

QT HarmonyNext SDK 集成手册（离线版本） v

2.0.2 版本

引入和配置SDK

1. 快速集成

- 1.1. 进入产品后台
- 1.2. 找到应用归属组织
- 1.3. 找到待集成应用

2. 集成代码明细

- 2.1. 参数准备
- 2.2. 引入SDK
 - 2.2.1. 离线安装SDK
 - 2.2.2. 引入配置（非必选）
 - 2.2.3. 权限配置
 - 2.2.4. 使用标准化的OHMUrl格式

基础集成

1. 域名设置

2. 合规初始化

- 2.1. 预初始化API
 - 2.1.1. API 说明
 - 2.1.2. 参数说明
- 2.2. 初始化API init
 - 2.2.1. API说明
- 2.3. 调用SDK初始化API
 - 2.3.1. 完整调用示例

3. 日志打印

埋点API

- 1. 如何查看埋点方案
- 2. 设备ID&账号ID&用户属性设置
 - 2.1. 设备ID

2.2. 账号ID

2.2.1. 用户登入

2.2.1.1. API说明

2.2.1.2. 参数说明

2.2.1.3. 示例代码

2.2.2. 用户登出

2.2.2.1. API说明

2.2.2.2. 示例代码

2.3. 用户属性上传

2.3.1. API说明

2.3.2. 示例代码

3. 渠道属性

3.1. H5链接拉活已安装APP应用

3.1.1. h5端唤起代码示例

3.1.2. 鸿蒙Next应用代码示例

3.1.2.1. 配置示例截图

3.1.2.2. 解析utm参数并设置为全局属性示例

3.2. H5链接拉新未安装APP应用

3.2.1. API说明

3.2.2. 调用示例

4. 全局属性

4.1. 注册全局属性

4.1.1. API 说明

4.1.2. 参数说明

4.1.3. 代码示例

4.2. 删除一个全局属性

4.2.1. API 说明

4.2.2. 参数说明

4.2.3. 代码示例

4.3. 根据key获取单个全局属性

4.3.1. API 说明

4.3.2. 参数说明

- 4.3.3. 代码示例
- 4.4. 获取所有全局属性
 - 4.4.1. API 说明
 - 4.4.2. 代码示例
- 4.5. 清除所有全局属性
 - 4.5.1. API 说明
 - 4.5.2. 代码示例
- 5. 页面浏览事件
 - 5.1. 自动采集
 - 5.2. 手动采集
 - 5.2.1. API 说明
 - 5.2.2. 参数说明
 - 5.2.3. 代码示例
 - 5.3. 设置页面事件属性
 - 5.3.1. API 说明
 - 5.3.2. 参数说明
 - 5.3.3. 代码示例
 - 5.4. 页面自动采集与页面手动采集混用
 - 5.4.1. API说明
 - 5.4.2. 参数说明
 - 5.4.3. 代码示例
- 6. 自定义事件
 - 6.1.1. API 说明
 - 6.1.2. 参数说明
 - 6.1.3. 代码示例
- 7. 手动采集应用启动、退出、激活事件
 - 7.1. 手动上报应用激活事件
 - 7.1.1. API 说明
 - 7.1.2. 参数说明
 - 7.1.3. 代码示例
 - 7.2. 手动上报应用启动事件
 - 7.2.1. API 说明

7.2.2. 参数说明

7.2.3. 代码示例

7.3. 手动上报应用退出事件

7.3.1. API 说明

7.3.2. 参数说明

7.3.3. 代码示例

应用内H5数据上报

1. H5桥接使用场景

场景1：如何将H5数据同时上报至H5应用和APP应用

操作说明

上报日志

场景2：如何将H5数据仅上报至唯一APP应用中

操作说明

上报日志

场景3：如何将H5数据仅上报给H5应用中

操作说明

上报日志

2. H5桥接原理说明

2.1. 鸿蒙Next 应用端代码

桥接API说明

参数说明

代码示例

2.2. H5应用侧代码

仅通过APP通道上报，关闭H5通道上报日志API

H5 全局属性全局生效接口

预制事件和预制属性

预制事件

预制属性

引入和配置SDK

使用HarmonyNext SDK必备操作，缺少任意步骤都可能造成集成失败或者数据丢失

1. 快速集成

在QuickTracking后台，为每一个应用生成了专属的集成代码，可以根据产品内的引导进行集成。

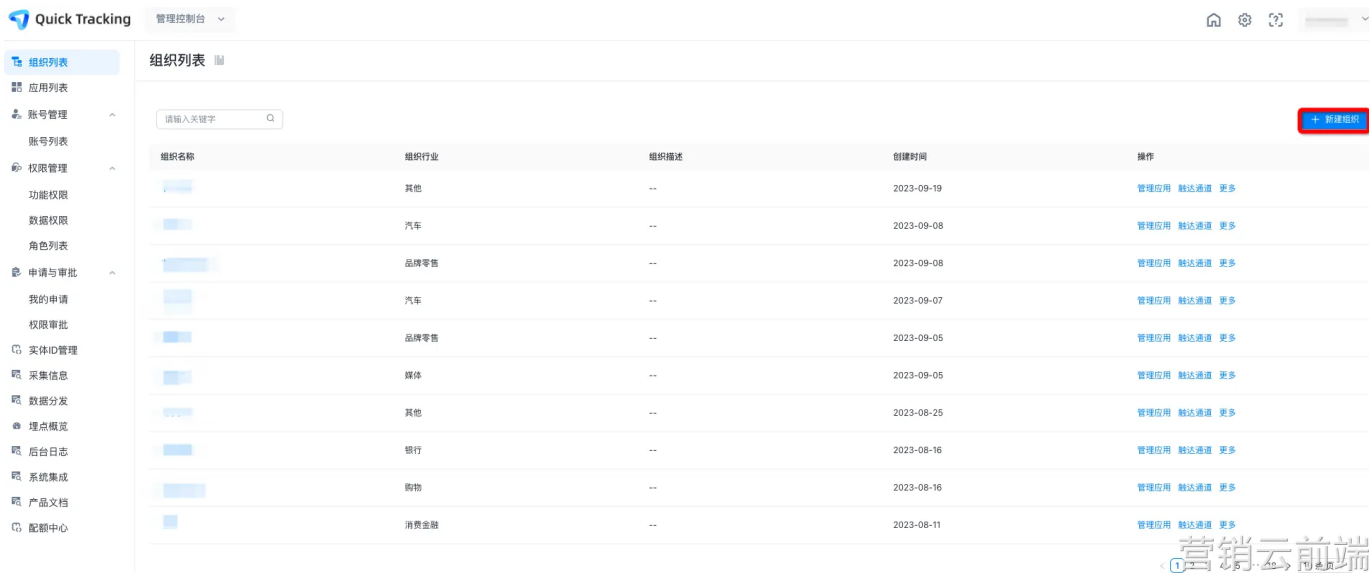
1.1. 进入产品后台

QuickTracking【首页】 – 【管理控制台】



1.2. 找到应用归属组织

在【管理控制台】进入【组织列表】功能，组织列表展示当前创建的组织，找到应用归属组织



1.3. 找到待集成应用



2. 集成代码明细

如果因为特定原因，无法访问上述页面，可以通过查看本章节进行SDK集成

2.1. 参数准备

appKey：在应用列表中获取

收数域名：在【管理控制台】-【采集信息】模块中获取

2.2. 引入SDK

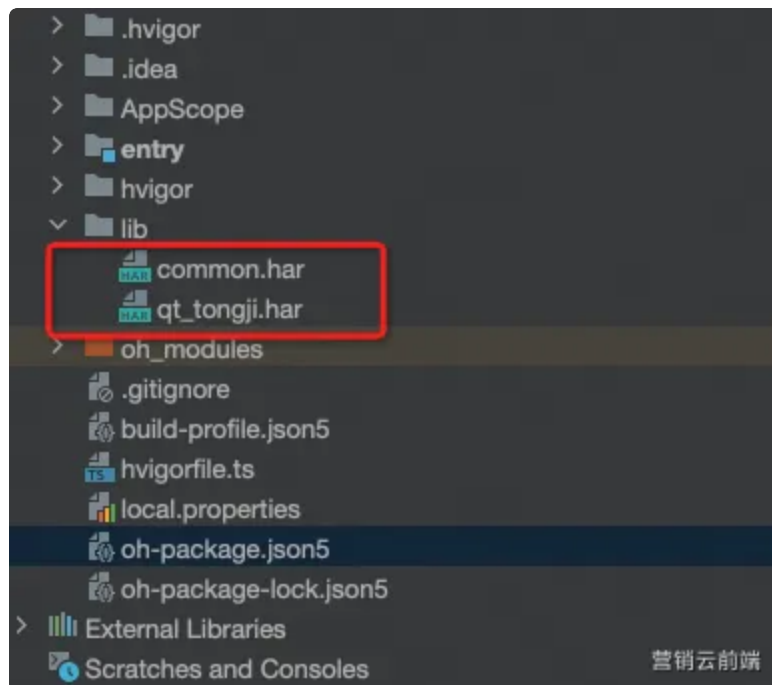
系统 API 要求，HarmonyNext API 12及以上，目前仅支持普通应用的Stage模型，暂不支持元服务

2.2.1. 离线安装SDK

引入离线版本鸿蒙SDK

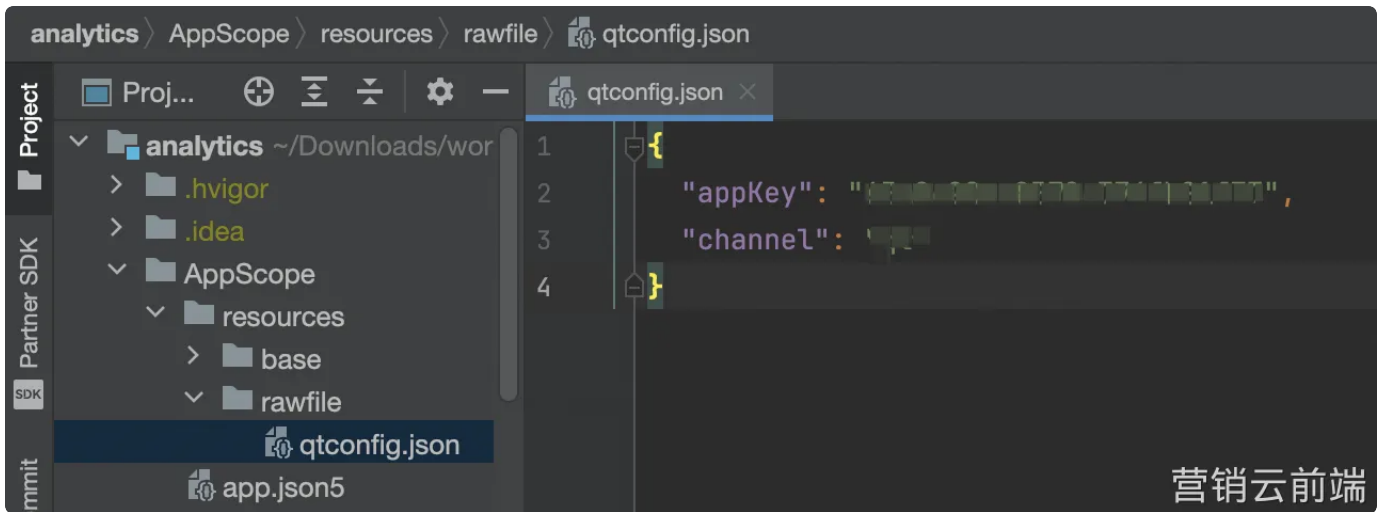
```
JSON |
1 {
2   "modelVersion": "5.0.0",
3   "description": "Please describe the basic information.",
4   "dependencies": {
5     "@quicktracking/common": "file:./lib/common.har",
6     "@quicktracking/analytics": "file:./lib/qt_tongji.har",
7   },
8   "devDependencies": {
9     "@ohos/hypium": "1.0.18",
10    "@ohos/hamock": "1.0.0"
11  }
12 }
13
```

SDK文件位置



2.2.2. 引入配置（非必选）

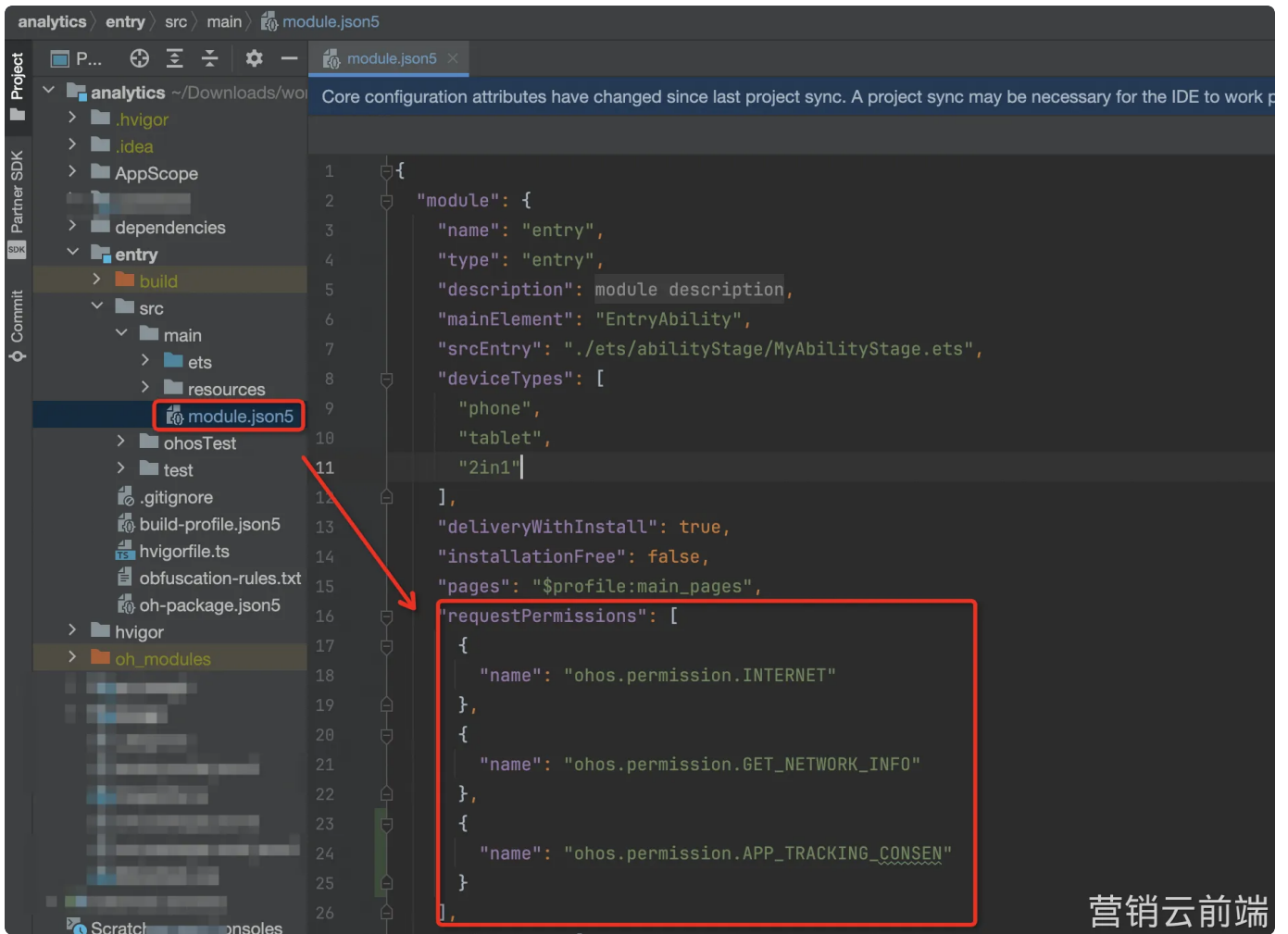
在项目工程的 **AppScope/resources/rawfile** 目录下新增一个配置文件 **qtconfig.json**



JSON

注意如果初始化的时候构建 `UMConfig` 时传入上述信息，以初始化时的值为准

在需要使用SDK的模块的 `module.json5` 文件中添加权限声明



营销云前端

！！注意：根据华为官网文档说明对于OAID的采集需要用户主动在设备设置界面开启，具体步骤请参考[华为官方文档](#)

注意

和其他user_grant权限不一样的是，当前版本上，调用requestPermissionsFromUser接口申请"ohos.permission.APP_TRACKING_CONSENT"，系统不会有弹框，而是直接将该应用的获取OAID的权限置为禁止。如有需要，应用可自行弹框引导用户到设置界面手动开启该权限。为避免后续系统功能变化而引发兼容性问题，应用应仍严格按照user_grant权限的申请流程进行"ohos.permission.APP_TRACKING_CONSENT"权限的申请。

营销云前端

2.2.4. 使用标准化的OHMUrl格式

请设置useNormalizedOHMUrl为true，查看工程级build-profile.json5：

```
1 {  
2   "app": {  
3     "signingConfigs": [],  
4     "products": [  
5       {  
6         "name": "default",  
7         "signingConfig": "default",  
8         "compatibleSdkVersion": "5.0.0(12)",  
9         "runtimeOS": "HarmonyOS",  
10        "buildOption": {  
11          "strictMode": {  
12            "useNormalizedOHMUrl": true  
13          }  
14        }  
15      }  
16    ],  
17  }  
18 }  
19 }
```

基础集成

1. 域名设置

在预初始化方法调用之前，开发者需要在调用SDK任意其他API之前最先调用 `setTrackDomain` API 设置私有化环境的收数域名。

```

1  import { setTrackDomain } from '@quicktracking/analytics';
2
3  /**
4   * 设置上传统计日志的主域名和备用域名。
5   * SDK会优先将统计数据上报到主域名，失败的情况下会再尝试将数据上报到备用域名。
6   * 主域名primaryDomain或不能传入null或者空串，如果传入null或者空串，SDK会打印"收数域
   * 名不能为空，请检查域名设置！"异常日志。
7   * 备用域名可以传入null或者空串，此时SDK认为备用域名和主域名完全相同。SDK上传数据失败后
   * 第二次也会向主域名上报数据。
8   * 传入的域名参数应该包含"https://" 前缀。
9   */
10
11 function setTrackDomain(primaryDomain: string, standaryDomain?: string)

```

参数	含义
primaryDomain	上传日志的主收数域名地址
standaryDomain	上传日志的备用收数域名地址

备注：仅支持HTTP、HTTPS协议前缀，如果不指定协议前缀，默认为HTTPS协议发送

2. 合规初始化

由于工信部的合规要求，应用在用户同意隐私政策前不可以获取任何个人信息，因此QuickTracking鸿蒙Next的SDK的初始化步骤分成 **预初始化** 和 **初始化** 两步，完成合规初始化需要进行以下操作：

1. 您需要确保App有《隐私政策》，并且在用户首次启动App时就弹出《隐私政策》并取得用户同意。
2. 您必须告知用户您选择QuickTrackingSDK采集服务，请在《隐私政策》中增加如下参考条款：“我们的产品集成QuickTracking SDK，QuickTracking SDK需要收集您的OAID/华为AAID/SIM卡 IMSI 信息/硬件序列号/MCC（移动国家编码）、MNC（移动网号）以提供统计分析服务。”
3. 用户同意隐私政策后才可以正式初始化QuickTracking SDK。

2.1. 预初始化API

调用SDK预初始化API，预初始化API调用不会采集设备信息，也不会向QuickTracking收数服务上报数据

2.1.1. API 说明

```

1  import { preInit } from '@quicktracking/analytics';
2
3  function preInit(cfg: UMConfig):void

```

2.1.2. 参数说明

参数	类型	含义	是否必填
context	common.Application Context	应用环境上下文变量	是
appKey	string	应用唯一标识appKey，入参的appkey一定要与QT后台保持一致 该appKey会跟随每一条事件日志进行上报，用于QT的应用平台数据区分标识	是
plugins	BasePlugin BasePlugin[]	当前开启的插件	是
channel	string	应用发布的渠道，QT分析平台的“系统属性”中“升级渠道”的数据来源	否
enableLog	boolean	是否开启调试日志，默认值是false	否
enableJSBridge	boolean	是否开启H5桥接功能，默认值是false	否
enableAutoTrackApplication	boolean	是否开启自动采集应用生命周期，默认值是true	否
enableAutoTrackPage	boolean	是否开启自动采集页面，默认值是true	否

2.2. 初始化API init

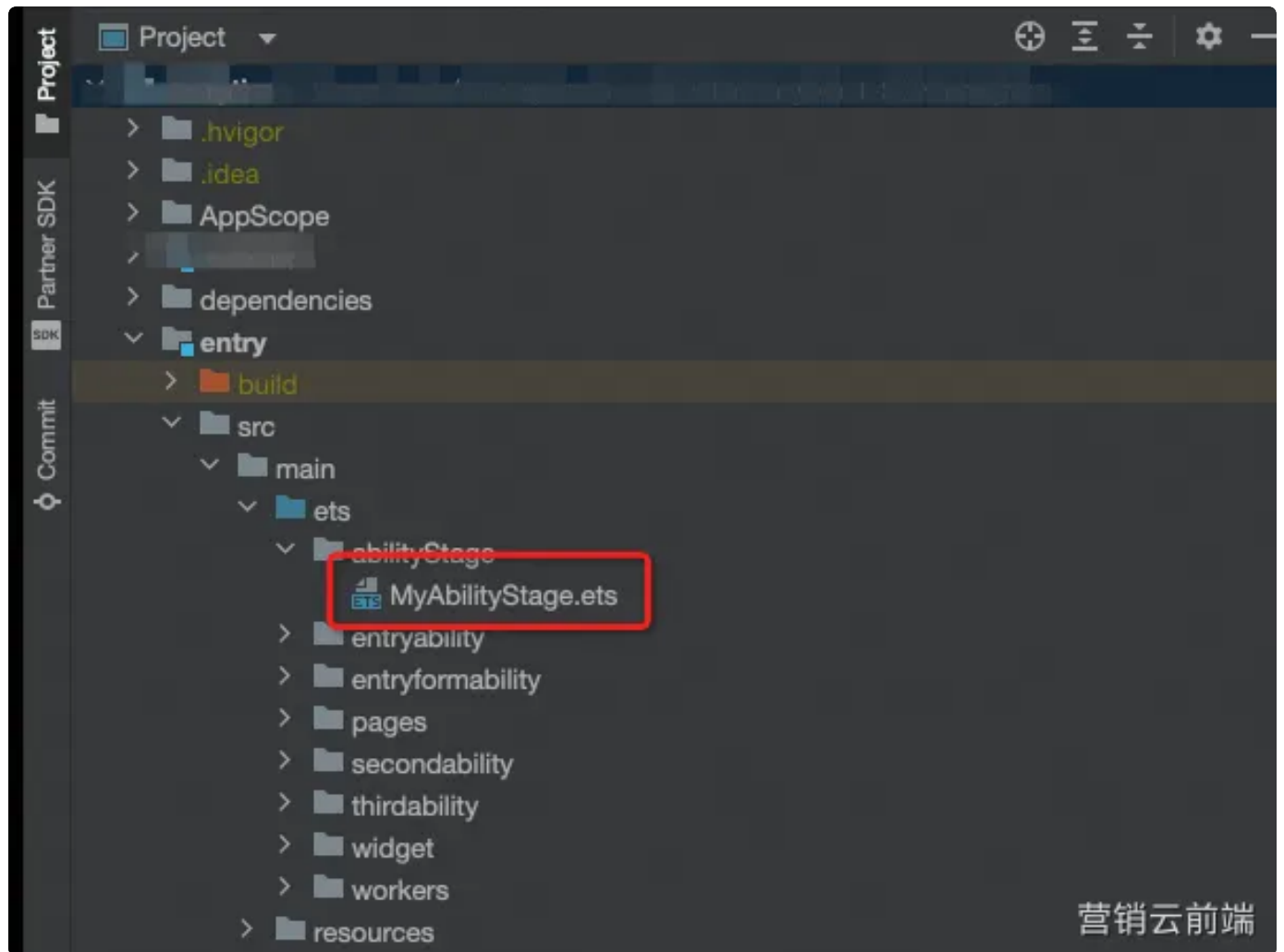
用户同意隐私协议后，调用SDK初始化API，只有调用了init方法，才会开始日志的采集和传输。

2.2.1. API说明

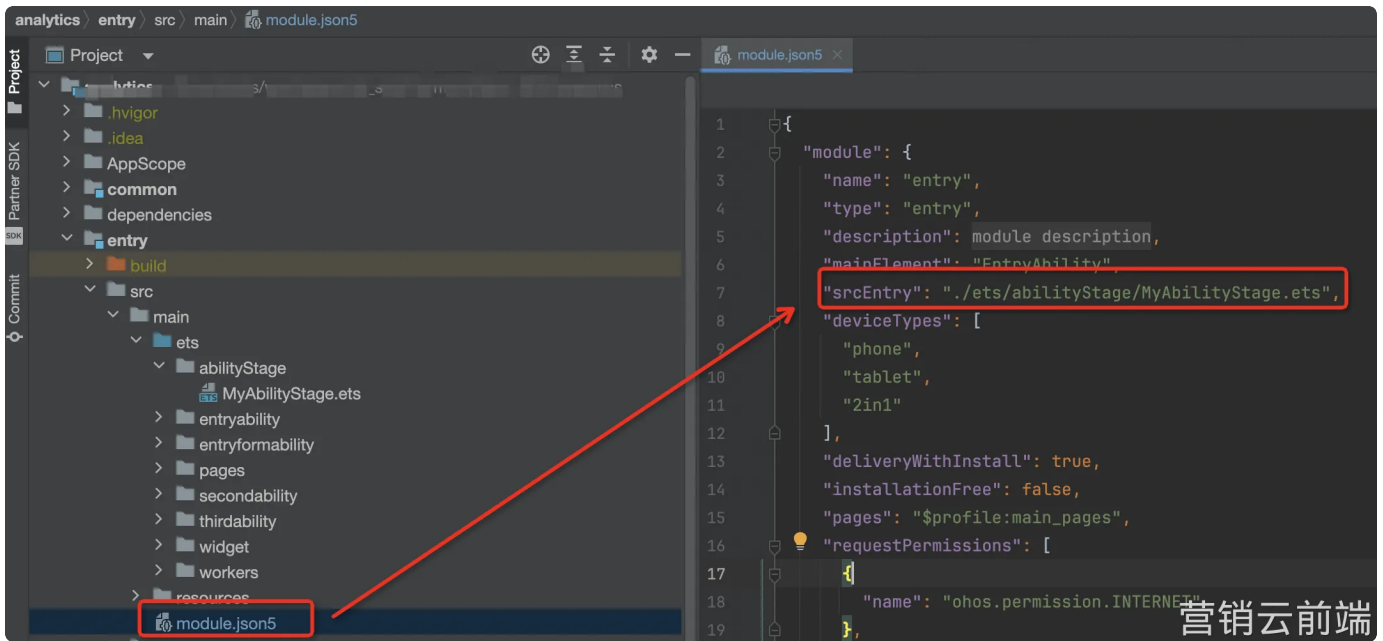
```
TypeScript |
1  import { init } from '@quicktracking/analytics';
2
3  function init():Promise<void>
```

2.3. 调用SDK初始化API

在应用模块目录下，添加abilityStage工程文件，例如：`entry/src/main/ets/abilityStage/MyAbilityStage.ets`，具体位置截图为



在模块的module.json5文件中添加srcEntry，指向abilityStage文件的地址



2.3.1. 完整调用示例

```
entry/src/main/ets/abilityStage/MyAbilityStage.ets TypeScript |

1 import AbilityStage from '@ohos.app.ability.AbilityStage';
2 import { preInit, InternalPlugin, setLogEnabled, init, setTrackDomain } from '@quicktracking/analytics';
3
4 setLogEnabled(true); //开启调试日志
5 setTrackDomain("主收数域名", "副收数域名"); //设置采集收数域名
6
7 export default class MyAbilityStage extends AbilityStage {
8   onCreate() {
9     preInit({
10       appKey: '您的AppKey'
11       context: this.context.getApplicationContext(),
12       enableJSBridge: true,
13       enableAutoTrackApplication: true,
14       enableAutoTrackPage: true,
15       plugins: [new InternalPlugin()]
16     });
17     init();
18   }
19 }
```

注意：！！在适当位置（preInit方法调用之后），经用户授权同意隐私政策后调用init方法，才会开始日志的采集和传输。

3. 日志打印

可通过主动调用 `setLogEnable` API 开启或关闭sdk日志信息

▼ TypeScript |

```
1  import { setLogEnabled } from '@quicktracking/analytics';
2
3  function setLogEnabled(enable: boolean):void
```

参数	含义
enable	是否开启sdk日志信息，默认不开启false

注意：

- 如果查看初始化过程中的LOG，一定要在调用初始化方法之前将LOG开关打开
- 日志分为三种等级
 - error 打印sdk集成或运行时的错误信息
 - warn 打印sdk的警告信息
 - info 打印sdk的提示信息

埋点API

1. 如何查看埋点方案

在进行埋点前，需要确定在哪里埋点、埋哪些点等，即需要梳理清楚明确的埋点需求。在QuickTracking平台中将明确的埋点需求称为埋点方案，并为埋点方案设计了规范模板。如下：

产品埋点方案										
事件主体				参数名称	参数ID	参数值类型	参数值示例或说明			
事件主体				账号ID						
				设备ID						
用户属性				参数名称	参数ID	参数值类型	参数值示例或说明			
用户属性				用户ID	userid					
				用户类型	user_type					
渠道属性				参数名称	参数ID	参数值类型	参数值示例或说明			
				投放渠道	utm_channel					
全局参数				参数名称	参数ID	参数值类型	参数值示例或说明			
全局属性				是否绑定车辆	is_banding					
所在页面名称	页面编码	事件编号	事件归类	事件名称	事件编码	参数名称	参数上级id	参数值	参数类型	采集范围与上报机制说明
无	无	1	点击事件-2101	底部按钮点击	bottom_clk	按钮名称	button_name		字符串	按钮点击时上报
首页	home_page	2	页面事件-2001	首页	home_page					页面加载时上报
		3	点击事件-2101	首页_搜索点击	hp_search_clk					搜索框点击时上报
		4	曝光事件-2201	首页_推荐banner曝光	hp_rec_banner_exp	life_cycle	banner位置	banner_sit	字符串	banner曝光时上报
		5	点击事件-2101	首页_推荐banner点击	hp_rec_banner_clk	life_cycle	banner位置	banner_sit	字符串	banner点击时上报
		6	点击事件-2101	首页_细分市场选择器点击	hp_market_segment_sele_clk	细分市场	market_segment		字符串	按钮点击时上报
		7	点击事件-2101	首页_车系名称选择器点击	hp_car_series_sele_clk	细分市场	market_segment		字符串	按钮点击时上报
		8	曝光事件-2201	首页_life_cycle选择器曝光	hp_life_cycle_sele_exp	车系名称	car_series_name		字符串	按钮点击时上报
		9	点击事件-2101	首页_life_cycle选择器点击	hp_life_cycle_sele_clk	细分市场	market_segment		字符串	车型卡片曝光时上报
						车系名称	car_series_name		字符串	车型卡片曝光时上报

在埋点方案中，明确的所需埋点内容有：

- 事件主体：指“谁”触发了这个事件，分为**设备ID**和**账号ID**，上报的事件务必具备其中之一。
 - 设备ID：HarmonyNext应用的**默认设备ID为应用级别唯一的设备ID**，由**Quicktracking自动生成**；
 - 账号ID：客户端用户登录后账号标识，当一个用户在不同的设备进行登录时，设备ID会发生变化，但是账号ID不会发生变化。例如一个用户使用手机和pad分别登录。
- 用户属性：针对账号ID的属性，例如账号ID为“testdemo@111”的用户，“生日”为“1999-02-13”，“会员等级”为“铂金”等。“生日”和“会员”等级就为用户属性。
- 渠道属性：广告投放的属性，例如投放渠道、投放方式、投放内容等。
- 全局属性：在全局设置一次后，每一个事件都会携带的属性
- 页面浏览事件：页面加载时上报的事件（埋点方案中页面编码和事件编码相等的事件，也是标记为蓝色的事件）
- 点击、曝光、自定义事件：客户端用户与客户端发生任意交互时上报的事件

2. 设备ID&账号ID&用户属性设置

2.1. 设备ID

当前版本，即@quicktracking/analytics_1.0.0版本设备ID的生成由QuickTrackingSDK自动生成，暂不支持自定义设备ID和获取当前生成的设备ID

2.2. 账号ID

QuickTracking SDK在统计用户时以设备为标准，如果需要统计自身的账号，请使用以下方法

2.2.1. 用户登入

2.2.1.1. API说明

TypeScript |

```
1 import { onProfileSignIn } from '@quicktracking/analytics';
2
3 function onProfileSignIn(provider: string, puid: string):void
```

2.2.1.2. 参数说明

参数	类型	含义	是否必填
provider	string	用户昵称，非空字符串，且长度不超过32	是
puid	string	用户账号ID，非空字符串，且长度不超过64	是

注意：账号ID设置后将被存入本地存储，只有卸载App、清空应用数据或者调用下述的登出接口时，账号ID才会失效，否则每一个事件都将携带账号ID。

2.2.1.3. 示例代码

TypeScript |

```
1 import { onProfileSignIn } from '@quicktracking/analytics';
2
3 Button('登入').onClick(() => {
4   onProfileSignIn("QuickTrackingUser", "QuickTrackingUser_demo");
5 })
```

2.2.2. 用户登出

如果不再需要绑定用户账号，可以调用SDK提供的用户登出方法，调用后，SDK不再发送用户账号相关内容

2.2.2.1. API说明

```
1 import { onProfileSignOff } from '@quicktracking/analytics';
2
3 function onProfileSignOff():void
```

2.2.2.2. 示例代码

```
1 import { onProfileSignOff } from '@quicktracking/analytics';
2
3 Button('登出').onClick(() => {
4   onProfileSignOff();
5 })
```

2.3. 用户属性上传

使用事件编码固定为 `$$_user_profile` 的自定义事件上传，该事件所携带的事件属性会被作为用户属性放在用户表中。

2.3.1. API说明

参见自定义事件API

2.3.2. 示例代码

```
1 import { trackEvent } from '@quicktracking/analytics';
2
3 Button('上传用户属性').onClick(() => {
4   trackEvent("$$_user_profile", {
5     sex: "boy",
6     age: 13,
7     favorite: "basketball"
8   });
9 })
```

3. 渠道属性

有两种场景需要统计渠道属性信息：

1. H5链接拉活已安装APP应用
2. H5链接拉新未安装APP应用

3.1. H5链接拉活已安装APP应用

唤起App的URL Scheme中携带这些渠道属性，且属性key务必以“utm_”开头，因为SDK识别的关键字为“utm_”。例如：<URL scheme>?utm_channel=gzh

注意：！！！！ 如果渠道属性已经与市面上渠道投放公司进行了合作，无法使用utm_开头，可以使用全局属性API将渠道属性进行埋点上报（属性key依然需要以utm_开头）

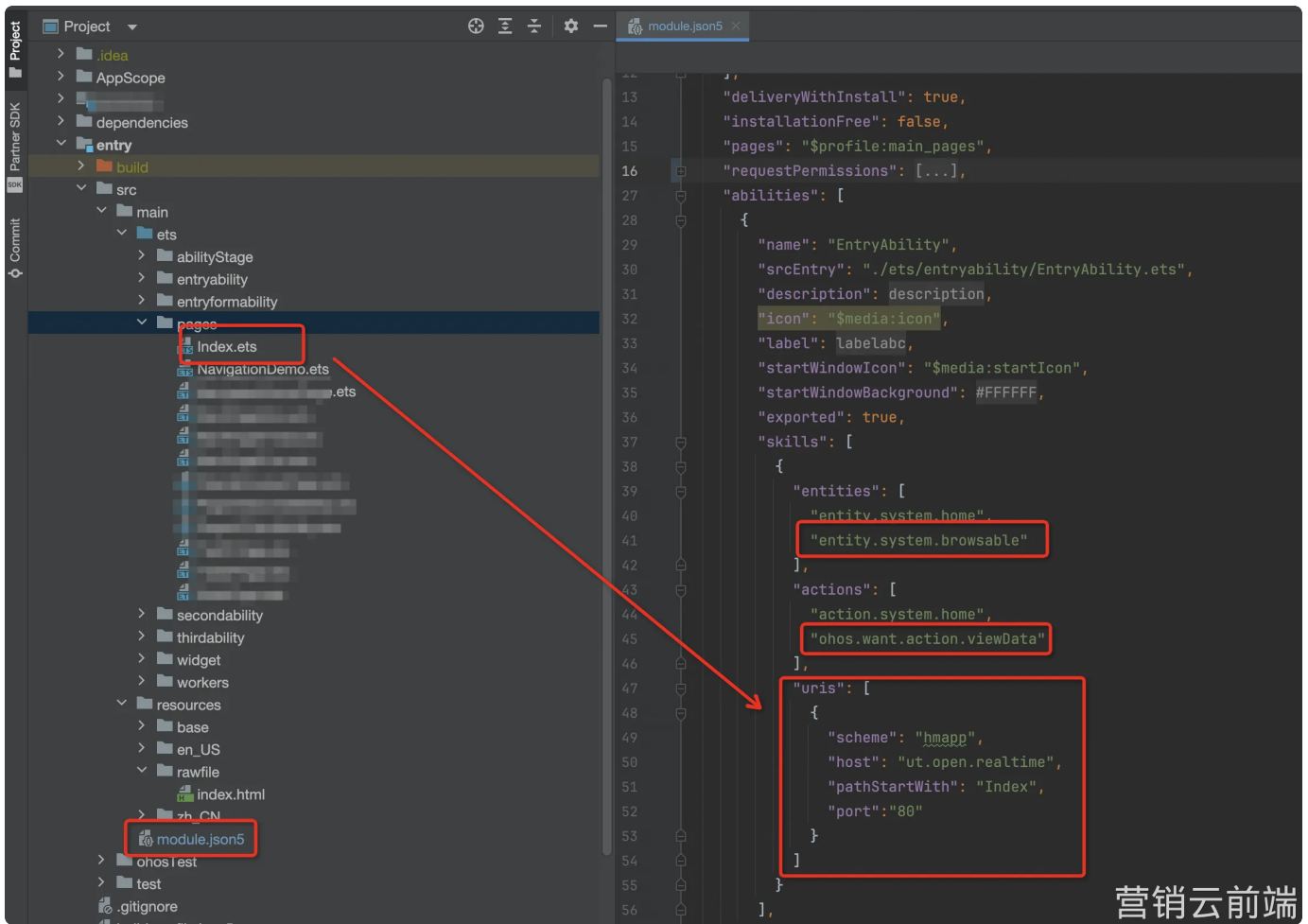
3.1.1. h5端唤起代码示例

```
1 <html>
2 <script>
3     function openHarmonyApp() {
4         var url = "hmap://ut.open.realtime:80/Index?utm_a=aaa&utm_b=bbb";
5         alert(url)
6         window.location.href = url
7     };
8 </script>
9 <body>
10     <button onclick="openHarmonyApp()">唤起鸿蒙应用</button>
11 </body>
12 </html>
```

3.1.2. 鸿蒙Next应用代码示例

应用如能被H5链接唤起，需对模块的skills标签进行额外配置，具体请参考[鸿蒙Next官网文档](#)

3.1.2.1. 配置示例截图



3.1.2.2. 解析utm参数并设置为全局属性示例


```

1  import AbilityConstant from '@ohos.app.ability.AbilityConstant';
2  import hilog from '@ohos.hilog';
3  import UIAbility from '@ohos.app.ability.UIAbility';
4  import Want from '@ohos.app.ability.Want';
5  import window from '@ohos.window';
6  import {registerGlobalProperties, trackAppInstall } from '@quicktracking/analytics'
7
8
9  export default class EntryAbility extends UIAbility {
10   handleWant(want: Want) {
11     if (want.uri && want.uri.indexOf('utm_')) {
12       const dUri = decodeURI(want.uri)
13       const utmArgs:Record<string, string> = {}
14       const paramsString = dUri.split("?")[1]
15       const paramsArray = paramsString.split("&")
16       for (let i of paramsArray) {
17         const kvs = i.split("=")
18         const k: string = kvs[0]
19         const v = kvs[1]
20         utmArgs[k] = v
21       }
22       registerGlobalProperties(utmArgs)
23     }
24   }
25   //H5链接拉起APP应用，热启动
26   onNewWant(want: Want, launchParam: AbilityConstant.LaunchParam): void {
27     this.handleWant(want)
28   }
29
30   //H5链接拉起APP应用，冷启动
31   onCreate(want: Want, launchParam: AbilityConstant.LaunchParam): void {
32     this.handleWant(want)
33
34     //!!! 仅在enableAutoTrackApplication = false时需要调用
35     trackAppInstall({
36       browser: `Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36`,
37       page_name: "ManIndex", //唤起页面的页面编码，可选，建议配置
38       url: "pages/Index" //唤起页面的页面路径，可选，建议配置
39     }, {
40       man_app_p1: 111, //自定义属性，可选
41       man_app_p2: 222
42     })
43

```

```

44     }
45
46     onWindowStageCreate(windowStage: window.WindowStage): void {
47         windowStage.loadContent('pages/Index', (err, data) => {
48             if (err.code) {
49                 return;
50             }
51         });
52     }
53 }

```

3.2. H5链接拉新未安装APP应用

该场景下，如果仅是H5链接中携带“utm_”参数，已经无法做到下载APP后的启动事件携带“utm_”参数。所以需要进行“H5唤起事件”与“应用激活事件”做关于“IP地址和浏览器UserAgent”的模糊匹配。

1. 当用户在H5中点击「唤起/下载App」的按钮时，上报“应用唤起事件（\$\$_app_link）”，在事件中需要携带唤起App的appKey和渠道属性。

```

▼ JavaScript
1  //示例
2  aplus_queue.push({
3      action: 'applus.recordAppLink',
4      arguments: [{
5          targetAppKey: '要唤起的应用appKey', // 必填，要唤起的应用appKey
6          custom1: 'custom1', // 选填，自定义参数
7          ...
8      }]
9  })

```

2. App下载后的通过自动或者手动上报第一次启动事件“应用激活事件（\$\$_app_install）”完成关联动作。
 - 如果 SDK 默认配置 enableAutoTrackApplication = true，自动上报\$\$_app_install应用激活事件
 - 如果 SDK 默认配置 enableAutoTrackApplication = false，需开发者主动上报应用激活事件

3.2.1. API说明

```

1  import { trackAppInstall } from '@quicktracking/analytics'
2
3  function trackAppInstall(params: {
4      browser?: string,
5      [key: string]: string
6  }, customProps?: object)

```

3.2.2. 调用示例

参见3.1.2鸿蒙Next应用端代码示例

4. 全局属性

4.1. 注册全局属性

4.1.1. API 说明

```

1  import { registerGlobalProperties } from '@quicktracking/analytics'
2
3  function registerGlobalProperties(params?: Record<string, string | number |
    string[]>)

```

4.1.2. 参数说明

参数	含义
params	<p>kv键值对</p> <p>k属性名，string类型，应为长度为128个字符以内的非空（包括undefined）字符串</p> <p>v属性值，string类型或number类型或字符串数组类型</p> <ul style="list-style-type: none"> 如果是字符串类型，应为长度为256个字符以内的字符串 如果是字符串数组类型，数组的长度不能超过100，含100

4.1.3. 代码示例

TypeScript |

```
1  import { registerGlobalProperties } from '@quicktracking/analytics'
2
3  Button('注册全局属性').onClick(async () => {
4    registerGlobalProperties({
5      "a": 1,
6      "b": 2
7    }) // 输出结果: {a: 1, b: 2}
8  })
9  Button('重新注册全局属性').onClick(async () => {
10   registerGlobalProperties({
11     "b": 6,
12     "c": 4
13   }) // 输出结果: {a: 1, b: 6, c: 4}
14 })
```

注意：如果和已经存在的全局属性key重复，则更新已有值；如果和已经存在的全局属性key不一致，则插入新的全局属性。

4.2. 删除一个全局属性

4.2.1. API 说明

TypeScript |

```
1  import { unregisterGlobalProperty } from '@quicktracking/analytics'
2
3  function unregisterGlobalProperty(key?: string)
```

4.2.2. 参数说明

参数	含义
key	需要删除的属性key名称，key应该有值并且是128个字符以内的字符串

4.2.3. 代码示例

```

1 import { unregisterGlobalProperty } from '@quicktracking/analytics'
2
3 Button('删除一个全局属性值').onClick(async () => {
4   unregisterGlobalProperty('a')
5 })

```

4.3. 根据key获取单个全局属性

4.3.1. API 说明

```

1 import { getGlobalProperty } from '@quicktracking/analytics'
2
3 function getGlobalProperty(key?: string): string | number | string[] | unde
  fined

```

4.3.2. 参数说明

参数	含义
key	属性key名称，key应该有值并且是128个字符以内的字符串

4.3.3. 代码示例

```

1 import { getGlobalProperty } from '@quicktracking/analytics'
2
3 Button('根据Key获取单个全局属性').onClick(async () => {
4   promptAction.showToast({
5     message: `获取的属性b = ${getGlobalProperty("b")}`,
6     duration: 3000
7   })
8 })

```

4.4. 获取所有全局属性

4.4.1. API 说明

▼ TypeScript |

```
1 import { getGlobalProperties } from '@quicktracking/analytics'
2
3 function getGlobalProperties(): Record<string, string | number | string[]>
```

4.4.2. 代码示例

▼ TypeScript |

```
1 import { getGlobalProperties } from '@quicktracking/analytics'
2
3 Button('获取所有全局属性').onClick(async () => {
4   promptAction.showToast({
5     message: `全局属性为\n${JSON.stringify(getGlobalProperties())}`,
6     duration: 5000
7   })
8 })
```

4.5. 清除所有全局属性

4.5.1. API 说明

▼ TypeScript |

```
1 import { clearGlobalProperties } from '@quicktracking/analytics'
2
3 function clearGlobalProperties()
```

4.5.2. 代码示例

```

1 Button('清除所有的全局属性').onClick(async () => {
2     clearGlobalProperties()
3     promptAction.showToast({
4         message: `全局属性为\n${JSON.stringify(getGlobalProperties())}`,
5         duration: 5000
6     })
7 })

```

5. 页面浏览事件

QuickTracking 鸿蒙Next SDK 提供页面自动采集和手动采集两种能力

5.1. 自动采集

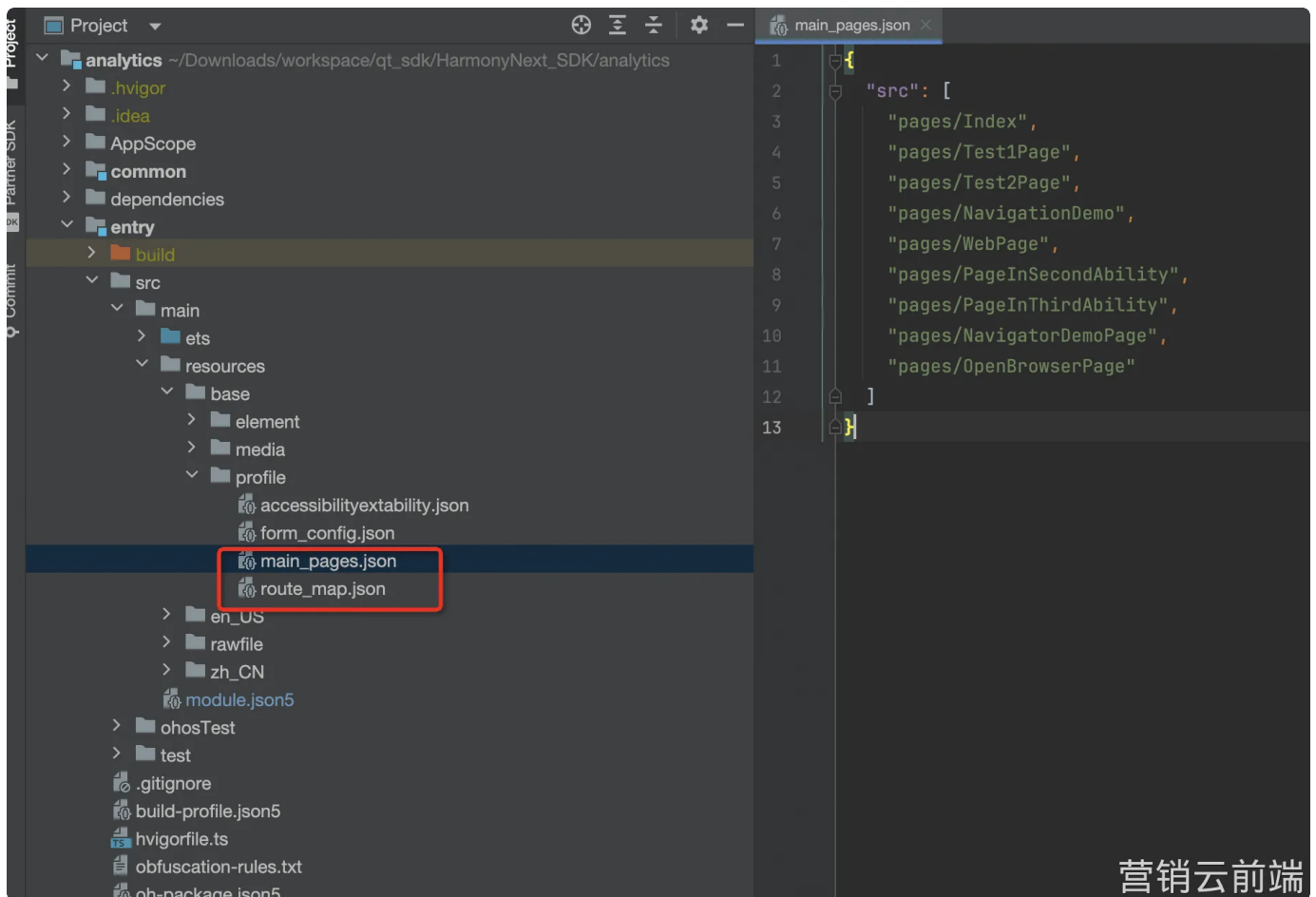
如果 SDK 默认配置 `enableAutoTrackPage = true`，则自动采集页面浏览事件，详情参见预初始化API说明

路由模式	自动页面浏览事件-页面编码	自动页面浏览事件-页面路径
RouterPage	触发生命周期的routerPage页面的名称	触发生命周期的routerPage页面的路径
NavDestination	NavDestination组件的名称	NavDestination组件的名称

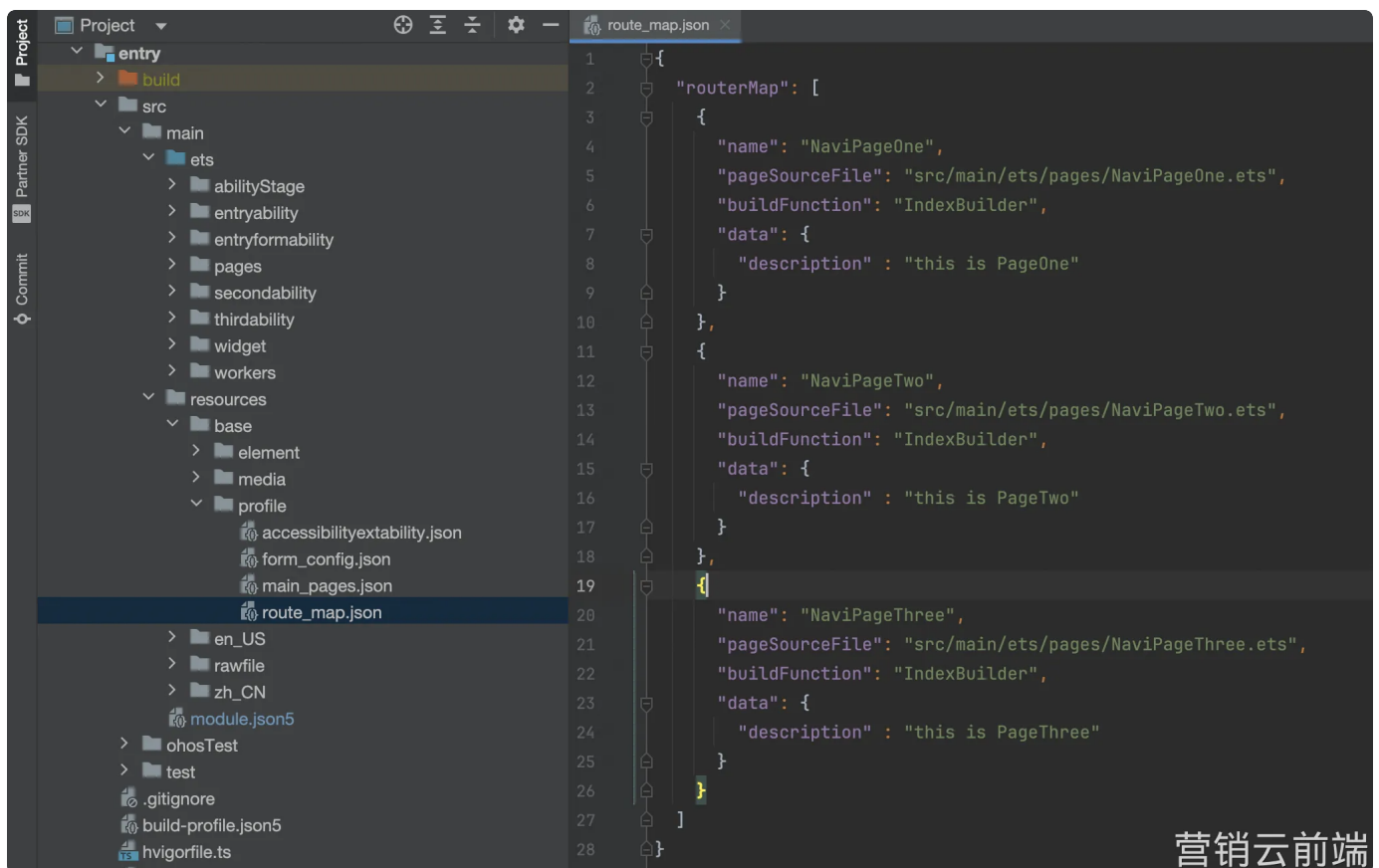
5.2. 手动采集

相比iOS系统或者Android系统，鸿蒙Next系统从概念上没有严格意义上的页面组件，任何通过装饰器语法声明的组件只要在 `main_pages.json` 文件或者 `route_map.json` 文件中添加了标识，都可以认为是业务意义上的页面组件。

- 定义在main_pages.json文件中的RouterPage页面



- 定义在route_map.json文件中NavDestination页面



针对上述两种场景，SDK 提供一组需成对调用的API `onPageStart` 和 `onPageEnd` 进行页面埋点。

5.2.1. API 说明

TypeScript |

```
1 import { onPageStart, onPageEnd } from '@quicktracking/analytics'
2
3 function onPageStart(context: PageContext | null, pageName: string, pageUrl
?: string) //页面进入时调用
4 function onPageEnd(context: PageContext | null, pageName: string, pageUrl?:
string) //页面退出时调用
```

5.2.2. 参数说明

参数	含义
context	页面环境上下文变量
pageName	页面事件的事件编码，长度为128位字符以内的字符串，且不能为空
pageUrl	页面url，若某些页面调用逻辑拿不到context，需要开发者主动传值，否则数据分析会不准备。默认值"undefined"

5.2.3. 代码示例

- RouterPage 页面

```

1  import { onPageStart, onPageEnd, updatePageProperties, skipAutoTrack } from '@quicktracking/analytics';
2
3  @Entry
4  @Component
5  struct RouterPageDemo {
6      @State message: string = 'RouterPageDemo';
7
8      onPageShow(): void {
9          //skipAutoTrack(this)
10         onPageStart(this, "RouterPageDemo")
11         updatePageProperties(this, "RouterPageDemo", {
12             a: 1,
13             b: 2
14         })
15     }
16
17     onPageHide() {
18         onPageEnd(this, "RouterPageDemo")
19     }
20
21     build() {
22         Column() {
23             Text(this.message)
24                 .id('HelloWorld')
25                 .fontSize(50)
26                 .fontWeight(FontWeight.Bold)
27                 .alignRules({
28                     center: { anchor: '__container__', align: VerticalAlign.Center }
29                 },
30                 middle: { anchor: '__container__', align: HorizontalAlign.Center }
31             )
32             .height('100%')
33             .width('100%')
34         }
35     }

```

- NavDestination 页面

```

1  import { onPageStart, updatePageProperties, onPageEnd, skipAutoTrack } fro
   m '@quicktracking/analytics';
2
3  @Builder
4  export function IndexBuilder(name: string, param: Object) {
5      NaviPageOne()
6  }
7
8  @Component
9  export default struct NaviPageOne {
10
11      @State name: string = 'NaviPageOne';
12
13      pathInfos: NavPathStack = new NavPathStack();
14
15      build() {
16          NavDestination() {
17              Column() {
18                  Text("")
19                      .id('NaviPageOneHelloWorld')
20                      .fontSize(50)
21                      .fontWeight(FontWeight.Bold)
22                      .alignRules({
23                          center: { anchor: '__container__', align: VerticalAlign.Center
24                          },
25                          middle: { anchor: '__container__', align: HorizontalAlign.Cent
26                          er }
27                      })
28                  .height('100%')
29                  .width('100%')
30              }
31              .onReady((context: NavDestinationContext) => {
32                  this.pathInfos = context.pathStack
33              })
34              .title(this.name)
35              .id(this.name)
36              .onShown(() =>{
37                  //skipAutoTrack(this)
38                  onPageStart(this, "NaviPageOne")
39                  updatePageProperties(this, "NaviPageOne", {
40                      a: 1,
41                      b: 2
42                  })
43              })
44          }
45      }
46  }

```

```

43     .onHidden(() => {
44         onPageEnd(this, "NaviPageOne")
45     })
46 }
47 }

```

5.3. 设置页面事件属性

5.3.1. API 说明

TypeScript |

```

1  import { updatePageProperties } from '@quicktracking/analytics';
2
3  function updatePageProperties(context: PageContext, pageName: string, param
   s?: Record<string, string | number | string[]>, pageUrl?: string)

```

5.3.2. 参数说明

参数	含义
context	页面环境上下文变量
pageName	页面事件的事件编码，长度为128位字符以内的字符串，且不能为空
params	<p>页面事件的事件属性，kv键值对集合</p> <p>k属性名，string类型，应为长度为128个字符以内的非空（包括undefined）字符串</p> <p>v属性值，string类型 或 number类型 或 字符串数组类型</p> <ul style="list-style-type: none"> 如果是字符串类型，应为长度为256个字符以内的字符串 如果是字符串数组类型，数组的长度不能超过100，含100

pageUrl	页面url，若某些页面调用逻辑拿不到context，需要开发者主动传值，否则数据分析会不准备。默认值"undefined"
---------	---

5.3.3. 代码示例

见5.2.3节代码示例

5.4. 页面自动采集与页面手动采集混用

某些场景下，用户埋点期望既用自动页面采集完成通用行为采集，又希望某些页面通过手动控制埋点时机完成精细化采集，然而如果自动页面浏览事件和手动页面浏览事件重复上报会引起用户行为数据的准确性，为此SDK通过了关闭单独某个页面自动采集的能力

5.4.1. API说明

TypeScript |

```
1  import { skipAutoTrack } from '@quicktracking/analytics';
2
3  function skipAutoTrack(context: PageContext)
```

5.4.2. 参数说明

参数	含义
context	页面环境上下文变量

5.4.3. 代码示例

见5.2.2代码示例

注意：！！！！

1. 为了保障页面事件编码的统计归一，当自动采集的页面在页面进入时开发者调用onPageStart，并指定了页面事件的事件编码pageName，此时自动的页面事件的事件编码也会变成onPageStart传入的pageName

2. 如果开发者开启页面事件自动采集功能，并且页面A没有调用skipAutoTrack，又通过代码埋点成对调用onPageStart、onPageEnd，此时当触发A页面离开时将同时上报 2 条页面浏览事件
3. onPageStart、onPageEnd 需要成对调用，否则影响页面事件采集的准确性

6. 自定义事件

6.1.1. API 说明

▼ TypeScript |

```
1  import { trackEvent } from '@quicktracking/analytics';
2
3  function trackEvent(eventID: string, params?: Record<string, string | number | string[]>)
```

6.1.2. 参数说明

参数	含义
eventID	自定义事件的事件编码

params	<p>自定义事件的事件属性，kv键值对集合</p> <p>k属性名，string类型，应为长度为128个字符以内的非空（包括undefined）字符串</p> <p>v属性值，string类型 或 number类型 或 字符串数组类型</p> <ul style="list-style-type: none"> 如果是字符串类型，应为长度为256个字符以内的字符串 如果是字符串数组类型，数组的长度不能超过100，含100 <p>注意：！！！！其中</p> <p>k="\$page_name"：为事件预制属性，用于声明当前自定义事件所归属页面的页面编码，可选参数，若传值，则长度为128位字符以内的字符串，且不能为空</p> <p>k="\$page_url"：为事件预制属性，用于声明当前自定义事件所归属页面的页面路径，可选参数，若传值，则长度为256位字符以内的字符串，且不能为空</p>
--------	---

6.1.3. 代码示例

- 在UI线程中埋点

TypeScript

```

1  import { trackEvent } from '@quicktracking/analytics';
2  Button('自定义事件').onClick(() => {
3    trackEvent("eventid", {
4      param1: "value"
5      param2: 2,
6      param3: ["productId1", "productId2"],
7      $page_name: "pageDemo", // 事件预制属性，可选参数
8      $page_url: "pages/pageDemo", // 事件预制属性，可选参数
9    });
10 });

```

- 在worker线程中埋点

```

1  import worker, { ThreadWorkerGlobalScope, MessageEvents, ErrorEvent } from
    '@ohos.worker';
2  import { trackEvent } from '@quicktracking/analytics';
3
4  const workerPort: ThreadWorkerGlobalScope = worker.workerPort;
5
6  workerPort.onmessage = function (e: MessageEvents) {
7      trackEvent("workerevent", {
8          param1: "value"
9          param2: 2,
10         param3: ["productId1", "productId2"],
11         $page_name: "pageDemo", // 事件预制属性, 可选参数
12         $page_url: "pages/pageDemo", // 事件预制属性, 可选参数
13     })
14 }
15
16 workerPort.onmessageerror = function (e: MessageEvents) {}
17
18 workerPort.onerror = function (e: ErrorEvent) {}

```

暂未提供c++中直接调用的方法，如需要c++层埋点，请参考鸿蒙Next官网c++与arkts互通的知识，通过NAPI来调用trackEvent方法实现埋点

7. 手动采集应用启动、退出、激活事件

SDK配置 `enableAutoTrackApplication = true` 默认开启，如果某些场景下不能满足启动、退出、激活事件的采集需求，可以通过手动调用API采集这三个场景行为。

注意：！！！！

使用手动采集启动、退出、激活事件不能与自动采集应用启动、退出、激活事件混用，否则影响数据统计准确性。即需要设置 `enableAutoTrackApplication = false`。

7.1. 手动上报应用激活事件

7.1.1. API 说明


```
1 import { trackAppInstall } from '@quicktracking/analytics';
2
3 function trackAppInstall(context: ApplicationContext, appKey: string, par
  ams: {
4     browser?: string,
5     $page_name?: string,
6     $page_url?: string,
7     [key: string]: string | number | string[]
8 } ) {
9     appTrack.manager.manualTrackAppInstall(context, appKey, params)
10 }
```

7.1.2. 参数说明

参数	含义
context	应用环境上下文变量
appKey	应用唯一标识appKey，入参的appkey一定要与QT后台保持一致

params	<p>应用激活事件的事件属性，kv键值对集合</p> <p>k属性名，string类型，应为长度为128个字符以内的非空（包括undefined）字符串</p> <p>v属性值，string类型 或 number类型 或 字符串数组类型</p> <ul style="list-style-type: none"> 如果是字符串类型，应为长度为256个字符以内的字符串 如果是字符串数组类型，数组的长度不能超过100，含100 <p>注意：！！！！其中</p> <p>k="browser"：为激活事件的预制属性，用于声明当前唤起应用的浏览器UserAgent信息，可选参数，默认为undefined</p> <p>k="utm_xxx"：为激活事件的预制属性，k以‘utm_’字符串开头，用于声明当前唤起应用的渠道参数信息，sdk会解析这类参数，并存储在全局属性中。</p> <p>k="\$page_name"：为激活事件的预制属性，用于声明当前激活事件所归属页面的页面编码，可选参数，默认为undefined。若传值，则长度为128位字符以内的字符串，且不能为空</p> <p>k="\$page_url"：为激活事件的预制属性，用于声明当前激活事件所归属页面的页面路径，可选参数，默认为undefined。若传值，则长度为256位字符以内的字符串，且不能为空</p>
--------	--

7.1.3. 代码示例

```

1  import UIAbility from '@ohos.app.ability.UIAbility';
2  import Want from '@ohos.app.ability.Want';
3  import window from '@ohos.window';
4  import { trackAppInstall } from '@quicktracking/analytics';
5
6  export default class EntryAbility extends UIAbility {
7
8      onCreate(want: Want, launchParam: AbilityConstant.LaunchParam): void {
9          trackAppInstall(
10             this.context.getApplicationContext(),
11             "您应用的appKey",
12             {
13                 $browser: 'Mozilla/5.0 Chrome/126.0.0.0 Safari/537.36', //可选参
数，默认为undefined
14                 $page_name: "home_page", //可选参数，默认为undefined
15                 $page_url: "pages/Index", //可选参数，默认为undefined
16                 utm_source: "hwbroser",
17                 man_app_p1: 111,
18                 man_app_p2: '222',
19                 man_app_p3: ['1', '2']
20             }
21         )
22     }
23 }

```

7.2. 手动上报应用启动事件

7.2.1. API 说明

```

1  import { trackAppStart } from '@quicktracking/analytics';
2
3  function trackAppStart(params: {
4      isFirstLaunch?: boolean,
5      $page_name?: string,
6      $page_url?: string,
7  }, customParams?: object )

```

7.2.2. 参数说明

参数	含义
params	<p>应用启动事件的预制属性，kv键值对集合。</p> <p>其中：</p> <p>k="isFirstLaunch"为冷热启动标识，默认值为false，代表热启动，如果想区分冷、热启动事件，需要开发者自己控制isFirstLaunch字段的传值</p> <p>k="\$page_name"：为激活事件的预制属性，用于声明当前激活事件所归属页面的页面编码，可选参数，默认为undefined。若传值，则长度为128位字符以内的字符串，且不能为空</p> <p>k="\$page_url"：为激活事件的预制属性，用于声明当前激活事件所归属页面的页面路径，可选参数，默认为undefined。若传值，则长度为256位字符以内的字符串，且不能为空</p>
customParams	<p>应用启动事件的事件属性，kv键值对集合</p> <p>k属性名，string类型，应为长度为128个字符以内的非空（包括undefined）字符串</p> <p>v属性值，string类型 或 number类型 或 字符串数组类型</p> <ul style="list-style-type: none"> 如果是字符串类型，应为长度为256个字符以内的字符串 如果是字符串数组类型，数组的长度不能超过100，含100

7.2.3. 代码示例

TypeScript

```
1  import UIAbility from '@ohos.app.ability.UIAbility';
2  import Want from '@ohos.app.ability.Want';
3  import window from '@ohos.window';
4  import { trackAppStart } from '@quicktracking/analytics';
5
6  export default class EntryAbility extends UIAbility {
7
8      onForeground(): void {
9
10         trackAppStart({
11             isFirstLaunch: true, //可选参数，默认为false
12             $page_name: "home_page", //可选参数，默认为undefined
13             $page_url: "pages/Index" //可选参数，默认为undefined
14         }, {
15             man_app_p1: 111,
16             man_app_p2: "222",
17             man_app_p3: ['1', '2']
18         })
19     }
20 }
21 }
```

7.3. 手动上报应用退出事件

7.3.1. API 说明

TypeScript

```
1  import { trackAppEnd } from '@quicktracking/analytics';
2
3  function trackAppEnd(params: {
4      $page_name?: string,
5      $page_url?: string,
6      duration?: number
7  }, customParams?: object)
```

7.3.2. 参数说明

参数	含义
----	----

params	<p>应用退出事件的预制属性kv键值对集合。</p> <p>其中：</p> <p>k="duration"：代表应用浏览时长，number类型，默认值为0</p> <p>k="\$page_name"：代表当前应用退出事件所归属页面的页面编码，可选参数，默认为undefined。若传值，则长度为128位字符以内的字符串，且不能为空</p> <p>k="\$page_url"：代表当前应用退出事件所归属页面的页面路径，可选参数，默认为undefined。若传值，则长度为256位字符以内的字符串，且不能为空</p>
customParams	<p>应用启动事件的事件属性，kv键值对集合</p> <p>k属性名，string类型，应为长度为128个字符以内的非空（包括undefined）字符串</p> <p>v属性值，string类型 或 number类型 或 字符串数组类型</p> <ul style="list-style-type: none"> • 如果是字符串类型，应为长度为256个字符以内的字符串 • 如果是字符串数组类型，数组的长度不能超过100，含100

7.3.3. 代码示例

```
1 import UIAbility from '@ohos.app.ability.UIAbility';
2 import Want from '@ohos.app.ability.Want';
3 import window from '@ohos.window';
4 import { trackAppEnd } from '@quicktracking/analytics';
5
6 export default class EntryAbility extends UIAbility {
7
8     onBackground(): void {
9
10         trackAppEnd({
11             $page_name: "home_page",
12             $page_url: "pages/Index",
13             duration: 666
14         }, {
15             man_app_p1: 111,
16             man_app_p2: "222",
17             man_app_p3: ['1', '2']
18         })
19     }
20 }
21 }
```

应用内H5数据上报

1. H5桥接使用场景

场景1：如何将H5数据同时上报至H5应用和APP应用

一个“春日”活动H5嵌入在多个App端，按照当前方式操作，可以满足下述分析诉求：

- App运营需要看到客户在App内参加H5活动前后的完整链路数据。
- H5活动运营需要看到H5在所有App端内的活动数据。

操作说明

1. 创建两个应用：

- 一个App应用，该App有自己的appKey_app
- 一个Web/H5应用，该Web/H5有自己的appKey_h5

2. SDK配置 **enableJSBridge=true** 开启，并在应用WebView页面注入自定义javaScriptProxy。

上报日志

如果此时在H5端触发一个埋点事件

上报客户端	日志内容
app应用	<ol style="list-style-type: none">1. appKey是app的“appKey_app”2. 用户账号为app的用户账号3. 用户属性为app的埋点上报的用户属性4. 设备ID为app的设备ID5. 系统属性为app的系统属性6. 全局属性是app内埋点的全局属性和H5设置的全局属性merge之后的结果7. 自定义事件的事件编码和事件属性均H5内触发的埋点内容
webview内H5应用	<ol style="list-style-type: none">1. appkey是H5的“appKey_h5”2. 用户账号为H5的用户账号3. 用户属性为H5的埋点上报的用户属性4. 设备ID为H5的设备ID5. 系统属性为H5的系统属性6. 全局属性是H5内埋点的全局属性7. 自定义事件的事件编码和事件属性均H5内触发的埋点内容

场景2：如何将H5数据仅上报至唯一APP应用中

由于技术架构，一部分页面为H5实现，按照当前方式操作，可以满足下述分析诉求：

- App业务、运营和PD等可以在App数据内看到完整的全链路数据。

操作说明

1. 创建一个App应用，仅有一个appKey_app
2. SDK配置 **enableJSBridge=true** 开启，并在应用WebView页面注入自定义javaScriptProxy，

详细示例见本章节2.1

3. H5 appKey设置为空，此时H5 SDK会打印警告日志，但不影响发数。并设置aplus-jsbridge-only等于true，关闭H5数据上报

上报日志

如果此时在H5端触发一个埋点事件

上报客户端	日志内容
app应用	<ol style="list-style-type: none">1. appKey是app的“appKey_app”2. 用户账号为app的用户账号3. 用户属性为app的埋点上报的用户属性4. 设备ID为app的设备ID5. 系统属性为app的系统属性6. 全局属性是app内埋点的全局属性和H5设置的全局属性merge之后的结果7. 自定义事件的事件编码和事件属性均H5内触发的埋点内容
webview内H5应用	无日志上报

场景3：如何将H5数据仅上报给H5应用中

App业务、运营和PD等不希望在App中看到H5的数据

操作说明

1. 创建两个应用：
 - 一个App应用，该App有自己的appKey_app
 - 一个Web/H5应用，该Web/H5有自己的appKey_h5
2. 不进行任何API的调用，app应用和h5应用分别埋点

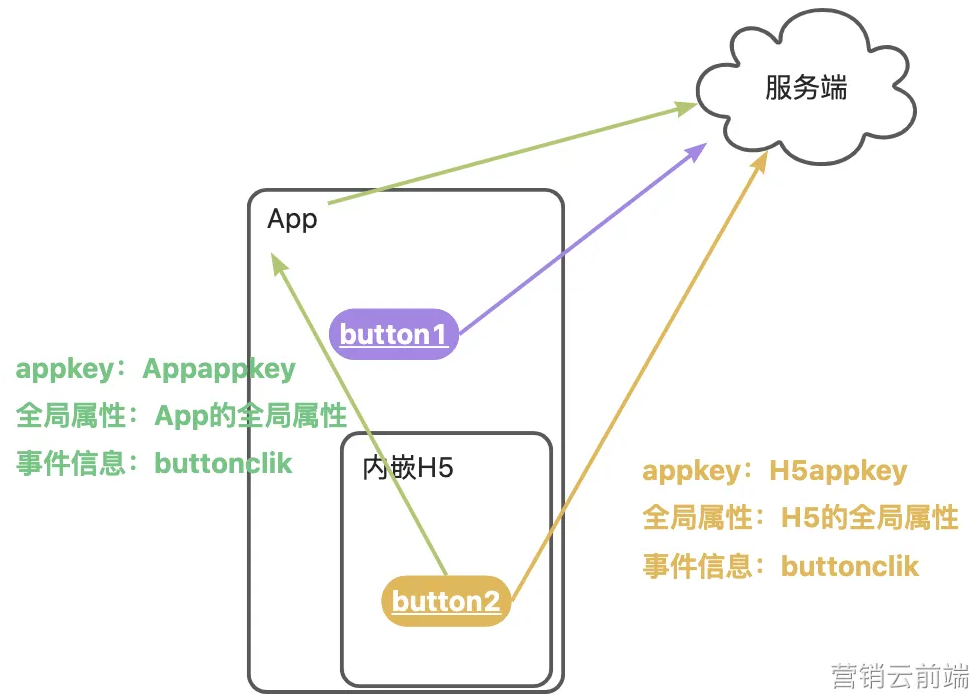
上报日志

如果此时在H5端触发一个埋点事件

上报客户端	日志内容
-------	------

app应用	无日志上报
webview内H5应用	<ol style="list-style-type: none"> 1. appkey是H5的“appKey_h5” 2. 用户账号为H5的用户账号 3. 用户属性为H5的埋点上报的用户属性 4. 设备ID为H5的设备ID 5. 系统属性为H5的系统属性 6. 全局属性是H5内埋点的全局属性 7. 自定义事件的事件编码和事件属性均H5内触发的埋点内容

2. H5桥接原理说明



上述场景1~3存在日志上报的前提是：

- 鸿蒙Next App集成了QuickTracking 鸿蒙Next SDK
- 应用内H5页面集成了QuickTracking H5 SDK

2.1. 鸿蒙Next 应用端代码

桥接API说明

需首先开启SDK配置 `enableJSBridge=true` ， 否则桥接API调用无效

TypeScript |

```
1  import { createJSBridgeProxy } from '@quicktracking/analytics';
2
3  function createJSBridgeProxy(controller: WebviewController)
```

参数说明

参数	含义
controller	鸿蒙Next系统webview控制器 WebviewController

代码示例

```

1  import { webview } from '@kit.ArkWeb'
2  import { BusinessError } from '@kit.BasicServicesKit'
3  import { promptAction } from '@kit.ArkUI'
4  import { createJSBridgeProxy } from '@quicktracking/analytics';
5
6  @Entry
7  @Component
8  struct WebPage {
9      @State message: string = 'Hello World';
10     controller: webview.WebviewController = new webview.WebviewController();
11     responseWeb: WebResourceResponse = new WebResourceResponse();
12
13     onPageShow(): void {
14         // 当需要引入其他用户自定义JSProxy
15         // const qtJSProxy:QTJSProxyObject = createJSBridgeProxy(this.controller);
16         // this.controller.registerJavaScriptProxy(qtJSProxy.object, qtJSProxy.name, qtJSProxy.methodList)
17         // this.controller.registerJavaScriptProxy(this.testObjtest,"objName",["test"],["asyncTestBool"]);
18     }
19
20     build() {
21         Column() {
22             Web({src: $rawfile('index.html'), controller: this.controller})
23                 .javascriptAccess(true)
24                 .javascriptProxy(createJSBridgeProxy(this.controller))
25         }
26         .height('100%')
27         .width('100%')
28     }
29 }

```

2.2. H5应用侧代码

仅通过APP通道上报，关闭H5通道上报日志API

```

1 aplus_queue.push({
2   action: 'aplus.setMetaInfo',
3   arguments: ['aplus-jsbridge-only', true]
4 });

```

注意：！！！！ H5 设置 `aplus-jsbridge-only` 为 `true`，则为禁止H5 SDK上报埋点事件，务必检查非应用内H5页面不要设置

H5 全局属性全局生效接口

在H5中设置开启下述接口后，H5中通过`aplus.appendMetaInfo`中`globalproperty`或`aplus.setMetaInfo`中`globalproperty`设置的全局属性将会同步生效至App原生中。（该接口默认为`false`）

注意：！！！！ 该API在`qt_web_v2.0.12`及以后版本开始支持

```

1 aplus_queue.push({
2   action: 'aplus.setMetaInfo',
3   arguments: ['aplus-globalproperty-sync-enable', true]
4 });

```

例如：在App中设置全局属性「当前所在城市：北京」，在App内H5中设置全局属性「当前所在城市：上海」，则在H5和App原生中触发的事件的全局属性都将为「当前所在城市：上海」。如果不开启该开关，则H5中设置的全局属性不会再App原生中生效。

预制事件和预制属性

预制事件

事件名称	事件编码	属性	属性含义	实现方式	采集时间
应用启动事件	\$\$_app_start	启动时间	启动事件发生时的客户端时间	<ul style="list-style-type: none"> 自动采集，开启SDK配置<code>enableAutoTrackA</code> 	app启动时
		启动类型	冷启动/热启动		

		启动页面	启动时所打开的页面	<p>pplication = true</p> <ul style="list-style-type: none"> 手动采集，调用 trackApp Start 	
		埋点方式	<ul style="list-style-type: none"> 0，手动埋点方式 1，自动埋点方式 		
应用退出事件	\$\$_app_end	退出时间	退出事件发生时的客户端时间	<ul style="list-style-type: none"> 自动采集，开启 SDK配置 enableAutoTrackApplication = true 手动采集，调用 trackApp End 	app退出时
		使用时长	从本次启动到退出的app使用时长		
		退出页面	退出时所在的页面		
		埋点方式	<ul style="list-style-type: none"> 0，手动埋点方式 1，自动埋点方式 		
页面浏览事件	\$\$_page_end	页面退出时间	页面退出时的客户端时间	<ul style="list-style-type: none"> 自动采集，开启 SDK配置 enableAutoTrackPage = true 手动采集，成对调用 onPageStart、onPageEnd 	页面离开时上报
		页面编码	<ul style="list-style-type: none"> 自动采集时，见5.1节说明 手动采集为API设置的 pageName值 		

		页面路径	<ul style="list-style-type: none"> 自动采集时，见5.1节说明 手动采集为API获取的url值 		
		页面进入时间	页面进入时的客户端时间		
		页面停留时间	页面退出时间与进入时间的时间差		
		来源页面编码	上一个页面编码		
		埋点方式	<ul style="list-style-type: none"> 0，手动埋点方式 1，自动埋点方式 		
应用激活事件	\$\$_app_install	utm属性	广告来源、广告类型等，开发者解析唤起参数主动设置给sdk的全局属性中	<ul style="list-style-type: none"> 自动采集，开启SDK配置enableAutoTrackApplication = true 手动采集，调用trackAppInstall 	app被下载后的第一次启动时上报
		browser	唤起应用的浏览器信息		

预制属性

属性字段	采集时间
------	------

应用标识	SDK初始化
应用发布渠道	SDK初始化
设备品牌	SDK初始化
设备型号	SDK初始化
操作系统	SDK初始化
操作系统版本	SDK初始化
网络访问类型	
分辨率	SDK初始化
系统语言	SDK初始化
cpu架构	SDK初始化
应用包名	SDK初始化
国家	事件上报到服务端时
省	事件上报到服务端时
市	事件上报到服务端时
应用版本	SDK初始化
SDK版本	SDK初始化
SDK类型	SDK初始化
平台	SDK初始化
平台类型	SDK初始化
渠道属性	上传渠道属性时
设备ID	SDK初始化
账号ID	调用用户账号设置接口
事件编码	触发事件时
事件触发客户端时间	触发事件时
事件触发服务端时间	事件上报到服务端时

事件触发所在页面编码	触发事件时
事件触发所在页面标题	触发事件时
事件触发所在页面路径	触发事件时
事件触发来源页面编码	触发事件时
事件触发来源页面路径	触发事件时
事件埋点方式	触发事件时