**BFC行为采集系统SDK**

BFC小组 制定

2014-09-15

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **发布日期** | **主要内容** | **负责人** |
| V1.0 | 2014.9.15 | 用户行为采集SDK第一个版本 | 谢钰铭 |
|  | 整合 | 核对SDK文档与实际接口 | 谢钰铭 |
|  | 修定 | 发布Component组件行为采集器 | 谢钰铭 |

目录

[1、版本信息 4](#__RefHeading__788_421321843)

[2、适用机型 4](#__RefHeading__790_421321843)

[3、依赖环境 4](#__RefHeading__792_421321843)

[4、原理简介 4](#__RefHeading__794_421321843)

[5、具体方法 4](#__RefHeading__796_421321843)

[(1)载入方法 4](#__RefHeading__798_421321843)

[(2)操作的采集调用 7](#__RefHeading__800_421321843)

[(三) 异常采集 8](#__RefHeading__802_421321843)

[附录 9](#__RefHeading__804_421321843)

1. 版本信息

版本号：V1.00

1. 适用机型

Android通用机型，包括第三方android手机平台

1. 依赖环境

* 需要将BFC\_ Component \_BehaviorCollector\_VX.XX.jar 放在工程libs目录下；
* 需要将BFC\_ Component \_UploadManager\_VX.XX.jar 放在工程libs目录下；
* 需要将gson-X.X.X.jar 放在工程的libs目录中；
* 需要将BFC\_Utils.jar 放在工程libs目录中；
* 需要将uploadserverentity.jar放在工程libs目录下；
* 需要将httpmime-4.1.3.jar 放在工程的libs目录中；

1. 原理简介

* BFC基础组件中上传服务提供了上传功能；行为采集服务管理插入的操作记录，包括系统级别和APP级别，负责判断是否上传的条件；
* BFC\_ Component \_BehaviorCollector\_VX.XX.jar 封装了APP级的事件类型，提供了操作记录插入行为采集服务对应数据库的接口；以继承Application的或在Application中实现指定接口的方式封装了采集流程，简化了APP的调用方法。

1. 具体方法
2. 载入方法

1.继承BFC\_ Component \_BehaviorCollector\_VX.XX.jar 提供的 BCApplication 类；

第一种载入行为采集功能的方式是直接继承功能包提供的Application类，实现必要的方法，在manifest中使用自定义的Application：

public class DemoApplication extends BCApplication{

@Override

public boolean onCrashNeedUIReport() {

return true;

}

@Override

public boolean onNeedCrashExeception() {

return true;

}

@Override

public boolean onNeedToast() {

return true;

}

/\*\*

\* 不同的用户系统,只要按照这个类传值就行

\*/

@Override

public UserAttr getUserAttr() {

/\*\*

\* 这里最好从目前数据库读信息出来,不要用一个变量,防止用户信息期间有改变

\*/

return new UserAttr("ＩＤ", "姓名", "性别", "年龄", "学校", "年级", "年级类型", "科目");

}

}

2.使用在Application中重载onCreate方法，加入DemoApplication2 .initApplication(Application app, IAppConfig config , IGetter getter) 的方式：

如果您的应用需要继承其他功能或模块提供的Application，由于java的单继承特性，采用这种方法来实现：

public class DemoApplication2 extends Application{

@Override

public void onCreate() {

super.onCreate();

BCApplication.initApplication(this, new IAppConfig() {

@Override

public boolean onNeedToast() {

return true;

}

@Override

public boolean onNeedCrashExeception() {

return true;

}

@Override

public boolean onCrashNeedUIReport() {

return true;

}

}, new IGetter() {

@Override

public UserAttr getUserAttr() {

return new UserAttr("ＩＤ", "姓名", "性别", "年龄", "学校", "年级", "年级类型", "科目");

}

@Override

public MachineAttr getMachineAttr() {

return BCApplication.makeMachinceAttr(DemoApplication2.this);

}

@Override

public ApplicationAttr getApplicationAttr() throws Exception {

return BCApplication.makeApplicationAttr(DemoApplication2.this);

}

});

}

}

1. 操作的采集调用

环境设置好之后，还需要在行为的触发点调用相应的接口，来保证操作能够插入行为采集系统的数据库。在相应位置调用对应接口即可。下面是点击事件示例:

/\*\*

\* 点击事件

\* @param view

\*/

public void onClickEvent(View view){

BehaviorCollector.getInstance().clickEvent(

new EventAttr(EType.CLICK)

.setActivity(this.getClass().getName())

.setEventName("点击测试")

.setFunctionName("测试")

.setModuleDetail("没有"), null

);

}

* 1. (三) 异常采集

异常采集实际上是在应用发生崩溃时，采集系统将崩溃信息作为行为记录插入数据库的一种功能，是否采用该功能由（一）中提到Application设置中的接口决定：

@Override

public boolean onNeedCrashExeception() {

return true;

}

若该接口返回true，系统会自动采集异常信息。可以在服务器上取到相关的操作采集后一并分析处理。

附录

* 1. APP事件表
  2. BehaviorCollector 接口

public interface ICollector {

/\*\*

\* @param preActivityName

\* 进入的activity

\* @param curActivityName

\* 当前的activity

\*

\* \*/

void activityIn(

String preActivityName,

String curActivityName);

/\*\*

\* @param curActivityName

\* 当前的activity

\* @param duaring

\* 本次在该 activity 的停留时间

\*

\* \*/

void activityOut(

String curActivityName,

long duaring);

/\*\*

\* @param curActivityName

\* 进入的activity

\* @param duaring

\* 本次在该 activity 停留时间

\*

\* \*/

void appExit(

String curActivityName,

long duaring);

/\*\*

\* @param tValue

\* 进入的时间

\* @param curActivityName

\* 进入的activity

\* \*/

void appLaunch(

String tValue,

String curActivityName);

/\*\*

\* 点击事件(次数)

\*

\* @param functionName

\* 功能名称

\* @param context

\* Context

\*

\* \*/

void clickEvent(

String functionName,

Context context);

/\*\*

\* 自定义事件(次数)

\*

\* @param eventName

\* 事件名称

\* @param functionName

\* 功能名称

\* @param curActivityName

\* 当前activity

\*

\* \*/

void customEvent(

String eventName,

String functionName,

String curActivityName);

/\*\*

\* 记录功能点开始（时长 + 次数）,begin 与 end 接口要配对使用

\*

\* @param functionName

\* 功能名称

\* @param curActivityName

\* 当前界面名称

\*

\* \*/

void recordFunctionBegin(

String functionName ,

String curActivityName);

/\*\*

\* 记录功能点结束（时长 + 次数）

\*

\* @param functionName

\* 功能名称

\* @param curActivityName

\* 当前界面名称

\*

\* \*/

void recordFunctionEnd(

String functionName ,

String curActivityName );

/\*\*

\* 搜索事件（次数 + 内容）

\*

\* @param functionName

\* 功能名称

\* @param content

\* 搜索的内容

\* @param context

\* Context

\*

\* \*/

void searchEvent(

String functionName ,

String content,

Context context);

/\*\*

\*

\* @param eventAttr

\* 事件属性

\* @param dataAttr

\* 数据属性

\*/

void exceptionEvent(

EventAttr eventAttr,

DataAttr dataAttr);

}