Analytics > Log & Crash Search > AndroidNDK SDK 사용 가이드

Log & Crash AndroidNDK SDK는 Log & Crash Search 수집 서버에 로그를 보내는 기능을 제공합니다.
Log & Crash AndroidNDK SDK 특·장점은 다음과 같습니다.

- 로그를 수집 서버로 보냅니다.
- 앱에서 발생한 크래시 로그를 수집 서버로 보냅니다.
- Log & Crash Search 에서 전송된 로그를 조회 및 검색이 가능합니다.
- 멀티 쓰레딩 환경에서 동작합니다.

지원 환경

- Android 2.3.3, API Level 10 이상
- AndroidNDK 최신 버전 권장
- 지원 ABI: armeabi, armeabi-v7a, x86

다운로드

TOAST Document에서 Android SDK를 받을 수 있습니다.

```
[DOCUMENTS] > [Download] > [Analytics > Log & Crash Search] > [AndroidNDK SDK] 클릭
```

설치

구성

AndroidNDK SDK는 다음과 같이 구성되어 있습니다.

```
docs/                       ; AndroidNDK SDK 문서
include/toast/logncrash.h   ; C++ 해더 파일
androidndk-sdk/
    obj/                    ; AndroidNDK SDK Static 라이브러리
    PrebuiltStaticLib.mk    ; Static 라이브러리 mk 파일
    libs/                   ; AndroidNDK SDK Shared 라이브러리
    PrebuiltSharedLib.mk    ; Shared 라이브러리 mk 파일
androidndk-sdk-sample/      ; Android JNI 샘플
```

SDK 샘플

같이 제공되는 androidndk-sdk-sample/에 대해 설명합니다.

1. androidndk-sdk-sample/로 이동합니다.

2. <ndk_path>/ndk-build로 NDK 부분을 빌드합니다.
3. Eclipse로 Android project를 불러옵니다.
4. AndroidNDKSample.java를 열어 발급받은 앱키로 수정합니다.
5. 실행합니다.

크래시 로그를 Log & Crash Search에서 보기 위해서는 심볼 파일을 올려줘야 합니다.

1. obj/local//liblogncrashjni_sample.so를 zip으로 압축합니다.
2. Toast Cloud Console에서 "Analytics - Log & Crash Search - Settings - 심볼 파일" 로 이동해서 압축된 zip 파일을 올립니다. 이때 androidndk-sdk-sample과 같은 프로젝트 버전으로 올려야 합니다.
3. androidndk-sdk-sample 을 실행하여 크래시 로그를 발생시킵니다.
    - 크래시를 발생시키 위해서 "Initialize" 버튼을 누르고 "Test 2" 버튼을 눌러 줍니다.
    - Toast Cloud Console에서 Analytics - Log & Crash Search - Log Search 에 크래시 로그가 도착한 것을 확인하고 "DmpData" 필드에 있는 "보기"를 눌러 확인합니다.
4. "DmpData" 필드 "보기"가 동작하지 않으면 다음 사항을 체크해야 합니다.
    - 크래시 로그와 심볼 사이에 프로젝트 버전이 맞지 않는 경우
    - 심볼 파일을 잘못 올린 경우

사용 예

1.jni/Application.mk에 다음 내용을 추가합니다.

```
 ...
 APP_ABI := armeabi armeabi-v7a x86
 APP_PLATFORM := android-9
 APP_STL := gnustl_static
 ...
```

2.jni/Android.mk에 다음 내용을 추가합니다.

```
...
 LOCAL_STATIC_LIBRARIES := logncrash_androidndk_static
 ...
 include $(LOCAL_PATH)/<androidndk_sdk_path>/androidndk-
sdk/PrebuiltStaticLib.mk
```

- Shared 라이브러리를 사용하시려면 다음과 같이 선언합니다.

```
...
  LOCAL_SHARED_LIBRARIES := logncrash_androidndk
  ...
  include $(LOCAL_PATH)/<androidndk_sdk_path>/androidndk-
sdk/PrebuiltSharedLib.mk
```

3.toast/logncrash.h 를 인클루드해 주고 ToastLog class를 사용합니다.

```
...
 #include "toast/logncrash.h"
 ...

     ToastLog* log = GetToastLog();

     if (LOGNCRASH_LOG_OK != log->initialize(APP_KEY, VERSION, COLLECTOR_ADDR,
COLLECTOR_PORT)) {
         fprintf(stderr, "ERROR in initialize()\n");
         return -1;
     }

     if (!log->info("info() TEST 1")) {
         fprintf(stderr, "ERROR in info()\n");
     }
     ...

     DestroyToastLog();
```

4.<ndk_path>/ndk-build로 NDK 부분을 빌드합니다.

5.AndroidNDK SDK가 정상적으로 동작하기 위해서 AndroidManifest.xml에 다음 퍼미션을 주어야 합니다.

```
...
 <uses-permission android:name="android.permission.INTERNET"/>
 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
 <uses-permission
android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
 ...
```

6.JNI 라이브러리를 불러옵니다.

```
static {
     System.loadLibrary("logncrashjni_sample");
 }
```

- Shared 라이브러리를 사용하실 때에는 미리 liblogncrash_androidndk.so를 불러줘야 합니다.

```
static {
     System.loadLibrary("logncrash_androidndk");
     System.loadLibrary("logncrashjni_sample");
 }
```

AndroidNDK SDK는 Java Exception을 처리할 수 없습니다. AndroidNDK SDK는 C++ Native code를 위해서 제작되었기 때문입니다.
androidndk-sdk-sample/jni/Android.mk 파일과 AndroidNDK에 포함된 문서(<ndk_path>/docs/)를 참조해 주세요.

API List

toast::logncrash::ToastLog class에서 제공하는 기능들을 설명합니다.

ToastLog 인스턴스 할당/해제

```
toast::logncrash::ToastLog* GetToastLog();

void DestroyToastLog();
```

- ToastLog instance를 할당하고 해제합니다.
- 싱글톤 방식으로 하나의 인스턴스만 반환됩니다.
- 반환된 ToastLog instance에 대해서 delete를 하면 안됩니다. 제거하기 위해서는 반드시 DestroyToastLog()를 호출하셔야 합니다.

초기화/해제

```
#define LOGNCRASH_VERSION         "1.0.0"
#define LOGNCRASH_COLLECTOR_ADDR  "api-logncrash.cloud.toast.com"
#define LOGNCRASH_COLLECTOR_PORT  80
#ifdef WIN32
#define LOGNCRASH_LOGSOURCE       "logncrash-windows"
#else   //#ifdef WIN32
#define LOGNCRASH_LOGSOURCE       "logncrash-linux"
#endif  //#ifdef WIN32
#define LOGNCRASH_LOGTYPE         "logncrash-log"

#define LOGNCRASH_LOG_OK            0
#define LOGNCRASH_LOG_ERROR         -1
#define LOGNCRASH_LOG_ERROR_APPKEY  -2
#define LOGNCRASH_LOG_ERROR_VERSION -3
#define LOGNCRASH_LOG_ERROR_ADDRESS -4
#define LOGNCRASH_LOG_ERROR_PORT    -5

int32_t initialize(
  const char* appKey,
  const char* version = LOGNCRASH_VERSION,
  const char* collectorAddr = LOGNCRASH_COLLECTOR_ADDR,
  const uint16_t collectorPort = LOGNCRASH_COLLECTOR_PORT,
  const char* logSource = LOGNCRASH_LOGSOURCE,
  const char* logType = LOGNCRASH_LOGTYPE);

void destroy();
```

- ToastLog를 초기화하고 해제합니다.
- ToastLog 기능이 제대로 동작하기 위해서는 반드시 initialize()가 호출되어야 합니다.

- 파라미터
  - appKey: 앱키
  - version: 앱 버전
  - collectorAddr: 수집서버 주소
    - Log & Crash 수집서버: api-logncrash.cloud.toast.com
  - collectorPort: 수집서버 포트
  - logSource: 로그 소스
  - logType: 로그 타입
  - clientHost: Host를 구하는 방식
    - true: ioctl 방식을 사용하여 client에서 host를 구함
    - false: server에서 전달된 값으로 host를 구함
  - asyncStart: SendThread를 Lock시킨 상태로 시작. Lock 상태에서 로그가 발생하는 경우 서버로 전송하지 않고 큐에 저장하며 대기. Crash가 발생하거나 StartSendThread 함수를 실행하는 경우 Lock 해제
- initialize() 반환값
  - LOGNCRASH_LOG_OK: 0, 초기화 성공
  - LOGNCRASH_LOG_ERROR: -1, 내부 에러 코드
  - LOGNCRASH_LOG_ERROR_APPKEY: -2, 앱키가 잘못된 경우
  - LOGNCRASH_LOG_ERROR_VERSION: -3, 버전이 잘못된 경우
  - LOGNCRASH_LOG_ERROR_ADDRESS: -4, 수집 서버 주소가 잘못된 경우
  - LOGNCRASH_LOG_ERROR_PORT: -5, 수집 서버 포트가 잘못된 경우

SendThread Lock 상태 해제

```
        void StartSendThread();
```

- SendThread를 전송 가능 상태로 변경

로그 보내기

```
bool sendLog(
    const LogNCrashLogLevel logLevel,
    const char* message,
    const char* errorCode = NULL,
    const char* location = NULL);
```

- 지정된 logLevel로 로그를 보냅니다.
- 파라미터
  - logLevel: 전송할 logLevel. setLogLevel()로 지정된 logLevel보다 큰 logLevel은 전송이 안됩니다.
  - message: 전송할 메시지
  - errorCode: 에러 코드. NULL이나 ""을 쓰면 전송되지 않습니다.
  - location: 에러 위치. NULL이나 ""을 쓰면 전송되지 않습니다.

- 반환값
  - 성공시 true
  - logLevel이 크거나, message가 비어있는 경우 false
- 참고
  - setLogLevel(), getLogLevel();

```
bool debug(const char* message, const char* errorCode = NULL, const char*
location = NULL);

bool info(const char* message, const char* errorCode = NULL, const char*
location = NULL);

bool warn(const char* message, const char* errorCode = NULL, const char*
location = NULL);

bool error(const char* message, const char* errorCode = NULL, const char*
location = NULL);

bool fatal(const char* message, const char* errorCode = NULL, const char*
location = NULL);
```

- 정해진 DEBUG, INFO, WARN, ERROR, FATAL 로그를 보냅니다.
- logLevel이 고정되어 있다는 점 이외에는 sendLog()와 같습니다.
- 반환값
  - 성공시 true
  - logLevel이 크거나, message가 비어있는 경우 false

로그 레벨 지정하기

```
typedef enum {
    LOGNCRASH_FATAL   = 0,
    LOGNCRASH_ERROR   = 3,
    LOGNCRASH_WARN    = 4,
    LOGNCRASH_INFO    = 5,
    LOGNCRASH_DEBUG   = 7,
    LOGNCRASH_TRACE   = LOGNCRASH_DEBUG,
} LogNCrashLogLevel;

LogNCrashLogLevel getLogLevel();

void setLogLevel(const LogNCrashLogLevel logLevel);
```

- ToastLog instance의 logLevel을 구하거나 지정합니다.
- ToastLog 기본값은 LOGNCRASH_INFO입니다. 따라서 debug() 함수를 사용하시려면 setLogLevel(LOGNCRASH_DEBUG)로 설정해주셔야 합니다.

커스텀 키 지정하기

```
bool addCustomKey(const char* key, const char* value);

void removeCustomKey(const char* key);

void clearCustomKeys();
```

- 커스텀 키를 추가, 삭제, 전부 삭제 기능을 제공합니다.
- 커스텀 키는 대소문자로 시작하고 대소문자, 숫자, '-', '*'만 사용하실수 있습니다. ([A-Za-z][A-Za-z0-9-]*)*
- 커스텀 키는 최대 64 문자입니다.
- 커스텀 키에 대소문자 관계없이 다음 이름은 사용하실 수 없습니다.
  - projectname, projectversion, host, body, logsource, logtype
  - logType, sendTime, logLevel, userId, platform
  - dmpdata, dmpreport
- addCustomKey() 반환값
  - 성공시 true
  - key 형식이 맞지 않으면 추가 실패시 false

호스트 타입 지정하기

```
bool setHostMode(int mode);
```

- mode 가 0인 경우: Private IP를 구하여 host 필드를 채웁니다. Private IP를 구하는데 실패 하면 Public IP 로 host 필드를 채웁니다.
- mode 가 1인 경우: Public IP 로 host 필드를 채웁니다.

크래시 처리하기

```
typedef enum {
    LOGNCRASH_LANG_DEFAULT     = 0,
    LOGNCRASH_LANG_ENGLISH     = 1,
    LOGNCRASH_LANG_KOREAN      = 2,
    LOGNCRASH_LANG_JAPANESE    = 3,
    LOGNCRASH_LANG_CHINESE     = 4,
    LOGNCRASH_LANG_CHINESE_TRADITIONAL = 5,
    LOGNCRASH_LANG_CHINESE_SIMPLIFIED  = LOGNCRASH_LANG_CHINESE,
} LogNCrashLangType;

#ifdef WIN32
typedef void (__cdecl *LogNCrashCallbackType)(void *data);
#else   //#ifdef WIN32
typedef void (*LogNCrashCallbackType)(void *data);
#endif  //#ifdef WIN32
```

```
bool openCrashCatcher(const bool bBackground = false, const LogNCrashLangType
langType = LOGNCRASH_LANG_DEFAULT);
void closeCrashCatcher();

void setCrashCallback(const LogNCrashCallbackType cb, void* cbData = NULL);
```

- 크래시 처리를 시작하거나 종료합니다.
- openCrashCatcher 파라미터
  - bBackground: 크래시 리포터 동작 방식 설정합니다.
  - langType: 크래시 리포터 GUI 언어를 설정합니다.
- openCrashCatcher 반환값
  - 설정 성공시 true
  - 설정 실패시 false
  - AndroidNDK SDK는 단말의 /sdcard 디렉토리를 사용합니다. 단말에 위 디렉토리가 없으면 제대로 동작 안할수 있습니다.

중복 제거 모드 설정

- 2.4.0 이상 SDK 부터 일반 로그에 중복 제거 로직이 적용되었습니다
- 중복 로그 기능이 켜져있는 경우 body와 logLevel의 내용이 같은 로그가 발생하면 전송하지 않습니다.

```
public static void setDuplicate(bool enable)
```

- true: (Default 값) 중복 제거 로직 활성화
- false: 중복 제거 로직 비활성화

기타 설정

```
const char* getUserId();

void setUserId(const char* userId);
```

- 사용자 ID를 구하거나 지정합니다.

```
void enableHostField();

void disableHostField();
```

- Host 필드를 활성화하거나 비활성화합니다.

```
void enablePlatformField();

void disablePlatformField();
```

- Platform 필드를 활성화하거나 비활성화합니다.