高级功能

用户自定义ID

用户自定义ID为当前用户设置唯一标示码，在任意位置均可设置UserID。

1、相关接口。

```
//UserID最多包含64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
NBSAppAgent.setUserIdentifier("userIdentifier");
```

2、代码示例。

```
public class MainActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String userIdentifier = getUserID();
        NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(true)
        .start(this.getApplicationContext());
        NBSAppAgent.setUserIdentifier(userIdentifier);
      }
}
```

面包屑

面包屑能够更好的协助用户调查崩溃发生的原因，可以知晓用户发生崩溃之前的代码逻辑与崩溃轨迹结合使用能够更好的复现用户崩溃场景。

1、相关接口。

```
//最多包含100个字符，支持中文、英文、数字、下划线
NBSAppAgent.leaveBreadcrumb("keyPressed”);
NBSAppAgent.leaveBreadcrumb("loginDone");
```

2、代码示例。

```
public MyActivity extends Activity {
         public void onCreate(Bundle savedInstanceState) {
             super.onCreate(savedInstanceState);
             NBSAppAgent.setLicenseKey("AppKey")

.withLocationServiceEnabled(true).start(this.getApplicationContext());
             NBSAppAgent.leaveBreadcrumb("login MyActivity onCreate");
         }
         public void onResume() {
             super.onResume();
             NBSAppAgent.leaveBreadcrumb("login MyActivity onResume");
         }
         public void logginPressed(View view) {
             NBSAppAgent.leaveBreadcrumb("login MyActivity logginPressed");
             new LoginAsyncTask.execute();
         }
```

```
        public void onStop() {
            super.onStop();
            NBSAppAgent.leaveBreadcrumb("login MyActivity onStop");
        }
}
```

自定义Event

自定义事件用于统计App中的任意事件，开发者可以在SDK初始化后的任意位置添加自定义事件，并设置对应上传参数。如：真实用户操作时候点击某个功能按钮或触发了某个功能事件等。

1、相关接口。

```
//EVENT_ID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
NBSAppAgent.onEvent(String EVENT_ID);
```

2、代码示例。

```
@Override
public void onClick(View v) {
    ……
        NBSAppAgent.onEvent("添加购物车");
      ……
}
```

自定义Trace

听云SDK默认采集系统类和方法的性能数据，无法采集开发者自定义类和方法的性能数据。使用“自定义Trace”接口就可以帮助开发者时刻了解所写代码的健壮性及其性能数据。如：开发者想要了解某个自定义方法的初始化耗时及性能消耗情况，就可以在该自定义方法前后添加“自定义Trace”接口即可。

> **注意：**由于自定义Trace是成对出现的，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。

1、相关接口。

```
//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
NBSAppAgent.beginTracer("String Name");
NBSAppAgent.endTracer("String Name");
```

2、代码示例。

```
//用户可以在SDK初始化后的任意方法前后添加自定义Trace
public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
       init();
    }

private void init() {
     //方法开始前添加beginTracer
     NBSAppAgent.beginTracer(“这是Init方法”);
        try {
            ……
```

```
        } catch (NameNotFoundException e) {
            e.printStackTrace();
        }
     //方法结束后添加endTracer
     NBSAppAgent.endTracer("这是Init方法");
}
```

自定义Log

开发者可通过Android日志系统的LogCat，来收集和查看系统调试输出的信息，通过打印输出的Log信息来调查Bug发生时的应用程序信息，并通过听云SDK上传自定义Log日志。

> **注意：**收集LogCat信息,需要使用READ_LOGS权限，请将下列代码添加到应用程序 的AndroidManifest.xml文件中，默认收集50行，最多收集100行日志。
>
> < uses-permission android:name="android.permission.READ_LOGS" >

1、控制开关。

```
NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(true).enableLoggi
ng(true).start(this.getApplicationContext());
```

2、相关接口。

```
NBSAppAgent.setLogging(int lineNumber);
NBSAppAgent.setLogging(String filter);
NBSAppAgent.setLogging(int lineNumber,String filter);
```

3、代码示例。

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
      // Enable logging
      NBSAppAgent.setLicenseKey("AppKey")
      .withLocationServiceEnabled(true).enableLogging(true)
      .start(this.getApplicationContext());
          // Log last 100 messages
          NBSAppAgent.setLogging(100);
}
```

自定义附加信息

用户可以在初始化之后任意位置配置该接口，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。

1、相关接口。

```
NBSAppAgent.setUserCrashMessage(String key,String value);
```

2、代码示例。

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     NBSAppAgent.setLicenseKey("AppKey")
     .withLocationServiceEnabled(true).start(this.getApplicationContext());
      //初始化后的任意位置插入自定义附加信息
      NBSAppAgent.setUserCrashMessage(“张三”,“13700001234”);}
```

控制台版本提示

用户调试时会在控制台输出版本更新提示，当开启调试模式且当前使用SDK版本低于线上最新版本时，App运行后打印输出。

> **说明**：项目工程只有debug模式才会开启此功能。
>
> 可通过在AndroidMainfest.xml文件中的application标签添加debuggable属性控制是否开启调试模式。

1、相关接口。

```
NBSAppAgent.closeLogForUpdateHint();
```

2、代码示例。

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(true)
     .closeLogForUpdateHint().start(this.getApplicationContext());
     //初始化时调用接口关闭版本提示功能
}
```

页面加载埋点

在页面加载分析模块，SDK默认以IdleHandler的queueIdle()回调时间点作为页面内容加载完成的时间点。用户可以通过setPageLoadingEndTime()方法手动埋点，设置页面结束时间点。

1、相关接口。

```
NBSAppAgent.setPageLoadingEndTime(Class clazz);
```

2、代码示例。

```
//用户可以在页面加载回调中（或其他合适的位置），调用setPageLoadingEndTime()方法
private void showSuccess() {
    networkStateView.showSuccess();
//传入当前Activity.class或当前Fragment.class
    NBSAppAgent.setPageLoadingEndTime(MainActivity.class);
}
```

首次启动功能开关

首次启动应用时，SDK默认只开启崩溃数据采集（之后的启动按连接服务器下发的配置开启相应功能）。用户可以通过接口自定义首次启动SDK的功能开关。

1、相关接口。

```
NBSAppAgent.setStartOption(int option);
//SDK定义了功能开关如下：
//网络数据采集
NBSAppAgent.HTTP_NETWORK_ENABLED = 1;
//UI数据采集（启动、页面、操作数据）
NBSAppAgent.UI_ENABLED = 2;
//崩溃数据采集
NBSAppAgent.CRASH_ENABLED = 4;
//WebView数据采集
NBSAppAgent.WEBVIEW_ENABLED = 8;
//Socket Hook
NBSAppAgent.SOCKET_DATA_ENABLED = 16;
//跨应用功能
NBSAppAgent.CROSS_APP_ENABLED = 32;
//卡顿数据采集
NBSAppAgent.ANR_ENABLED = 64;
//行为数据采集
NBSAppAgent.USER_ACTION_ENABLED = 128;
//CND数据采集
NBSAppAgent.CDN_ENDBLED = 256;
```

2、代码示例。

```
public void onCreate() {
  NBSAppAgent.setLicenseKey("AppKey")
  .setStartOption(NBSAppAgent.HTTP_NETWORK_ENABLED | NBSAppAgent.UI_ENABLED |
NBSAppAgent.CRASH_ENABLED)
//首次启动开启网络、UI、崩溃数据采集
  .start(this.getApplicationContext());
}
```

自定义启动

听云SDK采集冷启动(首次启动)数据，以第一个启动的Activity的onResume()方法结束作为启动时间结束点。用户也可以自己定义启动结束时间点。

1、相关接口。

```
NBSAppAgent.isCustomAppStart(boolean b);
//默认关闭，如需开启，设置为true
NBSAppAgent.setCustomOnResumeEndIns(String className);
//传入启动的第一个Activity的class.getName
```

2、代码示例。

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
    NBSAppAgent.setLicenseKey("AppKey")
    .isCustomAppStart(true)//在初始化SDK时调用,开启自定义启动时间功能
    .start(this.getApplicationContext());
    }
}
```

```
public class SplashActivity extends Activity {
//启动的第一个Activity
}

public class MainActivity extends Activity {
//启动的第二个Activity
    @Override
    public void onResume() {
    super.onResume();
    NBSAppAgent.setCustomOnResumeEndIns(SplashActivity.class.getName());
    //以第二个Activity的onResume()方法作为启动结束时间
    }
}
```

自定义渠道

开发者可以在初始化听云SDK时设置自定义的渠道名称。

1、相关接口。

```
NBSAppAgent.setChannelID(String channelID);
```

2、代码示例。

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
    NBSAppAgent.setLicenseKey("AppKey")
    .setChannelID("应用宝")//在初始化SDK时调用接口设置渠道
    .start(this.getApplicationContext());
    }
}
```