가장 많은 고객이 선택한 클라우드 모니터링 서비스 1위, 와탭

**당신의, IT 서비스의 성능을 극대화합니다.**

WhaTap Docs

# None-HTTP Endpoint Guide

Whatap Support

Version 1.0.2

# Table of Contents

# None-HTTP Endpoint 가이드

제목 : None-HTTP Endpoint Guide
작성자 : Whatap Support
이메일 : support@whatap.io
날짜 : 2019-10-22
버전 : 1.0.2

설명 : 본 문서는 Java 데몬, 배치, 또는 HTTP 서버 내부의 백그라운드 스레드에 대한 성능 추적을 위한 모니터링 방법을 설명합니다.

# Chapter 1. 트랜잭션 End-Point

트랜잭션이 시작되는 지점을 의미합니다.
HTTP 서블릿의 경우 HttpServlet.service() 혹은 Filter.doFilter()가 트랜잭션의 시작이고 이곳을 트랜잭션 엔드포인트라고 합니다.

# Chapter 2. 프로파일 정보

트랜잭션 엔드포인트로 지정된 메소드가 시작해서 종료될때까지의 정보를 프로파일 정보라고 합니다. HTTP 트랜잭션의 경우 HttpServlet.service() 가 트랜잭션 엔드포인트로 사전 정의되어 있으므로 대부분의 경우 프로파일 정보 수집을 위한 설정이 불필요 합니다. 사전 정의 되어 있지 않은 Non-HTTP 트랜잭션인 경우와 HTTP 트랜잭션임에도 HttpServlet.service() 가 사용되지 않는 경우 이들에 대한 프로파일 정보를 확보하기 위해서는 트랜잭션 엔드포인트를 지정해야 합니다.

# Chapter 3. 설정

설정 순서는 다음과 같습니다. 명확한 End-Point 를 알고 있는 경우라면 다른 과정 없이 트랜잭션 엔드포인트 지정 해 주시면 됩니다.

1. back stack 확보
2. 트랜잭션 엔드포인트 선별
3. 트랜잭션 엔드포인트 지정

## 3.1. Back Stack 확보

### 3.1.1. 사용하는 TCP 포트를 알고 있는 경우

소켓통신 데몬, DB작업을 수행하는 배치와 같이 트랜잭션에서 사용되는 TCP 포트가 명확한 경우 "stacklog_socket_port" 옵션을 사용하여 Back Stack을 확보 할 수 있습니다.

stacklog_socket_port 옵션을 통해 확보된 Back Stack 정보 Agent 로그에서 확인 할 수 있습니다.

whatap.conf

```
#오라클 1521포트를 사용하는 것이 명확한 DB배치 데몬인 경우
stacklog_socket_port=1521
```

### 3.1.2. 사용하는 Method 를 알고 있는 경우

트랜잭션에서 사용되는 Method를 대략이나마 알고 있는 경우 메소드 프로파일링 옵션을 사용하여 BackStack을 확보 할 수 있습니다. 사용하는 메소드를 알고 있더라도 트랜잭션 엔드포인트보다 먼저 호출되는 경우라면 적절한 포인트를 찾기 어려워 집니다. 따라서 트랜잭션 수행 과정중에 사용되는 메소드를 지정해야 합니다.

whatap.conf

```
# 메소드 프로파일을 설정합니다.
hook_method_patterns=io.whatap.KnownClass.knownMethod,*.IKnowClassName.*
# 프로파일 대상 메소드가 트랜잭션 시작점이 되도록 지정합니다.
trace_auto_transaction_enabled=true
# 트랜잭션 시작시 StackTrace를 기록합니다.
trace_auto_transaction_backstack_enabled=true
```

### 3.1.3. 사전 정보가 불명확한 경우

1. [서버] [더보기] [로디드 클래스] 의 목록을 바탕으로 사용될만한 클래스, 메소드를 유추할 수 있습니다. 이후 메소드 프로파일 설정을 통하여 BackStack를 시도합니다.
2. [서버] [더보기] [스레드 목록] 에서 전체 스레드를 확인 후 이들 중 실제 트랜잭션에 해당하는 스레드의 StackTrace 를 확보하여 유추합니다. 또는 kill -3 를 통한 javacore 나 jstack 명령어를 통한 수행시점 StackTrace 를 확보 합니다. 이후 메소드 프로파일 설정을 통하여 BackStack를 시도합니다.

## 3.2. 트랜잭션 엔드포인트 선별

BackStack 확보를 통해 확보한 Stack 정보로 트랜잭션 엔드포인트를 추정합니다.

확보한 Stack이 다음과 같다면, execute, process, run 중 하나의 메소드가 트랜잭션 엔드포인트가 될 수 있음을 메소드 명으로 유추할 수 있습니다. 이후 디컴파일, 소스분석 과정등을 통해 이들 중 엔드포인트로서 적절한 메소드를 찾아냅니다.

Sample Stack

```
jdbc.FakePreparedStatement.executeQuery(FakePreparedStatement.java),
com.virtual.dao.SelectDAO.execute2(SelectDAO.java:29),
com.virtual.web.SimulaNonHttp.execute(SimulaNonHttp.java:147), ①
com.virtual.web.SimulaNonHttp.process(SimulaNonHttp.java:76), ②
com.virtual.web.SimulaNonHttp.run(SimulaNonHttp.java:100) ③
```

① execute

② process

③ run

프로파일 정보는 트랜잭션 종료와 함께 전송하고 서버에서는 이를 히트맵 하나의 점으로 표현합니다.
트랜잭션 엔드포인트 선별의 중요한 기준은 "종료" 여부 입니다.

## 3.3. 트랜잭션 엔드포인트 지정

선별한 트랜잭션 엔드포인트를 hook_service_patterns, hook_httpservlet_classes 옵션을 사용하여 설정합니다.

whatap.conf

```
#Non-HTTP인 경우 hook_service_patterns 사용
hook_service_patterns=com.virtual.web.SimulaNonHttp.process

#HTTP인 경우 hook_httpservlet_classes 사용
hook_httpservlet_classes=io.whatap.KnownServletClass
```

## 3.4. 트랜잭션 이름 정의

대부분의 경우 메소드 명칭으로 충분히 트랜잭션을 구분할 수 있다. 그래서 와탭은 메소드의 첫번째 String 파라미터를 트랜잭션 이름으로 사용한다.그런데 문자열 파라미터가 없는 경우에는 커스터마이징을 해야하는데 이때 와탭의 플러그인을 사용한다

```
Object url =((java.util.HashMap)$point.getArgs()[0]).get("url");
$ctx.service((String)url);
//println("url="+url);
```

Java Agent Plugin 가이드 에서 Plugin 사용에 관한 설명을 확인 할 수 있습니다.

## 3.5. 관련 옵션

Java Agent 가이드 NON-Http 트랜잭션 추적 항목에서 사용 할 수 있는 다양한 옵션을 확인 할 수 있습니다.