TingYun_App_iOS_SDK_接口说明-V3.0

> 本文档介绍了 TingYun_SDK 基于请求数据、用户体验数据、异常数据、自定义业务和其他数据相关接口的使用及说明

一、请求数据相关接口

1、自定义请求头

> TingYun_SDK 支持配置采集「请求头」参数，用以进行「后端调用链追踪」。

1、相关接口

```
/*
 @key 设置为要采集的请求头中的headerkey值
 */
+(void)setRequestIDHeaderKey:(NSString *)key;
```

2、代码示例

```
int main(int argc, char * argv[]) {
 @autoreleasepool {
      //调用位置建议在初始化，其他位置首次不采集
      [NBSAppAgent startWithAppID:@"appkey" location:YES];
      [NBSAppAgent setRequestIDHeaderKey:@"key"];
      ...
    }
}
```

2、SDK传输数据使用国密加密

> SDK传输数据支持国密加密，调用该接口开启国密加密，默认不开启「前提:需工程中导入NBSGMKit.framework」
>
> 注:该接口需要在SDK初始化时调用

1、相关接口

```
/*
 @need 传入YES，则SDK传输数据使用国密加密。
 */
+ (void)encryptionRequired:(BOOL)need;
```

2、代码示例

```
int main(int argc, char * argv[]) {
    @autoreleasepool {
        //开启国密加密
        [NBSAppAgent encryptionRequired:YES];
        //默认为https，如果redirect_host为http协议的话则需要调用该接口。
        [NBSAppAgent setHttpEnabled:YES];
        //redirect_host为平台服务器地址
        [NBSAppAgent setRedirectURL:@"redirect_host"];
        //Your_appkey从平台中获取
        [NBSAppAgent startWithAppID:@"Your_appkey"];
        return UIApplicationMain(argc, argv, nil,
NSStringFromClass([AppDelegate class]));
    }
}
```

3、支持采集libcurl请求数据

> 获取libcurl请求性能数据，如：IP、DNS、TCP、SSL等性能指标。

1、相关接口

```
/**
 for record curl networl.
 */
void nbsGetCurlNetworkInfo(void *curl,int curlcode,void *ptr);
```

2、代码示例

```
- (void)libcurlNetwork
{
    ...
    CURL *curl = curl_easy_init();
    CURLcode res;
    if (curl) {
        curl_easy_setopt(curl, CURLOPT_URL, "http://www.baidu.com");
        res = curl_easy_perform(curl);
        nbsGetCurlNetworkInfo(curl, res, &curl_easy_getinfo);
        curl_easy_cleanup(curl);
    }
    ...
}
```

二、用户体验数据相关接口

1、自定义冷启动耗时

> TingYun_SDK 默认计算 SDK 初始化开始至第一个页面加载结束的时间为「冷启动耗时」，研发人员可以根据自身应用需求更改计算「冷启动耗时」的结束点。

- 开启控制开关

1、相关接口

```
/*
 @enable 传入YES，开启设置自定义启动结束功能
*/
+ (void)customLanuchEnd:(BOOL)enable;
```

2、代码示例

```
int main(int argc, char * argv[]) {
 @autoreleasepool {
    //设置开启自定义结束点功能，在启动SDK之前调用。
    [NBSAppAgent customLanuchEnd:YES];
    [NBSAppAgent startWithAppID:@"appkey" location:YES];
    ...
    }
}
```

- 设置冷启动耗时的结束点

> 该接口需要与「customLanuchEnd」配合使用，当设置「customLanuchEnd」接口为 YES 时，「lanuchFinish」接口设置生效

1、相关接口

```
/*
 自定义结束时间点，在启动结束时调用。
*/
+ (void)lanuchFinish:(NSString *)lanuchName;
```

2、代码示例

```
-(void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    [NBSAppAgent lanuchFinish:@"firstVC"];
}
```

2、自定义埋点

> 由于 TingYun_SDK 默认关注系统类和方法，无法关注「业务代码」的耗时情况，使用「自定义埋点」接口可以补全「页面体验分析」和「操作体验分析」模块中的【分解图】，能够帮助开发者清晰的了解其业务代码的耗时及调用情况。

1、相关接口

> 注意：「自定义埋点」接口需要成对调用，请勿跨方法、跨进程以及在异步加载和递归调用中使用该接口。

```
//@String Name为当前方法所在方法名或自定义名称，支持中文、英文、数字、下划线，但不能包含空格或其他的转义字符
beginTracer(@"String Name")
endTracer(@"String Name")
```

2、代码示例

```
- (void)doSomething
{
  //用户可以在SDK初始化后的任意方法前后添加自定义Trace
  beginTracer(@"String Name")
  //write you code here
  endTracer(@"String  Name")
}
```

3、自定义操作

> 通过「自定义操作」来定义一个【业务操作】用以了解其性能表现情况

1、相关接口

> 注意：「自定义操作」接口需要成对调用，actionName不可为空，支持跨方法、跨线程调用

```
+ (void)customActionStart:(NSString *)actionName;
+ (void)customActionEnd:(NSString *)actionName withTag:(NSString *)tag
withCustomInfo:(NSDictionary *)customInfo;
```

2、代码示例

```
- (void)doSomething
{
    [NBSAppAgent customActionStart:@"doSomething"];
        ...
    NSDictionary *cust = @{@"key":@"value"};
    [NBSAppAgent customActionEnd:@"doSomething" withTag:@"tag"
withCustomInfo:cust];
}
```

4、自定义页面结束

> 通过「自定义页面结束」来定义一个【页面实际加载时间】用以了解其性能表现情况

1、相关接口

> 注意：「自定义页面结束」接口需要成对调用，pageName不可为空，支持跨方法、跨线程调用，App启动过程中的视图控制器不支持自定义；需要将 customPageLoad: 返回的结果 customId 传入到 customPageLoadFinish:withPageName: 中来关联结束。
>
> 在待自定义视图控制器 viewDidLoad 中调用 customPageLoad: ；在待自定义视图控制器 viewDidAppear 之后调用customPageLoadFinish:withPageName: 。

```
/**
 *是否自定义页面结束点
 *@param enable 是否自定义.
 *@result identifier 关联结束。
 */
+ (NSInteger)customPageLoad:(BOOL)enable;

/**
 *自定义页面结束。
 *@param customId 传入customPageLoad返回的id。
 *@param pageName 页面名称
 */
+ (void)customPageLoadFinish:(NSInteger)customId withPageName:(NSString
*)pageName;
```

2、代码示例

```
- (void)viewDidLoad {
    [super viewDidLoad];
    customId = [NBSAppAgent customPageLoad:YES];
    ...
}

- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    ....
    dispatch_async(dispatch_get_main_queue(), ^{
        [NBSAppAgent customPageLoadFinish:customId withPageName:@"CustomPage"];
    });
}
```

三、异常数据相关接口

1、设置面包屑

> 研发人员可以在应用程序的任意位置调用「面包屑」接口进行埋点。当应用程序发生崩溃时，SDK会按代码的触发顺序收集埋点信息并在崩溃轨迹中高亮显示，以协助研发人员在应用崩溃时了解代码调用逻辑。

1、相关接口

```
/*
  @breadcrumb:自定义信息,最多包含100个字符，支持中文、英文、数字、下划线
*/
+(void)leaveBreadcrumb:(NSString *)breadcrumb;
```

2、代码示例

```
-(BOOL)application:(UIApplication *)application didFinishLaunchingWithOption:
(NSDictionary *)launchOptions
{
  //设置面包屑
  [NBSAppAgent leaveBreadcrumb:@"didFinishLaunchingWithOptions"];
  return YES;
}
```

2、自定义崩溃附加信息

> 在应用发生崩溃的时候，研发人员往往需要更多的信息以收集现场环境，可以通过调用「自定义崩溃附加信息」接口上传额外信息，协助分析崩溃问题。

1、相关接口

```
/*
 此函数可以在任何位置多次调用，最多可添加10条附加信息，每条附加信息最大支持100个字节，随崩溃上传。
 */
+(void)setCustomerData:(NSString*)data forKey:(NSString*)key;
```

2、代码示例

```
- (void)doSomething
{
    [NBSAppAgent setCustomerData:@"value" forKey:@"key"];
    ...
}
```

3、自定义错误

> 使用「自定义错误」接口可以采集研发人员「try / catch 异常」和「业务错误」并在听云平台「异常分析」→「错误」中进行展示，可以帮助研发人员收集异常和错误。

1、相关接口

```
/*
 参数说明：
 @message：不可以传空，最大长度1024字节，超出截取前1024字节
 @exception：上传exception取到抛出时的堆栈，不传只取到调用接口的堆栈，多线程下只取调用接口的线程的堆栈，堆栈最大深度为100
 @metaData：value值支持 NSNumber,NSString,NSArray，NSDictionary类型，最大128k，超出设置metadata为："metaData":{"error":"metaData的大小大于最大限制128K"}
 */
+ (void)reportError:(NSString *)message withMetaData:(NSDictionary *)metaData;
+ (void)reportError:(NSString *)message withException:(NSException *)exception withMetaData:(NSDictionary *)metaData;
```

2、代码示例

```
- (void)doSomething
{
    ...
    if(success){
        ...
    }else{
        NSDictionary *dict = @{@"k1":@"v1",@"K2":@"V2"};
        [NBSAppAgent reportError:@"errorMsg" withMetaData:dict];
    }
    ...
}
- (void)doSomething
{
    @try {
         // ... do something that might throw an exception ...
    }
    @catch (NSException *exception) {
        // report the error
        NSDictionary *dict = @{@"k1":@"v1",@"K2":@"V2"};
        [NBSAppAgent reportError:@"errorMsg" withException:exception withMetaData:dict];
    }
    @finally {
    }
    ...
}
```

4、设置崩溃类型采集

> TingYun_SDK 默认采集所有类型的信号错误，若无需采集某种 signal 类型的信号错误可调用此接口进行忽略，支持忽略单个类型和多个类型。

1 、相关接口

```
/*
 SDK支持采集的信号
 SIGABRT, //6
 SIGBUS,  //10
 SIGFPE,  //8
 SIGILL,  //4
 SIGSEGV, //11
 SIGSYS,  //12
 SIGTRAP, //5
 @ignore  参数支持字符串、NSNumber、NSArray
 */
+ (void)ignoreSomeSignalCrash:(id)ignore;
```

2 、代码示例

```
int main(int argc, char * argv[]) {
  @autoreleasepool {
      //初始化之前调用
      [NBSAppAgent ignoreSomeSignalCrash:@"SIGABRT"];
      [NBSAppAgent startWithAppID:@"appkey" location:YES];
      ...
  }
}
```

5、设置异常回调

> 使用「异常回调」接口后可以在发生卡顿或上传崩溃数据时获取SDK采集的异常数据「崩溃、卡顿数据」。
>
> 注：需要在崩溃上传或发生卡顿前调用「SDK连服务器成功后就会上传崩溃数据」

1 、相关接口

```
/**
 @brief 当发生crash或者卡顿时设置自定义回调
 @type 值为NBSCallBackOnCrash或NBSCallBackOnANR
 */
+ (void)setCallBack:(NBSCallBack)callback withType:(NBSCallBackType)type;
```

2 、代码示例

```
- (void)getExceptionInfo
{
    // context 为SDK采集的崩溃或卡顿的上下文信息
    [NBSAppAgent  setCallBack:^(NSDictionary *context) {
        ...
    } withType:NBSCallBackOnCrash];
}
```

四、JS自定义业务相关接口

1、JS自定义错误

> 使用「自定义错误」接口可以采集研发人员「try / catch 异常」和「业务错误」并在听云平台「异常分析」→「错误」中进行展示，可以帮助研发人员收集异常和错误。

1、相关接口

```
//您需要将以下代码，保存为tingyun@app-fix.js,添加到项目中

if(!window['NBSAppAgent']){
    function nbs_callMethod(functionName, args) {
        var wrap = {'method':functionName,'params':args};
        var info = JSON.stringify(wrap);
        if(typeof nbsJsBridge != 'undefined') {
            nbsJsBridge.parseArguments(info);
        }else if(typeof window.webkit != 'undefined'){
            if (!window.webkit.messageHandlers['nbsJsBridge']) return;
            window.webkit.messageHandlers['nbsJsBridge'].postMessage(info);
        }
    }

    var NBSAppAgent = {};
 /*
 自定义错误:
 message 长度最大1024字节
 metaData的value值支持 NSNumber,NSString,NSArray，NSDictionary类型，最大128k。
 */
    NBSAppAgent.reportError = function(message, metaData, exception) {
        if(!exception)
            return;
        if(!message)
            message = '';
        if(!metaData)
            metaData = {};
```

```
        var error =
{message:exception.message,line:exception.line,column:exception.column,sourceURL:exception.sourceURL,stack:exception.stack,type:exception.name}
        nbs_callMethod('reportError', { 'message': message, 'exception': error, 'metaData': metaData });
    };
    window['NBSAppAgent'] = NBSAppAgent;
}
```

2、代码示例

```
<script src="tingyun@app-fix.js"></script>
function jsCustomError() {
    try{console.log(abc)}
    catch(e){
         NBSAppAgent.reportError("JS-Customerror-Webview-jsMessage",{"metaDataKey":"metaDataValue"},e);
    }
}
```

2、JS自定义事件

> 自定义事件可以统计 App 中的任意事件，开发者在 SDK 初始化后的任意位置调用「自定义事件接口并设置对应上传参数」。
>
> 如：真实用户操作时点击某个功能按钮或触发了某个功能事件等

1、相关接口

```
//您需要将以下代码，保存为tingyun@app-fix.js,添加到项目中
if(!window['NBSAppAgent']){
    function nbs_callMethod(functionName, args) {
        var wrap = {'method':functionName,'params':args};
        var info = JSON.stringify(wrap);
        if(typeof nbsJsBridge != 'undefined') {
            nbsJsBridge.parseArguments(info);
        }else if(typeof window.webkit != 'undefined'){
            if (!window.webkit.messageHandlers['nbsJsBridge']) return;
            window.webkit.messageHandlers['nbsJsBridge'].postMessage(info);
        }
    }

    /*
 自定义事件:
 eventID 最多包含32个字符，支持中文、英文、数字、下划线，但不能包含空格或其他的转义字符
 eventTag 事件标签
 eventProperties
     其它附加属性，字典，超过30个键值对无效，可以为nil。
     key 不能包含中文，支持大小写英文字母下横线，数字，且必须以英文字母开头。最大长度：64
```

```
     Value值仅支持字符串（String）和数字（Number）类型
  */
    var NBSAppAgent = {};
    NBSAppAgent.onEvent = function(eventID, eventTag, eventProperties) {
        if(!eventID)
            return;
        if(!eventTag)
            eventTag = '';
        if(!eventProperties)
            eventProperties = {};
        nbs_callMethod('onEvent', { 'eventID': eventID, 'eventTag': eventTag,
'eventProperties': eventProperties });
    };
    window['NBSAppAgent'] = NBSAppAgent;
}
```

2、代码示例

```
<script src="tingyun@app-fix.js"></script>
function jsCustomEvent() {
    NBSAppAgent.onEvent("JS-eventid","eventTag",{"key":"value"});
}
```

五、其他数据相关接口

1、获取用户标识

> 通过添加「用户标识」可在听云报表平台通过该标识检索到具体用户的性能问题

1、相关接口

```
/*
 设置用户标识符,不能超过64个字符，可以在任意位置多次调用（值覆盖）
 @userId:唯一标识一个用户的信息
 */
+ (void)setUserIdentifier:(NSString *)userId;
```

2、代码示例

```
- (void)viewDidLoad
{
  [super viewDidLoad];
  //用户标识可为邮箱、手机号等能够标识用户身份的信息，如xxx@tingyun.com
  [NBSAppAgent setUserIdentifier:@"userId"];
}
```

2、版本更新提示开关（仅 SaaS 平台支持）

> TingYun_App_SDK 默认会校验当前使用的 SDK 版本是否为最新版本，新版 SDK 上线后，App 运行时会在控制台输出版本更新提示的 Log 日志，如不需要「版本更新提示」调用此接口关闭即可。

1、相关接口

```
/*
 关闭更新提示log
 @SDKVersion 为最新的SDK版本
 */
+ (void)closeLogForUpdateHint:(NSString *)SDKVersion;
```

2、代码示例

```
int main(int argc, char * argv[]) {
 @autoreleasepool {
    //初始化之前调用
    [NBSAppAgent closeLogForUpdateHint:@"sdkversion"];
    [NBSAppAgent startWithAppID:@"appkey" location:YES];
    ...
    }
}
```

3、自定义版本号

> TingYun_SDK 默认使用应用的「CFBundleShortVersionString」作为版本号上传，如需自定义版本号可以在初始化 SDK 时调用该接口进行配置。

1、相关接口

```
/**
 @brief 设置自定义App版本号，最多包含64个字符，支持中文、英文、数字、下划线
 */
+ (void)setVersionName:(NSString *)versionName;
```

2、代码示例

```
int main(int argc, char * argv[]) {
 @autoreleasepool {
    [NBSAppAgent startWithAppID:@"appkey" location:YES];
    [NBSAppAgent setVersionName:@"1.0.0"];
    ...
    }
}
```

4、获取听云设备 ID

> 应用首次启动时，听云服务器会下发 deviceId 用以标识设备。用户可以通过接口获取听云的 deviceId 值。

1、相关接口

```
/**
 @return 返回听云平台生成的设备id，该id为服务端返回，首次获取可能为空
 */
+ (NSString *)getTingyunDeviceId;
```

2、代码示例

```
- (void)getTyDeviceId
{
  NSString *tyDid = [NBSAppAgent getTingyunDeviceId];
  ...
}
```

5、设置地理位置信息

> 可通过调用接口设置经纬度以便精准获取设备地理位置信息。

1、相关接口

```
/**
 @brief 设置经纬度。
 */
+ (void)setLatitude:(double)latitude longitude:(double)longitude;
```

2、代码示例

```
- (void)doSomething
{
    ...
    [NBSAppAgent setLatitude:latitude  longitude:longitude];
    ...
}
```