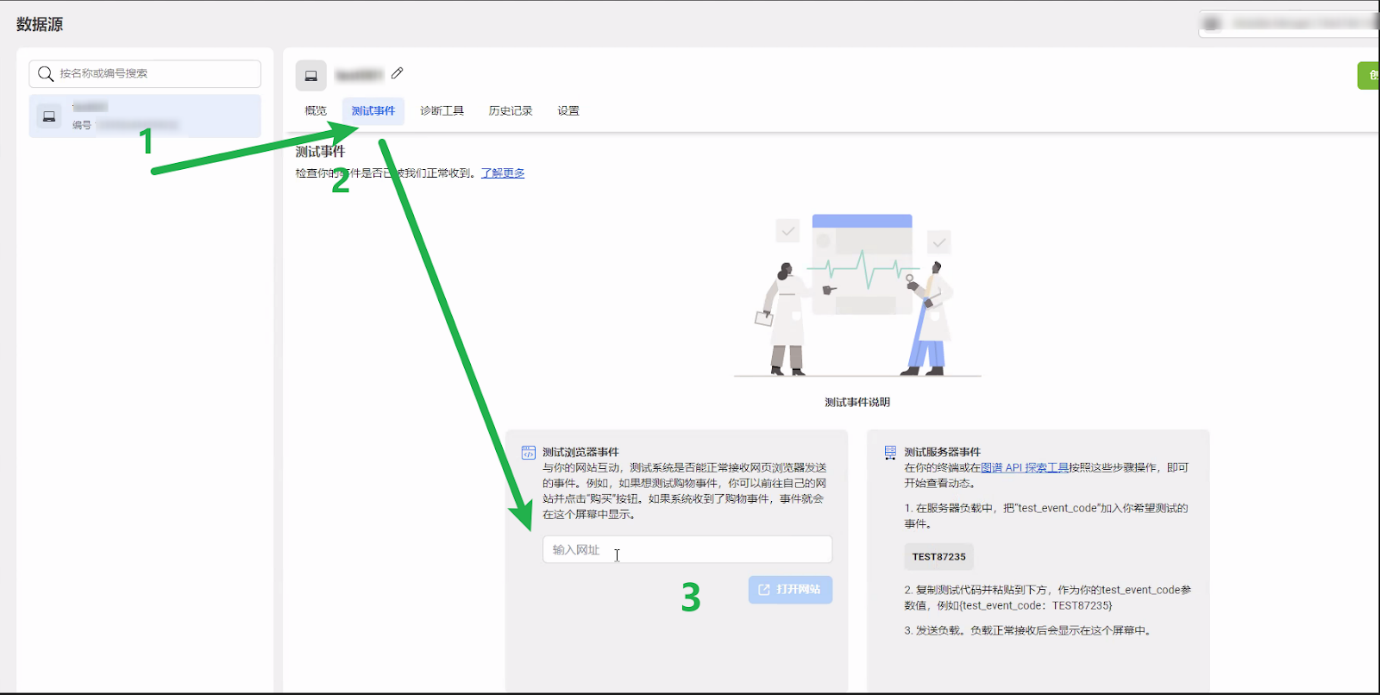
5、根据下图步骤选择网页——输入像素名称——关闭

**[](https://www.cg.xxx/file-read-417520.png)**

**[](https://www.cg.xxx/file-read-417521.png)**

**[](https://www.cg.xxx/file-read-417522.png)**

6、选择刚刚新建的数据源，像素id展示在数据源名称下方，点击事件测试，输入拼接好的链接（带有像素id的地址），点击打开网站按钮/浏览器中直接打开拼接好的链接触发事件

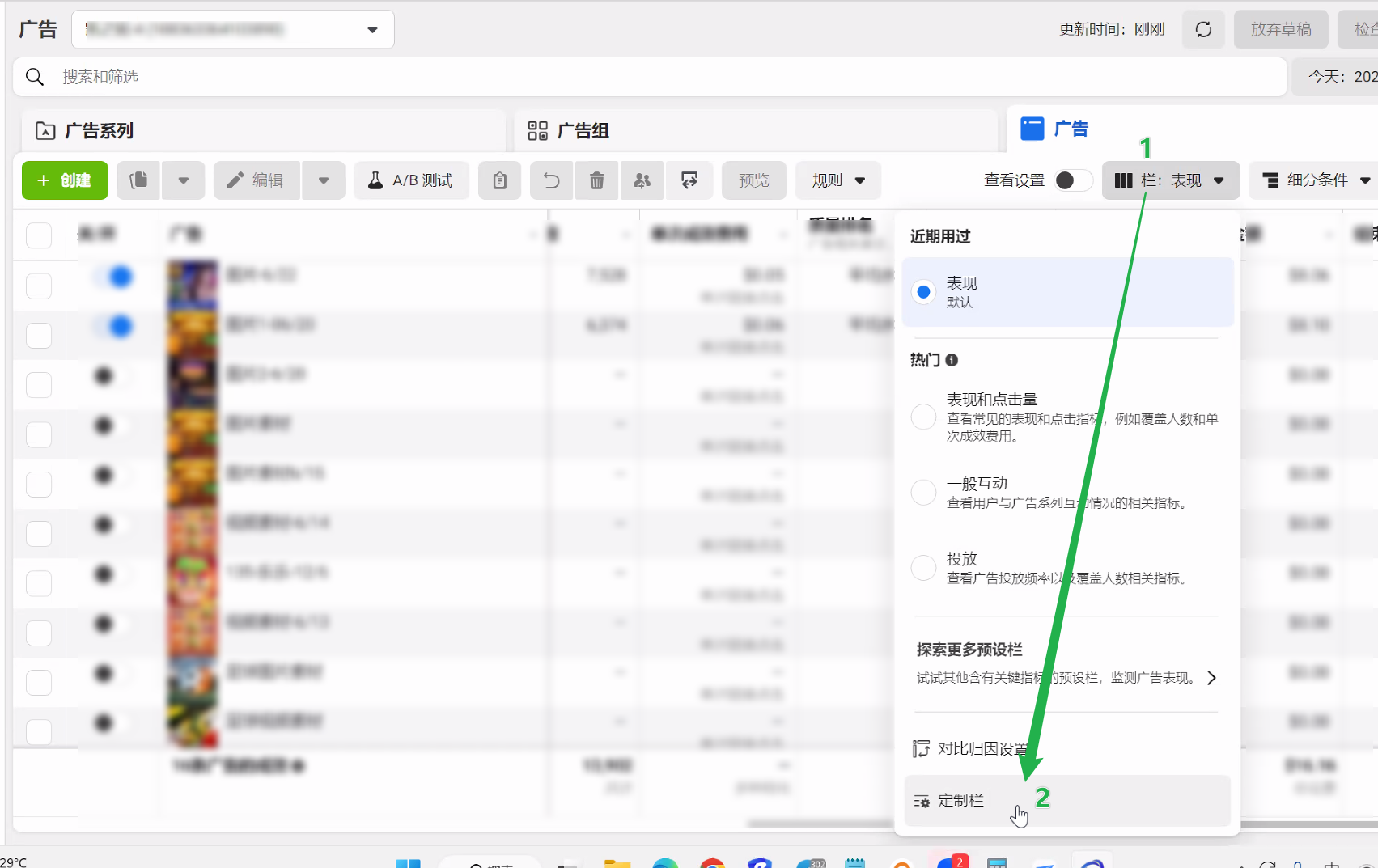


6、触发的事件返回会展示在页面中，如下图

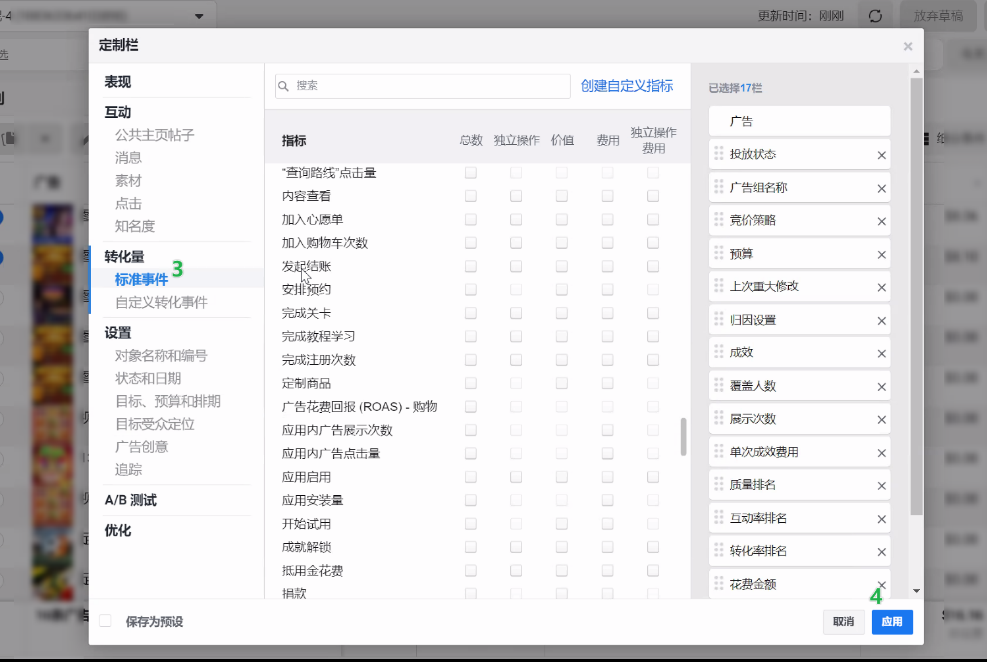


标准事件配置

1、进入定制栏

[](https://www.cg.xxx/file-read-399126.png)

2、勾选标准事件，应用就可以查看了

[](https://www.cg.xxx/file-read-399134.png)

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

快手埋点 国际版：

kwai pixel：请将辨别id（例：1253619738841000）加在推广地址上

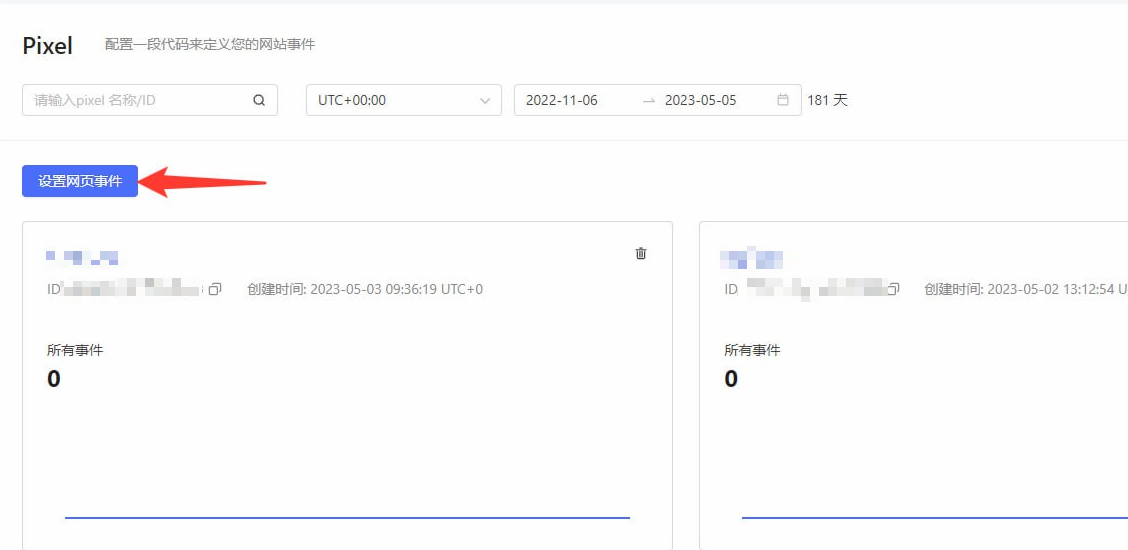
示例：https://web2023web.cg.vin/home/game?id=25660923&kwaiPixelBaseCode=xxxxxxxxx

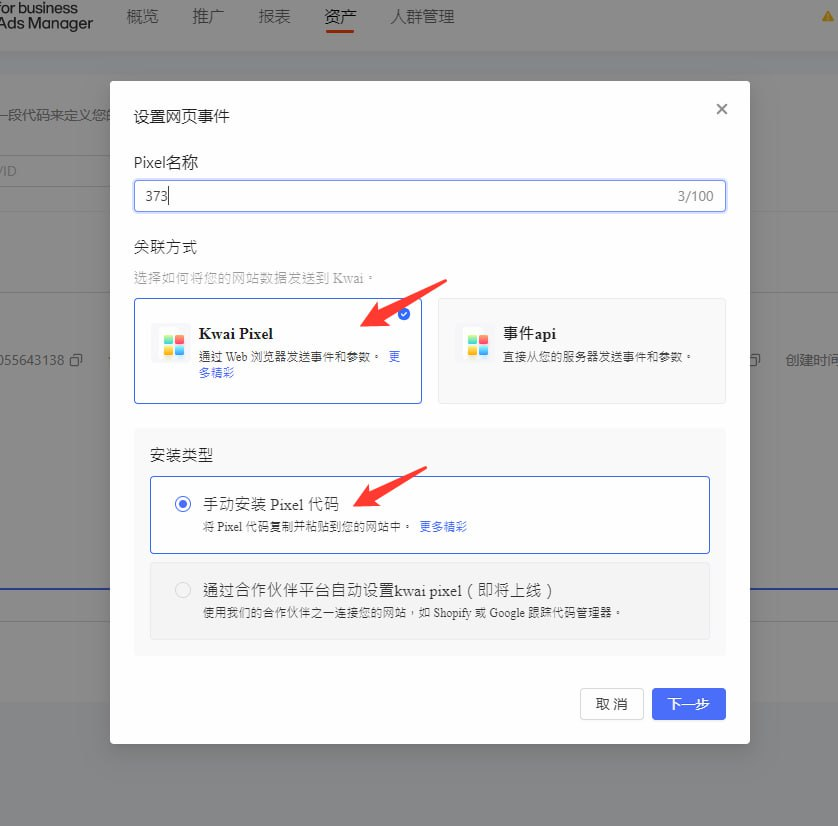
kwaiPixelBaseCode这个代表著快手埋点用此id载入并且初始化，接下来操作的上报都会反馈到这个id所属的后台

快手不支持自定事件，我们只能用原本事件去映射

映射事件：  
register: 'completeRegistration',完成注册  
rechargeClick: 'addToCart',加购（获取到三方地址就是了）。  
firstrecharge: 'purchase',首充  
recharge: 'addToCart' ，复充

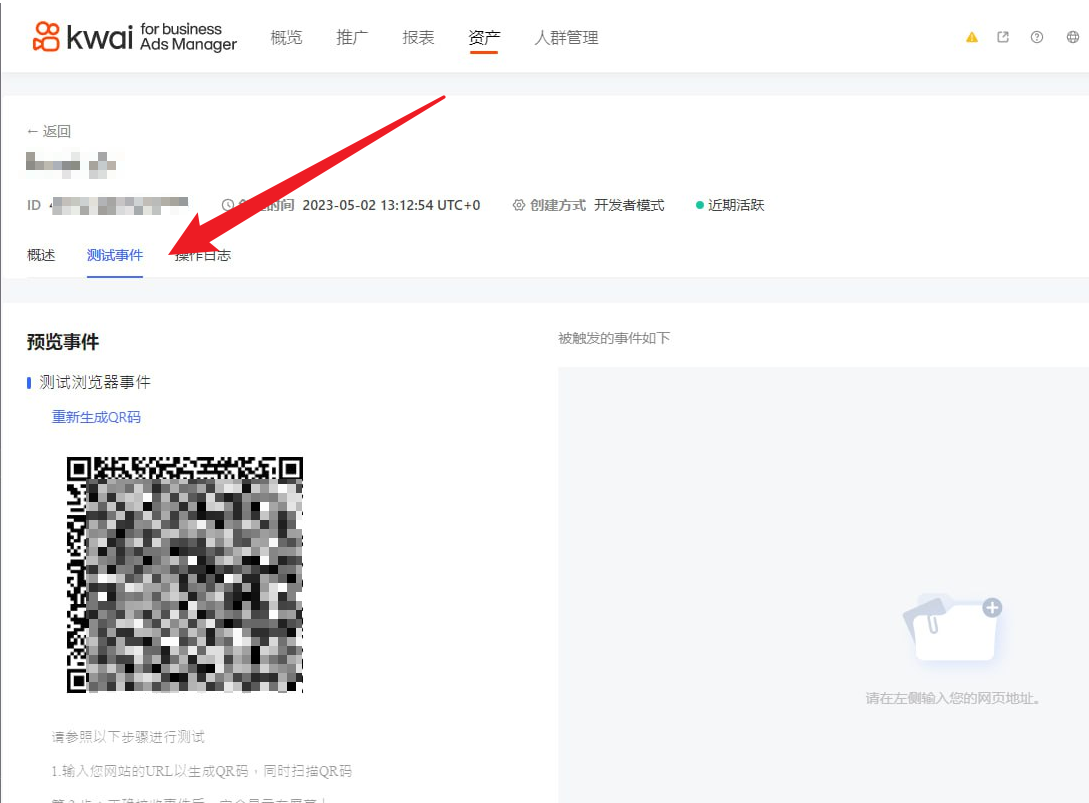
其他对不上的事件都当成点击事件  
自检器是可以监测到事件的，关于后台数据，需要从广告来的他才会执行后面那段的上报到平台，可以先自检器看事件正确性 但是后台是不会收到的  
kwai后台的增加方式  
1、点击设置网页事件

[](https://www.cg.xxx/file-read-371553.png)

2、选择pixel，手动安装Pixel代码  
[](https://www.cg.xxx/file-read-371549.png)  
3、选择模式：开发者模式（因为标准模式下是做不了任何操作，我们埋入的代码是不会生效）

[](https://www.cg.xxx/file-read-371550.png)

4、把加了id的推广地址放到测试事件里面进行测试就可以了

[](https://www.cg.xxx/file-read-371554.png)

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

抖音埋点 国际版：

tiktok：请将辨别id（例：1253619738841000）加在推广地址上

示例：https://web2023web.cg.vin/home/game?id=25660923&tiktokBaesCode=xxxxxxxxx

tiktokBaesCode这个代表著抖音埋点用此id载入并且初始化，接下来操作的上报都会反馈到这个id所属的后台

提示：tiktok支持自定义事件，具体请参考titok官方文档 ： https://ads.tiktok.com/help/article/tiktok-app-id?lang=zh

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

2.马甲包推送

h5与马甲包的沟通，请马甲包在webview下的window插入对象jsBridge，内容提供回调函数postMessage

h5回调代码如下

window.jsBridge?.postMessage(eventName, params)

传递参数

eventName ：字串;事件名称(下方备注)

params：JSON字串，基础内容提供，代理线 id、渠道 cid、域名 domain，几个特殊事件会有其他的内容

例：{ id: 123, cid:123, domain:'https://web2023web.cg.vin/'}

马甲包安卓代码示例

1. android 原生接入步骤 

**webView**.addJavascriptInterface(**new** JsInterface() , **"jsBridge"**);

**webView**.getSettings().setJavaScriptEnabled(**true**);

**webView**.getSettings().setJavaScriptCanOpenWindowsAutomatically(**true**);

webView.getSettings().setSupportMultipleWindows(**true**);

2. postMessage方法  回调web的数据

public class JsInterface{

// Android 调用 Js 方法1 中的返回值

@JavascriptInterface

public void postMessage(String name,String data){

Log.e("TAG", "postMessage name=="+name);

Log.e("TAG", "postMessage data=="+data);

}

}

--------------------------------------------------------------------------------------------------

1. IOS 原生接入步骤（完整webview代码示例，包括埋点，开窗，下分） 

import UIKit

import WebKit

import AppsFlyerLib

typealias CloseBlock = () -> Void

class WebViewController: UIViewController,WKNavigationDelegate, WKUIDelegate, WKScriptMessageHandler {

///关闭webviewVC回调

var closeBlock: CloseBlock?

private var webView: WKWebView?

//点击充值标记

var recharge = ""

@objc init(url: String) {

super.init(nibName: nil, bundle: nil)

navigationController?.navigationBar.barTintColor = .systemBlue

navigationController?.navigationBar.tintColor = .white

let closeButton = UIBarButtonItem(image: UIImage(systemName: "xmark"), style: .plain, target: self, action: #selector(closeButtonClicked))

navigationItem.rightBarButtonItem = closeButton

//注入JS事件代码

let config = WKWebViewConfiguration()

let userContent = WKUserContentController()

let jsCode = "window.jsBridge = {\n postMessage: function(name, data) {\n window.webkit.messageHandlers.Post.postMessage({name, data})\n }\n};\n"

let zuowan = WKUserScript(source: jsCode, injectionTime: WKUserScriptInjectionTime.atDocumentStart, forMainFrameOnly: false)

userContent.addUserScript(zuowan)

userContent.add(self, name: "Post")

config.userContentController = userContent

webView = WKWebView(frame: view.frame, configuration: config)

webView?.navigationDelegate = self

webView?.uiDelegate = self

view.addSubview(webView!)

webView!.load(URLRequest(url: URL(string: url)!))

}

@objc private func closeButtonClicked() {

closeBlock?()

dismiss(animated: true, completion: nil)

}

func userContentController(\_ userContentController: WKUserContentController, didReceive message: WKScriptMessage) {

//监听JS消息

if message.name == "Post" {

if let body = message.body as? [String: Any] {

//消息名称

let name = body["name"] as? String ?? ""

//消息数据

let data = body["data"] as? String ?? ""

var dic = [String: Any]()

if let subData = data.data(using: .utf8) {

do {

if let json = try JSONSerialization.jsonObject(with: subData, options: []) as? [String: Any] {

dic = json

if (name != "openWindow") {

//记录点击充值

recharge = name

//埋点事件上报AF

sendAppsFlyer(name, dic)

return

}

// 如果为充值则不跳转新窗口，而是下面 UIApplication.shared.open(url) 跳到浏览器

if (recharge == "rechargeClick") {

recharge = ""

return

}

// openWindow 跳转新窗口

let url = dic["url"] as? String ?? "";

if (!url.isEmpty) {

openWindow(url)

}

}

} catch {

sendAppsFlyer(name, [name:subData])

}

}

}

}

}

//埋点事件上报AF

func sendAppsFlyer(\_ name: String, \_ message: Dictionary<String, Any>) {

if (name == "firstrecharge" || name == "recharge" || name == "withdrawOrderSuccess") {

//上报首充，充值，提现

let ci = message["amount"]

let jian = message["currency"]

if ci != nil && jian != nil {

if let tai = Double(ci as! String) {

AppsFlyerLib.shared().logEvent(name: name, values: [AFEventParamRevenue: name == "withdrawOrderSuccess" ? -tai: tai,AFEventParamCurrency:jian!])

}

}

}else {

//上报其他事件

AppsFlyerLib.shared().logEvent(name, withValues: message)

}

}

//马甲包内部打开新窗口

func openWindow(\_ url: String) {

let gameWebVC = WebViewController(url: url)

//关闭三方游戏时，发送JS消息 进行下分刷新

gameWebVC.closeBlock = {

let jsMessage = "window.closeGame();"

self.webView?.evaluateJavaScript(jsMessage, completionHandler: nil)

}

let gameController = UINavigationController(rootViewController: gameWebVC)

gameController.modalPresentationStyle = .fullScreen

self.present(gameController, animated: true)

}

func webView(\_ webView: WKWebView, createWebViewWith configuration: WKWebViewConfiguration, for navigationAction: WKNavigationAction, windowFeatures: WKWindowFeatures) -> WKWebView? {

if navigationAction.targetFrame == nil {

if let url = navigationAction.request.url {

//跳转浏览器

UIApplication.shared.open(url)

}

}

return nil

}

func webView(\_ webView: WKWebView, didReceive challenge: URLAuthenticationChallenge, completionHandler: @escaping (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {

let authentisaMethod = challenge.protectionSpace.authenticationMethod

if authentisaMethod == NSURLAuthenticationMethodServerTrust {

var credential: URLCredential? = nil

if let serverTrust = challenge.protectionSpace.serverTrust {

credential = URLCredential(trust: serverTrust)

}

completionHandler(URLSession.AuthChallengeDisposition.useCredential, credential)

}

}

required init?(coder aDecoder: NSCoder) {

super.init(coder: aDecoder)

}

func animateView(\_ view: UIView, duration: TimeInterval, delay: TimeInterval, options: UIView.AnimationOptions, animations: @escaping () -> Void, completion: ((Bool) -> Void)? = nil) {

UIView.animate(withDuration: duration, delay: delay, options: options, animations: animations, completion: completion)

}

}

**iOS添加包信息获取支持**

**在页面加载后执行**

let appVersion = Bundle.main.infoDictionary?["CFBundleShortVersionString"] as! String

let BallStr1 = "window.WgPackage = {name: '\(Bundle.main.bundleIdentifier!)', version: '\(appVersion)'}"

let ScrUser1 = WKUserScript(source: BallStr1, injectionTime: WKUserScriptInjectionTime.atDocumentStart, forMainFrameOnly: false)

userBallController.addUserScript(ScrUser1)

userBallController.add(self, name: "Ball")

ConfBall.userContentController = userBallController

BallView = WKWebView(frame: view.frame, configuration: ConfBall)

BallView?.navigationDelegate = self

BallView?.uiDelegate = self

view.addSubview(BallView!)

3.马甲包开启游戏以及外链

**\*\*\***

**1. 开启外链方式web使用js的window.open的函数，请检查各自马甲包的兼容性，必须是额外打开webview的方式，不能是原webview转跳（安卓需注意）。  
2. 如马甲包无法支援js的window.open的方式，web也提供一个事件呼叫可供马甲包接受地址做开启动作 ，会给予"openWindow 事件并且带上url给app（说明如下）。**

**3. 马甲包测试必须测试好开启三方外链游戏、三方存款页面这些的开启是否正常开启，如不确认是否正确请提供打好的马甲包供我方测试。**

**\*\*\***

**传递参数**

eventName ：字串：openWindow

params：JSON字串，例：{ url: "xxxxx" }

/\*\*

\* 如果监听到是openWindow 事件 需要新开一个WebView 加载url

\* 同时再新开的 WebView 监听页面的关闭事件

\* 在onActivityResult 监听关闭事件 同时调用closeGame 方法

\*/

if ("openWindow".equals(name)) {

Intent intent = new Intent(MainActivity.this, WebActivity.class);

intent.putExtra("url",data);

MainActivity.this.startActivityForResult(intent, 1);

}

**\*\*\***

**自动下分**

**关闭游戏原web监听视窗关闭自动下分的功能在马甲包会失效，所以需要马甲包在关闭外链游戏返回大厅之后主动回调web提供的下分函数**

**window.closeGame()**

**\*\*\***

 @Override

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {

        super.onActivityResult(requestCode, resultCode, data);

        if (resultCode == RESULT\_OK) {

            if (requestCode == 1) {

                if (webView == null) {

                    return;

                }

                webView.evaluateJavascript("j a v a s c r i p t :window.closeGame()", new ValueCallback<String>() {

                    @Override

                    public void onReceiveValue(String value) {

                    }

                });

            }

        }

    }

4.埋点事件

我司定义的事件名称如下：

登录：“login”

登出：“logout”

点击注册：“registerClick”

注册成功 ：“register”

点击充值：“rechargeClick”

\*首充成功 ：“firstrecharge”

\*复充成功 ：“recharge”

提现点击：“withdrawClick”

\*提现成功 ：“withdrawOrderSuccess”

进入游戏(包含三方与自营)：“enterGame”

领取vip奖励：“vipReward”

领取每日奖励：“dailyReward”

活动中心（进入页面）：“enterEventCenter”

任务中心（进入页面）：“enterTask”  
实时返水（进入页面）：“enterCashback”  
推广赚钱（进入页面）：“enterPromote”  
6张banner图（每张图的点击事件）：“bannerClick”

存取款的事件内容有额外带上金额以及币种

例：

存款: {\"amount\":\"200\",\"currency\":\"PHP\",\"isFirst\":0,\"success\":1}  
取款: {\"amount\":\"104\",\"currency\":\"PHP\",\"success\":1}

**5、马甲包埋点demo**

1、https://github.com/wgdemo/af-demo