Data & Analytics > Log & Crash Search > Logback SDK 사용 가이드

Log & Crash Logback SDK는 Log & Crash Search 수집 서버에 로그를 보내는 기능을 제공합니다. Log & Crash Search 에서 전송된 로그를 조회 및 검색이 가능하고 멀티 쓰레딩 환경에서 동작합니다.

1. Log & Crash Logback SDK 추가

gov-logncrash-java-sdk3-3.0.0.jar를 의존성에 추가합니다. NHN Cloud Document에서 Log & Crash Logback SDK를 다운로드 할 수 있습니다.

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

- Log & Crash Logback SDK는 `logback-classic 1.2.3+, apache httpclient 4.5+, json 20171018+` 의 라이브러리에 의존성을 갖고 있습니다.
- 참조하는 library가 중복될 경우, 문제가 발생할 수 있으므로 상위 버전을 사용하는 것을 권장합니다.

2. Log & Crash Logback SDK에 필요한 의존성 추가

2.1 Maven 설치

pom.xml에 dependency를 추가합니다.

```
<dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
    <version>20171018</version>
</dependency>
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5</version>
</dependency>
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.2.3</version>
</dependency>
```

2.2 Gradle 설치

```
dependencies {
    compile 'org.json:json:20171018'
    compile 'org.apache.httpcomponents:httpclient:4.5'
```

```
    compile 'ch.qos.logback:logback-classic:1.2.3'
}
```

3. Logger 설정 및 옵션

logback.xml을 기준으로 설명합니다.

- Logback의 AsyncAppender를 기본으로 사용하여 로그를 비동기로 수집하여 로그 수집 서버로 전송합니다.
- Log & Crash Logback SDK의 LogNCrashHttpAppender로 전송 로그의 자세한 항목을 설정할 수 있습니다.

```
<!-- LogNCrashHttpAppender 선언 -->
<appender name="logNCrashHttp" class="com.toast.java.logncrash.logback.LogNCrashHttpAppender">
    <appKey>앱키</appKey>
    <logSource>운영</logSource>
    <version>1.0.0</version>
    <logType>감사 로그</logType>
    <debug>true</debug>
    <category>로그 서비스</category>
    <errorCodeType>action</errorCodeType>
</appender>
<!-- LogNCrashHttpAppender를 포함한 AsyncAppender 선언 -->
<appender name="LNCS-APPENDER" class="ch.qos.logback.classic.AsyncAppender">
    <!-- Logback의 AsyncAppender 옵션 -->
    <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
        <level>INFO</level>
    </filter>
    <includeCallerData>false</includeCallerData>
    <queueSize>2048</queueSize>
    <neverBlock>true</neverBlock>
    <maxFlushTime>60000</maxFlushTime>
    <appender-ref ref="logNCrashHttp"/>
</appender>

<logger name="user-logger" additivity="false">
    <appender-ref ref="LNCS-APPENDER"/>
</logger>
```

3.1 Logback의 AsyncAppender 옵션

- 설정값들의 상세한 정보는 공식문서를 참조하십시오.

키	설명
includeCallerData	발신자의 정보 (class명, 줄번호 등)가 추가되어 수집 서버로 전송여부를 결정합니다. true 설정 시, 성능 저하를 일으킬 수 있습니다.
queueSize	blocking queue의 최대 수용 갯수로 기본값은 256입니다.
neverBlock	false로 설정한 경우 큐가 가득찬 상황에서 appender는 메세지 유실을 방지하기 위해 application을 block 합니다. true로 설정된 경우 application을 멈추지 않기 위해 메세지를 버립니다.

키	설명
maxFlushTime	LoggerContext이 정지하면 AsyncAppender의 stop 메소드는 작업 스레드가 timeout 될때까지 대기합니다. maxFlushTime를 사용하면, timeout 시간을 밀리초로 설정할 수 있습니다. 해당 시간안에 처리하지 못한 이벤트는 삭제됩니다.

3.2 Log & Crash Logback SDK의 LogNCrashHttpAppender 옵션

appKey를 제외한 나머지 값들은 선택 항목으로 param을 기입하지 않으면, 기본값으로 정보가 입력됩니다.

키	설명	기본값
appKey	logncrah 콘솔에서 활성화한 projectKey입니다.	필수값
logSource	alpha, beta, real 등 실행 되는 환경 설정 정보입니다. Spring Profile을 사용하는 경우, ${spring.profiles.active} 사용합니다.	http-logback
version	발신자의 프로젝트 버전을 명시합니다.	1.0.0
logType	수집하는 log의 type을 명시합니다.	log
debug	boolean 값으로 console에 sdk 로그 정보 출력이 필요 여부를 결정합니다.	true
category	수집하는 log의 category를 명시합니다.	
errorCodeType	에러 발생 시 수집되는 에러 정보의 타입을 설정합니다. default, action, message, mdc 타입이 존재합니다. - default: Throwable 정보를 사용합니다. - action: URL path의 정보도 포함하여 에러를 반환합니다. - message: logger에 설정한 메시지만 반환합니다. - mdc: MDC의 errorCode 항목값을 설정해서 사용합니다.	default

3.3 사용자 정의 옵션

slf4j의 MDC를 사용하여 Log & Crash의 LogNCrashHttpAppender에서 정의되지 않은 항목을 추가할 수 있습니다. (단, `category` 는 변경할 수 있습니다.)

```
MDC.put("userid", "nhnent-userId");
MDC.put("userIp", "127.0.0.1");
...
MDC.clear();
```

MDC로 변경불가능한 에약어 (대소문자를 구분하지 않습니다.)

projectName	clientIp	projectVersion	url	logSource	headers
form	logType	cookie	body	agent	logLevel

projectName	clientIp	projectVersion	url	logSource	headers
host	referer	sendTime	dmpData	dmpFormat	

4. LogNCrash SDK 사용 예

Java에서 다음과 같이 사용합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.slf4j.MDC;

public class LogNCrash {
    private static final Logger USER_LOG = LoggerFactory.getLogger("user-logger");

    public void logging() {
        logger.debug("LogNCrash Debug.") ;

        // Log & Crash에서 예약된 항목 이외, 사용자 정의 항목 사용 시 MDC 활용
        MDC.put("userid", "nhnent-userId");
        logger.warn("Customize items...") ;
        MDC.clear();

        try {
            String logncrash = null;
            if(true) {
                System.out.print(logncrash.toString());
            }
        } catch(Exception e) {
            logger.error("LogNCrash Exception.", e)
        }
    }
}
```

5. FAQ

Q. Log & Crash Logback SDK의 LogNCrashHttpAppender만으로 로그를 전송할 수는 없나요?

Log & Crash Logback SDK의 LogNCrashHttpAppender는 sync방식으로 수집 서버에 로그를 전송하므로 가능합니다. 로그의 유실이 최소화되지만, Log & Crash Logback 시스템 장애 시 Application의 성능 저하가 발생할 수 있으므로, Async Appender를 사용하는 것을 권장합니다.

Q. batch program(project)에서 logncrash client를 사용하려면?

batch 프로그램의 마지막에 몇초간 대기하는 코드를 추가합니다.

```
try {
    Thread.sleep(3000L);
```

```
} catch (InterruptedException ignore){}
```

Java batch program에서는 main thread가 바로 종료되기 때문에 Logback의 AsyncAppender의 데몬 스레드가 생성되어 로그를 보내기 전에 batch 애플리케이션이 종료됩니다. 데몬 스레드와 상관없이 살아 있는 일반 스레드가 없을 경우에 JVM은 바로 종료됩니다. 따라서, 위와 같이 프로그램 마지막에 대기하는 코드를 추가하여 모든 로그를 보내고 나서 종료하도록 합니다.

Q. Java stack trace를 logback(Log & Crash Search 포함)에 로깅하려면?

logback 이용하여 stack trace를 출력하려면 log.error(e.getMessage(), e); 형태를 사용합니다. SLF4J Logger는 메소드의 인자로 Throwable만 받는 로깅 메소드는 지원하지 않습니다

```
try {
    ...
} catch (Exception e){
    logger.error(e.getMessage(), e);
}
```

Q. WAS 에서 사용시 안정적인 종료를 하려면?

에러로그가 전송중인 상황에서 WAS(Tomcat 등)가 종료되는 경우에는, 다음과 같은 Exception이 발생하며 WAS가 정상적으로 종료되지 않을 때가 있습니다.

이러한 현상을 방지하기 위해서는 WAS 종료시에 LoggerContext 인스턴스에 대해 stop() 메소드를 호출하여 LogNCrashHttpAppendet를 close하면 안정적으로 종료가 가능합니다.

Spring에서는 Log4J에 대해서는 org.springframework.web.util.Log4jConfigListener를 제공하지만, Logback에 대해서는 안정적인 종료를 지원하는 Listener를 제공하지 않고 있습니다.

Log & Crash Logback SDK에서는 자체적으로 LogbackShutdownListener를 제공합니다. web.xml에 다음과 같은 설정을 추가하면 WAS 종료시에 에러로그 전송이 일어나도 안정적인 종료가 가능합니다.

```
<listener>
    <listener-class>
        com.toast.java.logncrash.logback.LogbackShutdownListener
    </listener-class>
</listener>
```