로그프레소 CLI 레퍼런스 매뉴얼

목차

1장. 로그프레소 설치 4

1.1. 패키지 설치 및 시작, 중지,모니터링 4

1.2. JVM 실행 옵션 8

1.3. 웹 서버 설정 11

1.4. 라이센스 관리 13

1.5. 성능 측정 15

2장. 클라이언트 도구 17

2.1. 자바 클라이언트 콘솔 17

2.2. 데이터 익스포터 18

3장. 데이터베이스 관리 19

3.1. 계정 관리 19

3.2. 쿼리 22

3.3. 테이블 관리 26

3.4. 데이터 보안 30

3.5. 데이터 임포트 32

3.6. 파서 설정 33

3.7. 인덱스 관리 41

3.8. 캐시 설정 47

3.10. 데이터 백업 및 복원 51
3.11. FTP 연동 설정 58
3.12. JDBC 연동 설정 59
3.13. SSH 연동 설정 60
3.14. 페더레이션 구성 61

4장. 실시간 로그 수집 설정 **62**

4.1. 로그 수집 설정 62
4.2. 디렉터리 로그 파일 수집 설정 67
4.3. 선택자 로그 수집 설정 68
4.11. SNMP 트랩 로거 설정넷플로우 로그 수집 설정 80
4.12. JDBC 수집 설정 83
4.13. FTP 디렉터리 로그 파일 수집 설정 84
4.14. FTP 로테이션 로그 파일 수집 설정 85
4.15. SFTP 디렉터리 로그 파일 수집 설정 86
4.16. SFTP 로테이션 로그 파일 수집 설정 87
4.17. SSH 표준 출력 수집 설정 89
4.18. JMX 수집 설정 89

5장. 실시간 로그 필터링, 태깅 **92**

5.1. 로그 트랜스포머 설정 92
5.2. 로그 트랜스포머 설정정규표현식을 이용한 로그 필터링 93

6장. 관리되는 로거 **94**

6.1. 로그 저장 설정 94
6.2. 관리되는 로거 설정 95
6.3. 로그 수집 HA 설정 95

7장. 실시간 로그 출력 설정 96

7.1. 롤링 로그 파일 쓰기 설정 96

7.2. 시간대별 파일 쓰기 설정 97

7.3. 시스로그 전송자 로거 98

8장. 쿼리 관리 98

8.1. 스트림 쿼리 98

8.1.1. 스트림 쿼리 생성 99

8.1.2. 스트림 쿼리 삭제 100

8.1.3. 스트림 쿼리 목록 조회 100

8.1.4. 스트림 쿼리 목록 상세 조회 100

8.2. 예약된 쿼리 경보 101

9장. 경보설정 103

9.1. SMTP 설정 103

10장. 엑셀 및 외부 시스템 연동 107

10.1. API 키 관리 107

11장. 하둡 커넥터 109

11.1. 하둡 연동 설정 109

11.2. 실시간 HDFS 로그 수집 설정 110

11.3. 실시간 HDFS 로그 파일 출력 설정 110

1장. 로그프레소 설치

1.1. 패키지 설치 및 시작, 중지,모니터링

로그프레소는 java 플랫폼에 기반을 두고 운영되는 OSGi 번들 및 패키지 시스템이다.

- 사용 Platform : Linux, windows 서버(Linux 권장)
- Java 버전 : jre 7
- 구성 : 로그프레소, araqne DB

1.1.1. 패키지 설치

- 서버 설치 패키지를 설치할 서버의 프로그램 설치 디렉토리에 복사한다.
- ftp, sftp 혹은 기타 제공되는 파일 전송 가능한 방법을 사용하여 패키지를 복사한다.

1.1.2. 로그프레소 디렉토리 및 엔진 파일

- cache : 설치된 OSGi 번들 캐시 저장 위치
- log : 일자별로 롤링되는 araqne.log 파일 (기본 7일치 보관)
- data : 번들 데이터 저장 위치(설정, 로그DB 데이터, DB 인덱스)(데이터 저장 위치는 옵션에 따라서 변경 가능.)
- araqne-core-2.7.8-package.jar : 로그프레소 엔진 파일 (파일 버전 번호는 다를 수 있음.)
- logpresso.sh : 패키지 시작/중지 스크립트

프로그램 시작/중지 스크립트인 logpresso.sh 파일 내용 샘플

```
#!/bin/bash

##################################################################################################
export MALLOC_ARENA_MAX=1
export JAVA_HOME=/data/logpresso/jre1.7.0_45
HOSTNAME=`hostname`
SCRIPTNAME=$(basename $0)
INSTANCE_ID="$HOSTNAME"
#DATADIR="/data01"
#JAVA_OPTS="$JAVA_OPTS -Daraqne.data.dir=$DATADIR/logpresso-data"
```

```
##############################################################################################
 # MAYBE YOU DON'T NEED TO TOUCH BELOW HERE
##############################################################################################
 # GENERAL CONFIGURATION
 JAVA_OPTS="$JAVA_OPTS -DINSTANCE_ID=$INSTANCE_ID"
 JAVA_OPTS="$JAVA_OPTS -Dipojo.proxy=disabled"
 JAVA_OPTS="$JAVA_OPTS -Daraqne.ssh.timeout=0"
 JAVA_OPTS="$JAVA_OPTS -XX:+UseParallelGC -XX:+UseParallelOldGC -XX:+PrintGCDetails"
 JAVA_OPTS="$JAVA_OPTS -XX:NewRatio=1"
 JAVA_OPTS="$JAVA_OPTS -Xms5G -Xmx5G"
 JAVA_OPTS="$JAVA_OPTS -XX:MaxPermSize=300M"
 JAVA_OPTS="$JAVA_OPTS -XX:MaxDirectMemorySize=16G"
 ..........
```

1.1.3. 로그프레소 제공 포트

로그프레소가 실행이 되면 기본적으로 제공되는 포트.(옵션 설정으로 변경 가능.)

- 로그프레소 엔진 및 DB 접속을 위한 제공 포트 텔넷 : (7004/tcp) 및 SSH (7022/tcp)
- 로그프레소 엔진 접속 방법 : ssh -p7022 localhost, telnet localhost 7004

1.1.4. 로그프레소 시작 및 중지

로그프레소 엔진은 `logpresso.sh` 스크립트를 사용하여 시작과 중지 기능을 수행한다. 서버가 윈도우 OS인 경우는 윈도우 서비스를 사용하여 시작과 중지 기능을 수행한다.

- [start]

```
[root@logpresso]# ./logpresso.sh start
starting araqne-core with INSTANCE_ID=centos..
[root@logpresso]#
```

- [stop]

```
[root@ logpresso]# ./logpresso.sh stop
waiting for shutdown...done
[root@logpresso]#
```

1.1.5. 로그프레소 시작 여부 점검

- `ps -ef` 명령어를 사용하여 java 프로세스가 정상적으로 구동되고 있는지 확인한다.

[root@logpresso]# ps -ef|grep java root 5259 1 90 05:31 pts/0 00:00:04 /usr/java/jdk1.7.0_25/bin/java -Daraqne.data.dir=/home/logpresso/data -DINSTANCE_ID=centos -Dipojo.proxy=disabled -Daraqne.ssh.timeout=0 -XX:+UseParallelGC -XX:+UseParallelOldGC -XX:+PrintGCDetails -XX:NewRatio=1 -Xms500M -Xmx900M -XX:MaxPermSize=300M -XX:MaxDirectMemorySize=100M -jar araqne-core-2.5.6-package.jar

[root@logpresso]#

1.1.6. 로그 수집 룰 정상 동작 여부 확인

- 수집 로거의 상태를 `logapi.loggers` 명령어를 사용하여 확인한다.

```
araqne@leehong Logpresso> logapi.loggers
Loggers
----------------------
+-----------+--------------+---------+------------+------------+--------------------------+----
|   name    |   factory    |  status | intvl.(ms) |  log count |    last log   | stop reason |
+-----------+--------------+---------+------------+------------+--------------------------+----
| local\118 | sftp-dirwatch | stopped |         0 | 35,554,638 | 2014-0:17 +00 |           |
+-----------+--------------+---------+------------+------------+--------------------------+----
araqne@leehong Logpresso>
```

1.1.7. 로그 DB 저장 룰 등록 여부 확인

- 수집된 로그를 DB에 저장하는 로거의 상태를 `logpresso.loggers` 명령어를 사용하여 확인한다.

```
araqne@leehong Logpresso> logpresso.loggers
Managed Loggers
-------------------
+--------+--------------------------+--------------+----------+------------+-----------+
|  type  |          name            |     table    | host tag |   HA mode  | HA logger |
+--------+--------------------------+--------------+----------+------------+-----------+
| sentry | 172.20.20.3\leehong-sftp | leehong-sftp | null     | standalone | null      |
| sentry | 172.20.20.3\snmpget      | snmpget      | null     | standalone | null      |
```

```
| sentry | 172.20.20.3\snmpttt      | snmptest     | null     | standalone | null      |
| local  | local\118                | test         | null     | standalone | null      |
+--------+--------------------------+--------------+----------+------------+-----------+
```

1.1.8. 로그 DB 저장 모니터링

- 초당 처리되는 로그 건수를 `logpresso.trends` 명령어를 사용하여 모니터링 한다.

```
araqne@centos logpresso> logpresso.trends
Log Input Trend
Press Ctrl-C to stop..
0 logs/sec
100 logs/sec
……
```

1.1.9. DB 테이블 정보 확인

- `logstorage.table` 명령어를 사용하여 DB 테이블 전체 리스트를 확인한다.

```
araqne@centos araqne> logstorage.tables
```

- `logstorage.table` 명령어를 사용하여 특정 DB 테이블에 적용되어 있는 정보를 확인한다.

```
araqne@centos araqne> logstorage.table
Table
Table Metadata
---------------
_filetype=v3p
compression=snappy
parser=secure_chain
Storage information
--------------------
Data path: /home/araqne/logpresso-data/araqne-logstorage/log/11
Consumption: 56 bytes
araqne@centos araqne>
```

1.1.10. 로그수집하는 logapi logger

- `logapi.loggers` 명령어를 사용하여 등록된 전체 로거 목록 및 상태를 확인한다.

```
araqne@centos araqne> logapi.loggers
```

- `logapi.startLoggers` 명령어를 사용하여 등록 전체 로그 수집 로거 일괄 시작. (시작 시 interval을 주어야 함. 단위:밀리세컨드)

  araqne@centos araqne〉 logapi.startLoggers * 1000

- `logapi.startLogger` 명령어를 사용하여 특정 로그 수집 로거를 시작한다. (시작 시 interval을 주어야 함. 단위: 밀리세컨드)

  araqne@centos araqne〉 logapi.startLogger [로거이름] 1000

- `logapi.stopLoggers` 명령어를 사용하여 등록 전체 로그 수집 로거 일괄 중지한다.

  araqne@centos araqne〉 logapi.stopLoggers *

- `logapi.stopLogger` 명령어를 사용하여 특정 로그 수집 로거를 중지한다.

  araqne@centos araqne〉 logapi.stopLogger [로거이름]

- `logapi.removeLogger` 명령어를 사용하여 특정 로그 수집 로거 삭제한다.

  araqne@centos araqne〉 logapi.removeLogger [로거이름]

1.1.11. DB에 수집한 데이터 저장하는 logpresso logger

- `logpresso.loggers` 명령어를 사용하여 등록된 전체 로거 목록을 확인한다.

```
araqne@centos araqne> logpresso.loggers
```

- `logpresso.removeLogger` 명령어를 사용하여 등록된 로거를 삭제한다.

  araqne@centos araqne〉 logpresso.removeLogger [로거이름]

1.2. JVM 실행 옵션

아라크네 코어는 로그프레소를 구동하는 OSGi 애플리케이션 서버입니다. 로그프레소가 설치될 환경(CPU 코어 수, 물리 메모리 크기, 실시간 로그 입력량)에 맞추어 실행 매개변수를 조정할 수 있습니다.

자바 가상머신의 실행 매개변수에 따라 성능이 큰 폭으로 좌우될 수 있습니다:

1.2.1. 메모리 설정

- -Xms: 최소 자바 힙 메모리 양을 지정합니다. 최대치와 같게 설정하면, 실행 초기에 메모리 영역을 확장 및 재배치하면서 발생하는 성능 저하를 최소화 할 수 있습니다.
- -Xmx: 최대 자바 힙 메모리 양을 지정합니다. 캐시로 사용할 크기는 제외한 최대 메모리 크기를 지정합니다. 일반적으로 코어 수가 많을수록 메모리를 크게 잡아야 합니다. 하지만 100GB 단위로 너무 크게 잡아도 GC로 인한 큰 지연시간이 발생하므로, 시스템 사양에 따라 시험 및 조정이 필요합니다.
- -XX:MaxPermSize: Jython 스크립트 등 클래스 동적 생성을 자주 사용한다면 PermGen 영역을 크게 잡아야 합니다. 일반적으로는 300M 정도면 충분합니다.
- -XX:MaxDirectMemorySize: 다이렉트 버퍼를 사용하여 직접 메모리 할당이 가능한 메모리 영역의 최대 크기를 지정합니다. 이후에 다루겠지만, 캐시되는 데이터는 다이렉트 버퍼를 사용하므로 전체 캐시 용량의 총합보다 크게 지정해야 합니다. 만약 지정하지 않으면 -Xmx 설정과 동일한 값을 사용하게 됩니다. 다이렉트 버퍼 메모리가 모자라면 실행 중 OutOfMemoryException이 발생할 수 있습니다.
- -XX:NewRatio: YoungGen과 OldGen의 비율을 설정합니다. 예를 들어, NewRatio가 2이면 YoungGen과 OldGen이 1:2의 비율로 설정됩니다. 이 비율은 수명주기가 짧은 개체와 긴 개체의 비율이 어느 정도이냐에 따라 달라지는데, GC 통계를 보고 튜닝할 수 있습니다. 일반적으로 코어 수가 많을수록 YoungGen을 크게 설정할 필요가 있습니다. 물리 코어 수가 16개 이상일 때는 NewRatio를 1이나 2로 설정할 것을 권장합니다.

리눅스의 경우 glibc 메모리 풀 관련해서 반드시 아래 환경변수를 미리 설정해야 합니다:

```
export MALLOC_ARENA_MAX=1 ( 1,  2)
```

CPU 코어 수에 따라 가변적이지만 일반적인 구성은 아래와 같습니다:

물리메모리	운영체제/기타	자바 힙	다이렉트 버퍼	역인덱스 캐시	블룸필터 L0 캐시	블룸필터 L1 캐시
8G	2G	4G	2G	0G	0.5G	0.5G
16G	4G	6G	6G	0G	2.5G	2.5G
32G	4G	8G	20G	7G	6G	5G
64G	8G	12G	44G	24G	9G	7G
128G	12G	16G	100G	67G	15G	12G
256G	16G	25G	215G	175G	15G	15G

1.2.2. GC 설정

JVM은 다양한 GC 방식을 지정하는데, 기본적으로 ParallelGC나 ConcurrentGC 중 하나를 사용하는 것이 좋습니다. G1GC는 100GB 이상의 대용량 힙을 사용하는 시나리오에서 불가피할 때 고려해볼 수 있으나 추천되는 설정은 아닙니다. ParallelGC는 GC를 수행할 때 다수의 스레드를 사용하여 병렬로 GC를 수행합니다. 기본적으로는 YoungGen 영역만 병렬로 GC를 수행하지만, 설정에 따라 OldGen 영역도 병렬로 GC를 수행하게 할 수 있습니다. GC 시간이 최소화 되기 때문에 로그프레소처럼 Throughput 위주의 응용에 적합합니다:

- -XX:+UseParallelGC: ParallelGC를 사용하도록 설정합니다.
- -XX:+UseParallelOldGC: OldGen 영역도 병렬로 GC하도록 설정합니다. ConcurrentGC는 최대한 응용프로그램의 스레드가 동작하고 있는 상태에서 GC를 병행합니다. 응용프로그램이 동작하고 있을 때도 GC가 동작하므로 다른 GC에 비해 상대적으로 Throughput이 떨어지고, 대신 메모리가 빠르게 자주 회수되기 때문에 응용프로그램의 응답성이 좋아지는 효과가 있습니다. 물리 코어 수가 16개 이상으로 많을 때는 고려해 볼 수 있으나 기본 설정으로 권장하지는 않습니다:
- -XX:+UseConcMarkSweepGC: ConcurrentGC를 사용하도록 설정합니다.
- -XX:+UseParNewGC: YoungGen 영역의 메모리 재배치를 수행할 때 다수의 코어를 사용해서 병렬로 복사를 수행합니다. GC로 인한 정지 시간이 최소화 됩니다. 시스템 사양과 환경에 맞추어 다양한 GC 옵션을 시험해서 최적의 설정을 찾으시기 바랍니다.

1.2.3. 디렉터리 설정

- -Daraqne.dir: 아라크네 코어가 실행되는 기본 디렉터리 경로를 지정합니다. 뒤에 나오는 cache, data, log, download, home, plugin 디렉터리는 araqne.dir 경로 이하에 각각의 이름으로 된 디렉터리를 기본값으로 사용합니다.
- -Daraqne.cache.dir: 실행 바이너리가 저장되는 디렉터리의 경로를 지정합니다.
- -Daraqne.data.dir: 설정, 로그, 인덱스 데이터가 저장되는 디렉터리 경로를 지정합니다.
- -Daraqne.log.dir: 아라크네 코어의 운영 로그가 저장되는 디렉터리 경로를 지정합니다.
- -Daraqne.download.dir: 아라크네 번들 및 패키지 다운로드 시 사용되는 임시 디렉터리 경로를 지정합니다.
- -Daraqne.home.dir: SFTP 서버에 접속 시 각 사용자의 디렉터리가 위치할 홈 디렉터리 경로를 지정합니다.
- -Daraqne.plugin.dir: 아라크네 코어 부팅 시 설치할 플러그인 디렉터리 경로를 지정합니다. 플러그인으로 설치되는 번들은 심볼릭 이름으로 기존의 번들이 없을 때에만 설치됩니다. 번들의 버전만 다른 경우에는 설치되지 않습니다. (araqne-core 2.6.2 버전부터 지원)

1.2.4. IP 바인딩 및 포트 설정

- -Daraqne.ssh.address: SSH 서버를 바인딩할 IP를 지정합니다. 기본값은 0.0.0.0 입니다.
- -Daraqne.ssh.port: SSH 포트를 지정합니다. 기본값은 7022입니다.

- -Daraqne.telnet.address: 텔넷 서버를 바인딩할 IP를 지정합니다. 기본값은 0.0.0.0 입니다.
- -Daraqne.telnet.port: 텔넷 포트를 지정합니다. 기본값은 7004입니다.

1.2.5. 트러블슈팅

- -XX:+PrintClassHistogram: SIGQUIT 시그널을 전달하면 클래스별 메모리 사용 히스토그램을 출력합니다. 메모리 고갈 상태를 진단할 때 사용할 수 있습니다.

1.2.6. 윈도우 레지스트리 설정

아라크네 코어를 NT 서비스로 구동하는 경우에는 레지스트리의 매개변수를 추가 또는 수정할 수 있습니다.

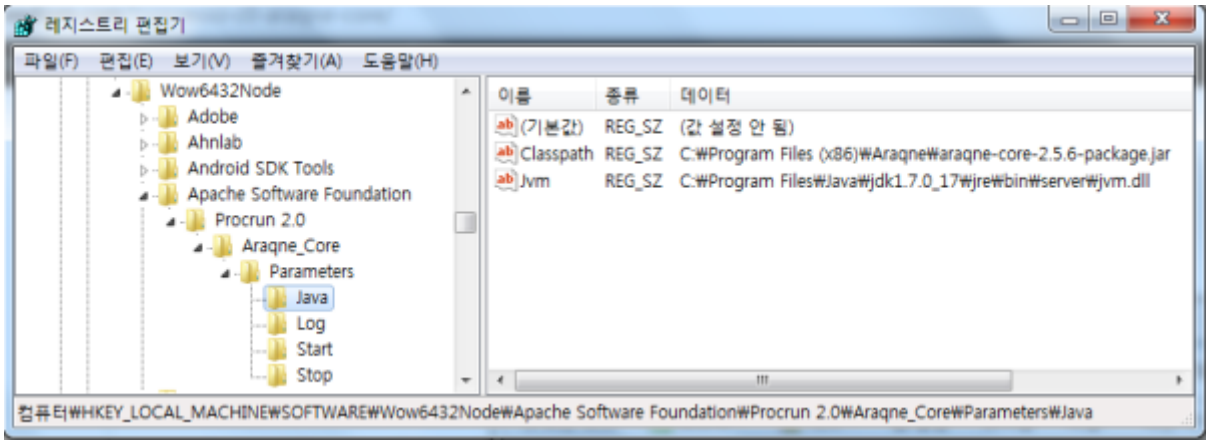

그림 1: 아라크네 코어 서비스 레지스트리 편집 화면

1.3. 웹 서버 설정

로그프레소 웹 UI와 클라이언트 라이브러리는 웹소켓 프로토콜을 사용하여 웹서버와 통신합니다. CLI를 통하여 웹서버를 관리할 수 있습니다.

1.3.1. 웹 서버 포트 열기

```
araqne> httpd.open
Description

open port
```

```
Arguments

1. port: bind port (required)
2. default context: default http context (required)
```

- 가령 80포트로 웹서버를 열고 로그프레소 웹 UI 컨텍스트를 바인딩하려면 아래와 같이 명령합니다.

```
araqne> httpd.open 80 webconsole
```

1.3.2. 웹 서버 포트 닫기

```
araqne> httpd.close
Description

close port

Arguments

1. listen port: bind port (required)
2. listen addr: bind address (optional)
```

- 가령 현재 열려있는 80포트를 닫으려면 아래와 같이 명령합니다.

```
araqne> httpd.close 80
```

1.3.3. 웹 서버 포트 목록 조회

- 현재 열려있는 포트 목록을 조회하려면 아래와 같이 명령합니다

```
araqne> httpd.bindings
Port Bindings
-------
/0.0.0.0:80, opened, default context: webconsole, idle timeout: 0seconds
```

1.4. 라이센스 관리

로그프레소 라이센스는 하드웨어 고유키, 사용기간, EPS (초당 로그 건수), 일별 로그량으로 구성됩니다. EPS가 제한된 상태에서 속도를 초과하게 되면 제한 속도에 맞추어 지연 처리 됩니다. 일별 원본 로그량은 바이트 단위입니다.

logpresso-license 1.5.1 이전 버전까지는 라이센스가 설치되지 않았거나, 허용된 일별 원본 로그량을 초과한 경우 로그가 저장되지 않고 버려집니다. 1.5.1 버전부터는 아래와 같이 라이센스가 동작합니다:

- 라이센스 미설치 시 일일 원본 로그량 기준으로 500MB까지 허용
- 일일 원본 로그량을 초과한 경우, 라이센스 위반 경고 발생
- 최근 30일간 라이센스 위반 경고가 5회 누적된 경우 라이센스 잠금
- 라이센스가 잠기면 fulltext 쿼리 시 라이센스 예외가 반환되며, table 쿼리는 아무 결과를 반환하지 않음
- 라이센스 설치 시 일일 허용 원본 로그량 이하의 이전 라이센스 위반 경고는 무시됨

1.4.1. 라이센스 상태 조회

유효한 라이센스가 설치되지 않은 상태에서는 아래와 같이 no license가 표기됩니다.

```
araqne> logpresso.licenseStatus
no license
```

유효한 라이센스가 설치된 상태에서는 아래와 같이 라이센스 정보가 표시됩니다. rate limit과 volume limit이 null이면 무제한입니다.

```
araqne> logpresso.licenseStatus
duration=2013-03-02~2015-03-02, issued=2013-03-15, rate limit=null, volume limit=null
```

1.4.2. 하드웨어 고유키 조회

라이센스를 발급받으려면 하드웨어 고유키를 보내주셔야 합니다. 아래의 명령으로 하드웨어 식별 고유키를 확인합니다. 네트워크 접속 상태에 따라 하드웨어 고유키가 변경될 수 있습니다.

```
araqne> logpresso.hardwareKey
hardware key=94a59683060b65fda6ae7a195023e28d42d2121d
```

1.4.3. 설치된 라이센스 목록 조회

현재 시스템에 설치되어 있는 모든 라이센스를 조회합니다. 유효한 라이센스에는 * 표시됩니다.

```
araqne> logpresso.installedLicenses
Installed Licenses
-------
[ ] duration=2013-04-08~2113-04-08, issued=2013-04-09, rate limit=null, volume limit=null
[*] duration=2013-03-02~2015-03-02, issued=2013-03-15, rate limit=null, volume limit=null
[ ] duration=2013-05-09~2013-12-31, issued=2013-05-09, rate limit=null, volume limit=null
```

1.4.4. 라이센스 설치

하드웨어 고유키와 일치하는 라이센스가 올바르게 설치된 경우에는 아래와 같이 installed 메시지가 표시됩니다.

```
araqne> logpresso.installLicense demo.lic
duration=2013-03-01~2013-03-31, issued=2013-03-02, rate limit=100000, volume limit=null
installed
```

하드웨어 고유키가 일치하지 않는 경우 아래와 같이 오류 메시지가 표시됩니다.

```
araqne> logpresso.installLicense invalid-license.lic
license hardware key mismatch
```

HA 구성 시나리오의 경우 하드웨어 고유키가 일치하지 않아도 라이센스를 미리 설치해두어야 하는 경우가 있습니다. 이 때에는,

```
araqne> logpresso.installLicense mismatch.lic true
```

와 같이 두번째 인자를 true로 주면 라이센스가 당장 유효하게 동작하지는 않지만 설치됩니다. 이후 다른 머신에서 해당 데이터 파티션을 사용하여 로그프레소가 부팅되면 라이센스가 유효하게 기능합니다. (logpresso-license 1.4.2부터 가능)

1.4.5. 라이센스 삭제

이미 설치된 라이센스를 삭제하려면 아래와 같이 명령합니다.

```
araqne> logpresso.uninstallLicense 94a59683060b65fda6ae7a195023e28d42d2121d
uninstalled
```

1.4.6. 일별 로그 기록량 조회

최근 10일간 기록된 일일 원본 데이터량을 조회합니다.

```
araqne> logpresso.licenseUsages
License Usages
------
[2013-08-13] 2,000,000 logs, 758,000,000 bytes
[2013-08-12] 2,000,000 logs, 758,000,000 bytes
```

1.4.7. 라이센스 위반 경고 조회

최근 10일간 발생한 라이센스 위반 경고를 조회합니다.

```
araqne> logpresso.licenseAlerts
License Alerts
------
[2013-08-12] 2,002,000 logs, 758,600,000 bytes
```

1.5. 성능 측정

로그프레소 라이센스를 설치한 후 해당 머신에서의 성능을 측정할 수 있습니다.

1.5.1. 로그스토리지 성능 테스트

logstorage.benchmark 명령어를 사용하여 스토리지 엔진의 로그 쓰기 및 읽기 성능을 측정할 수 있습니다. 이 때 사용되는 샘플 웹로그는 다음과 같습니다. (약 270바이트)

```
2011-08-22 17:30:23 Google 111.222.33.44 GET /search q=cache:xgLxoOQBOoIJ:araqne.org
/+araqne&cd=1&hl=en&ct=clnk&source=www.google.com 80 - 123.234.34.45 Mozilla/5.0
(Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko)
Chrome/13.0.782.112 Safari/535.1 404 0 3

araqne> logstorage.benchmark
=== Test #1 ===
text(write): 1000000 log/1136 ms (880281 logs/s)
```

```
text(read): 1000000 log/1134 ms (881834 logs/s)
map(write): 1000000 log/2464 ms (405844 logs/s)
map(read): 1000000 log/4997 ms (200120 logs/s)
```

1.5.2. 실시간 인덱싱 성능 테스트

logpresso.benchmark 명령어를 사용하여 실시간 인덱싱 성능을 측정할 수 있습니다. 이 명령어는 인자로 임포트 대상 텍스트 파일을 받습니다.

```
araqne> logpresso.benchmark
Description

index generation benchmark

Arguments

1. file path: text file or zip file path (required)
2. zip entry: text file entry in zip file (optional)
3. UID: unique id (optional)
4. offset: skip offset (optional)
5. limit: load limit count (optional)
```

ZIP 파일에서 지정된 텍스트 로그 파일 엔트리를 로딩하면서 소요 시간 및 속도를 측정합니다. 아래 예시는 iis.zip 파일로 압축된 iis.txt 로그 파일의 199만 건을 로딩합니다.

```
araqne> logpresso.benchmark iis.zip iis.txt
creating index_benchmark table…
creating fulltext index…
loaded 10000
loaded 20000
…

loaded 1999194 logs in 11814 ms, 169222 logs/sec
dropping index_benchmark table…
done
```

2장. 클라이언트 도구

2.1. 자바 클라이언트 콘솔

로그프레소 자바 클라이언트 라이브러리 패키지는 콘솔 모드를 포함하고 있습니다. 로그프레소 자바 클라이언트 콘솔을 사용하여 원격 호스트에서 각종 관리 작업이나 쿼리를 실행할 수 있습니다.

최신 버전 다운로드

- 0.8.4 버전 (2013년 11월 9일자)

원샷 쿼리

-e "쿼리문자열" 스위치를 이용하면 인터랙티브 콘솔로 들어가지 않고 쉘에서 즉시 쿼리를 실행할 수 있습니다. 표준 출력을 리다이렉트하여 파일에 저장하면 BI 솔루션 등 타 시스템과 쉽게 연동할 수 있습니다. 아래 스위치를 지정할 수 있습니다:

- [필수] -h: 로그프레소 서버가 설치된 도메인 혹은 IP 주소를 입력합니다.
- [필수] -P: 로그프레소 서버의 웹 포트 번호를 입력합니다.
- [필수] -u: 계정 이름을 입력합니다.
- [필수] -p: 암호를 입력합니다. 빈 암호인 경우 -p ""로 입력합니다.
- [선택] -c: CSV 파일 포맷으로 출력하며, 출력할 필드 이름들을 쉼표로 구분하여 순서대로 나열합니다.
- [선택] -f: -e 스위치 뒤에 쿼리 문자열을 입력하지 않고 쿼리가 저장된 파일을 참조할 때 사용합니다. 가령 -f query.txt 를 지정하면 query.txt 파일에 저장된 쿼리 문자열을 사용합니다. 이 때 파일 내용 중 #으로 시작하는 줄은 주석으로 무시됩니다.

JSON 출력 예시

```
$ java -jar araqne-logdb-client-VERSION-package.jar
-e "logdb tables | join table [ logdb logdisk | stats sum(disk_usage) as usage by table
  | sort limit=2 usage]" -h "localhost" -P "80" -u "araqne" -p "PASSWORD"


{usage=300, table=test}
{usage=358, table=test2}
```

CSV 출력 예시

```
$ java -jar araqne-logdb-client-VERSION-package.jar
 -e "logdb tables | join table [ logdb logdisk | stats sum(disk_usage) as usage by table
 | sort limit=2 usage]" -h "localhost" -P "80" -u "araqne" -p "PASSWORD" -c "table, usage


"test","300"
"test2","358"
```

2.2. 데이터 익스포터

로그프레소 데이터 익스포터를 사용하여 로그프레소 전용 압축 파일 포맷으로 된 데이터 파일로부터 원본 데이터를 추출할 수 있습니다.

최신 버전 다운로드

- 0.2.0 버전 (2013년 12월 11일자)

스위치

- [선택] -F : 출력 형식을 지정합니다. TXT, CSV, JSON 형식을 지원하며 입력하지 않을 경우 TXT로 지정됩니다.
- [선택] -c : 출력할 컬럼을 지정합니다. 지정하지 않을 경우 line 컬럼의 데이터만 출력됩니다.
- [선택] -k : 암호화 된 경우 PKCS#12 파일과 암호를 지정합니다. "PFX 파일경로,암호" 형식으로 입력합니다.
- [선택] -l : 출력할 로그의 최대 갯수를 지정합니다. 지정하지 않을 경우 모든 로그가 출력됩니다.
- [선택] -d : 출력할 경로를 지정합니다. 지정하지 않을 경우 실행 위치에 저장됩니다.
- [선택] -z : -z 옵션이 있을 경우 출력 파일을 gz 형식으로 압축합니다.
- [선택] -O : -O 옵션이 있을 경우 -d, -z 옵션을 무시하며 표준 출력으로 결과를 표시합니다.
- [필수] 로그를 추출할 .dat 파일을 지정합니다.

1) CSV 파일 포맷, _time과 line 필드, 최대 5개 행, 표준 출력으로 설정

```
$ java -jar araqne-logstorage-exporter-0.1.0-package.jar
  -F csv -c _time,line -l 5 -O "D:\2013-11-26.dat"
```

```
"_time","line"
"Tue Nov 26 11:43:59 KST 2013","#Software: Microsoft Internet Information Services 6.0"
"Tue Nov 26 11:43:59 KST 2013","#Version: 1.0"
"Tue Nov 26 11:43:59 KST 2013","#Date: 2007-10-13 06:20:46"
```

2) JSON 파일 포맷, _time과 line 필드, 최대 5개 행, aes.pfx 인증서와 암호 1234, 표준 출력으로 설정

```
$ java -jar araqne-logstorage-exporter-0.1.0-package.jar -F json -c _time,line -l 5
  -k D:\test\aes.pfx,1234  -O "D:\2013-12-04.dat"
```

```
{"line":"#Software: Microsoft Internet Information Services 6.0","_time":"2013-12-04 13:24:19+0900"}
{"line":"#Version: 1.0","_time":"2013-12-04 13:24:19+0900"}
{"line":"#Date: 2007-10-13 06:20:46","_time":"2013-12-04 13:24:19+0900"}
```

3장. 데이터베이스 관리

로그프레소 DB 콘솔에 로그인을 한 후에 데이터베이스 관리 및 쿼리를 수행할 수 있습니다.

3.1. 계정 관리

3.1.1. 로그인

logdb.console 명령을 사용하여 DB 콘솔에 로그인 할 수 있습니다.

```
araqne> logdb.console
Description
open console

Arguments

1. login name: db account name (required)
```

예를 들어, 기본으로 설치되는 관리자 DB 계정인 araqne로 아래와 같이 로그인 가능합니다. 초기 설치 시에는 암호가 없습니다.

```
araqne> logdb.console araqne
password?
Araqne LogDB Console
Type "help" for more information
araqne@logdb>
```

3.1.2. 로그아웃

콘솔에 로그인 한 상태에서 Ctrl-C를 누르거나, quit를 치면 로그아웃하고 빠져나갑니다. 로그아웃을 하게 되면 명시적으로 쿼리를 취소하지 않더라도 포어그라운드 상태에 있는 모든 쿼리가 자동으로 중지 및 삭제됩니다.

예시1) Ctrl-C

```
araqne@logdb> interrupted
logout
```

예시2) quit 명령

```
araqne@logdb> quit
logout
```

3.1.3. DB 계정 생성

관리자 권한을 가진 계정으로 로그인 한 상태라면, 새 DB 계정을 생성할 수 있습니다. 새로 생성된 계정은 초기에 일반 테이블 조회가 불가능하고, 별도로 권한을 부여해줘야만 합니다.

```
araqne@logdb> create_account
Usage: create_account [login name]
```

3.1.4 DB 계정 삭제

관리자 권한을 가진 계정으로 로그인 한 상태라면, DB 계정을 삭제할 수 있습니다.

```
araqne@logdb> remove_account
Usage: remove_account [login name]
```

3.1.5. DB 계정 암호 변경

관리자 계정은 임의의 계정의 암호를 변경할 수 있고, 일반 계정은 자기 계정의 암호를 변경할 수 있습니다.

예시1) 관리자 계정

```
araqne@logdb> passwd test
Changing password for user test
New password:
Retype new password:
password changed
```

예시2) 일반 계정

```
test@logdb> passwd test
Changing password for user test
(current) password:
New password:
Retype new password:
password changed
```

3.1.6. 관리자 권한 부여

관리자 계정으로 로그인 한 상태라면, 다른 계정에 관리자 권한을 부여할 수 있습니다.

```
araqne@logdb> grant_admin
Usage: grant_admin [login name]
```

예시)

```
araqne@logdb> grant_admin subadmin
granted
```

3.1.7. 관리자 권한 회수

관리자 권한으로 로그인 한 상태라면, 다른 계정의 관리자 권한을 회수할 수 있습니다. 현재 로그인 된 세션의 계정 관리자 권한을 회수할 수는 없습니다.

```
araqne@logdb> revoke_admin
Usage: revoke_admin [login name]
```

예시)

```
araqne@logdb> revoke_admin subadmin
revoked
```

3.1.8. 테이블 읽기 권한 부여

일반 계정을 생성한 후에 테이블 읽기 권한을 부여해야 합니다.

```
araqne@logdb> grant
Usage: grant [login name] [table name]
```

3.1.9. 테이블 읽기 권한 회수

계정의 테이블 읽기 권한을 회수합니다.

```
araqne@logdb> revoke
Usage: revoke [login name] [table name]
```

3.2. 쿼리

3.2.1. 쿼리 실행

쿼리를 실행하고 완료될 때까지 대기한 다음, 전체 쿼리 결과를 콘솔에 출력합니다. 아래 예시는 쿼리 로그 테이블에서 쿼리 로그를 조회하는 예제입니다.

```
araqne@logdb> query table limit=1 araqne_query_logs
{_id=3, _table=araqne_query_logs, _time=Wed Jun 05 01:25:21 KST 2013, cancelled=false,
 duration=0, eof_at=Wed Jun 05 01:25:21 KST 2013, login_name=araqne, query_id=3,
 query_string=table limit=1 araqne_query_logs, rows=1,
 start_at=Wed Jun 05 01:25:21 KST 2013}
total 1 rows, elapsed 0.1s
```

3.2.2. 실행 중인 쿼리 목록 조회

현재 세션에서 실행 중인 모든 쿼리 목록을 조회합니다. 동일 DB 계정이라도 다른 세션의 쿼리는 조회되지 않습니다. 오른쪽 끝의 숫자는 쿼리 결과의 갯수를 가리킵니다.

```
araqne@logdb> queries
Log Queries
-----
[5:araqne at 2013-06-05 01:26] table araqne_query_logs | stats count => 1
```

3.2.3. 쿼리 생성

결과 건수가 매우 많거나 오래 걸리는 쿼리의 경우 query 명령어 대신, 직접 쿼리를 생성 및 시작시키고 쿼리의 실행 상태를 모니터링하며 결과의 일부만 출력할 수 있습니다. 쿼리 문법 오류 없이 정상적으로 쿼리가 생성되면 쿼리 ID가 출력됩니다.

```
araqne@logdb> create_query table araqne_query_logs | stats count
created query 5
```

3.2.4. 쿼리 시작

쿼리 ID에 해당되는 쿼리를 시작시킵니다.

```
araqne@logdb> start_query 5
started query 5
```

3.2.5. 쿼리 결과 조회

쿼리가 완전히 완료되기 전이라도, 부분적인 쿼리 결과를 조회할 수 있습니다.

```
araqne@logdb> fetch
Usage: fetch [query id] [offset] [limit]
```

예시)

```
araqne@logdb> fetch 5 0 1
{count=2127}
```

3.2.6. 쿼리 중지

실행 중인 쿼리를 중지시킵니다. 해당 시점까지 조회된 쿼리 결과들은 삭제하기 전까지 유지됩니다.

예시)

```
araqne@logdb> stop_query 5
stopped
```

3.2.7. 쿼리 삭제

지정된 쿼리 ID를 가진 쿼리를 삭제합니다. 디스크의 임시 쿼리 결과 파일은 삭제됩니다.

```
araqne@logdb> remove_query
Usage: remove_query [query id]
```

예시)

```
araqne@logdb> remove_query 5
removed query 5
```

3.2.8.쿼리 백그라운드 전환

장기간 수행되는 쿼리의 경우 백그라운드로 동작시킬 수 있습니다. 백그라운드로 동작하는 쿼리는 로그아웃을 하더라도 그대로 동작하게 됩니다.

```
araqne@logdb> bg
Usage: bg [query id]
```

예시)

```
araqne@logdb> create_query table iis
created query 6
araqne@logdb> start_query 6
started query 6
araqne@logdb> bg 6
run as a background task
```

```
araqne@logdb> queries
Log Queries
-----
[6:araqne at 2013-06-05 01:37 (bg)] table iis => 450901
```

3.2.9. 쿼리 포어그라운드 전환

동일 계정에서 수행했던 백그라운드 쿼리는 포어그라운드로 다시 전환하고 쿼리 결과를 조회할 수 있습니다.

```
araqne@logdb> fg
Usage: fg [query id]
```

예시)

```
araqne@logdb> queries
Log Queries
-----
[6:araqne at 2013-06-05 01:37 (bg)] table iis => 1999194
araqne@logdb> fg 6
run in the foreground
araqne@logdb> fetch 6 0 1
{_id=1999194, _table=iis, _time=Wed May 29 13:51:34 KST 2013,
line=2007-10-29 00:16:27 W3SVC1 123.223.21.233 GET /solution/1.982/asp/strawlv01982_msg.asp
 t=1&m=0019D1EFEDD4 80 - 125.283.236.257 UtilMind+HTTPGet 200 0 0}
```

3.2.10. 모든 쿼리 삭제

현재 세션에서 실행 중인 모든 쿼리를 중지 및 삭제합니다.

예시)

```
araqne@logdb> remove_all_queries
removed query 6
cleared all queries
```

3.3. 테이블 관리

3.3.1. 테이블 목록 조회

전체 테이블 목록을 조회합니다. 설치 직후에는 기본으로 4개의 테이블이 있습니다. logstorage.tables 명령의 매개변수는 아래와 같습니다:

- [선택] 필터링 문자열 : 필터링 문자열을 입력하면 해당 문자열을 포함하는 테이블 이름만 표시합니다. (araqne-logstorage 2.2.3부터 지원)

예시)

```
araqne> logstorage.tables
Tables
---
[1] logpresso-log-trend: 2013-06-04
[2] alerts: none
[3] logpresso-alert-trend: none
[4] araqne_query_logs: 2013-06-05
```

숫자는 테이블 ID를 의미하고 파일시스템에 기록되는 디렉터리 이름과 일치합니다. 날짜는 가장 마지막으로 저장된 로그 일자를 의미합니다. 테이블이 비어있으면 none이 출력됩니다.

3.3.2. 테이블 상세 조회

- 테이블 메타데이터, 데이터 경로, 디스크 사용량을 확인할 수 있습니다.

```
araqne> logstorage.table iis
Table iis

Table Metadata
------
_filetype=v3p

Storage information
-------
Data path: D:\demo\data\araqne-logstorage\log\14
Consumption: 39,915,947 bytes
```

3.3.3. 테이블 생성

- 새로운 테이블을 생성합니다.

```
araqne@bombom sds> logstorage.createTable
Description

create new table

Arguments

1. name: log table name (required)
2. type: log file type (v1, v2, etc) (required)
```

로그프레소 스토리지 엔진을 이용하려면 type으로 v3p를 지정해야 합니다. 매개변수 설명에는 없으나 key=value 형태로 뒤에 이어붙이면 테이블 메타데이터를 지정할 수 있습니다. 이를 이용하여 기본 압축 방식이나 테이블 데이터 파티션을 변경할 수 있습니다.

- 예시) v3p 스토리지 엔진과 snappy 압축을 사용하는 demo 테이블 생성

```
araqne> logstorage.createTable demo v3p compression=snappy
table created
```

- 예시) /data1 경로를 테이블 데이터 파티션으로 사용하는 demo 테이블 생성 (araqne-logstorage 2.0.2부터 지원)

```
araqne> logstorage.createTable demo v3p base_path=/data1
```

3.3.4. 테이블 삭제

```
araqne> logstorage.dropTable
Description

drop log table

Arguments

1. name: log table name (required)
```

- 예시) demo 테이블 삭제

```
araqne> logstorage.dropTable demo
table dropped
```

3.3.5. 테이블 데이터 파기

- 특정 기간의 로그 데이터를 지정하여 파기할 수 있습니다.

```
araqne> logstorage.purge
Description

purge log files between specified days

Arguments

1. table name: table name (required)
2. from: yyyyMMdd (required)
3. to: yyyyMMdd (required)
```

- 예시) 2013년 5월 29일 iis 테이블 데이터 파기

```
araqne> logstorage.purge iis 20130529 20130529
purging table iis day 2013-05-29
completed
```

파기된 일자별로 purging table 메시지가 출력됩니다.

3.3.6. 테이블 스키마 설정

araqne-logstorage 2.5.1 버전부터 지원

테이블에 필드 정의 목록을 설정하여 쿼리 UI의 자동완성을 지원할 수 있습니다. 로그프레소는 비정형 데이터를 처리하므로, 스키마 설정은 참고용일 뿐 실제 데이터의 입출력을 강제하지 않는다는 점을 유의하시기 바랍니다.

- 필드 정의 목록 설정

```
araqne> logstorage.setFields
Description

set table fields

Arguments

1. table name: table name (required)
```

- 필드 정의 목록 삭제

```
araqne> logstorage.unsetFields
Description

unset table fields

Arguments

1. table name: table name (required)
```

- 예시) iis 테이블에 스키마 설정

```
araqne> logstorage.setFields iis
Use ‘name type(length)’ format. Length can be omitted.
Allowed types: string, short, int, long, float, double, date, and bool
Enter empty line to finish.
field definition? line string
field definition?
schema changed

araqne> logstorage.table iis
Table iis

Table Metadata
------
type=iis
```

```
_filetype=v3p

Table Schema
-----
line string
(   )
```

3.4. 데이터 보안

지원 버전: araqne-logstorage 2.2.0, araqne-logdb 1.5.0, logpresso-logstorage 2.6.0 이상

로그프레소는 데이터 원본을 실시간 압축하는 동시에 암호화 및 다이제스트 생성을 통하여 기밀성과 무결성을 보장합니다. 데이터 암호화를 설정하려면 먼저 암호화 프로파일을 만든 후, 테이블 생성 시 crypto 옵션으로 암호화 프로파일을 지정합니다.

로그프레소는 PKCS#12 인증서 파일에서 공개키와 비밀키를 읽어들여 사용합니다. 일자별 데이터 원본 파일(.dat)을 생성할 때 일자별 암호키 파일 (.key)을 생성합니다. 암호키 파일에는 해당 일자의 데이터 원본 파일을 암호화 및 HMAC 다이제스트 하는데 사용한 키, 암호 알고리즘, 다이제스트 알고리즘에 대한 정보가 포함됩니다. 암호키 파일은 PKCS#12 공개키를 사용하여 암호화되고 비밀키를 사용하여 복호화되므로, 사용자가 설정한 인증서가 없으면 로그 원본 데이터 파일이 유출되더라도 임의로 내용을 읽을 수 없습니다.

3.4.1. 암호화 프로파일 목록 조회

```
araqne> logstorage.cryptoProfiles
Crypto Profiles
-----------------
name=hashonly, cipher=null, digest=HmacSHA256, path=D:\crypto.pfx, password=PASSWORD
name=aes, cipher=AES/CBC/PKCS5Padding, digest=HmacSHA256, path=D:\crypto.pfx, password=PASSWORD
```

현재 설정된 암호화 프로파일 목록을 조회할 수 있습니다.

3.4.2. 암호화 프로파일 추가

예시) aes 이름으로 암호화 프로파일 추가

```
araqne> logstorage.createCryptoProfile aes
pkcs12 path? D:\crypto.pfx
```

```
pkcs12 password? PASSWORD
cipher algorithm? AES/CBC/PKCS5Padding
digest algorithm? HmacSHA256
```

유효기간이 만료된 PKCS#12 파일이라도 데이터 암호화 하는데 문제 없이 사용할 수 있습니다. PKCS#12 비밀키를 읽어들이는데 필요한 암호를 입력하고, 이어서 암호 알고리즘과 다이제스트 알고리즘을 입력합니다. 암호 알고리즘을 입력하지 않으면 데이터 암호화 없이 다이제스트만 생성하게 됩니다.

3.4.3. 암호화 프로파일 삭제

```
araqne> logstorage.removeCryptoProfile aes
removed
```

암호화 프로파일 이름을 인자로 전달하여 삭제할 수 있습니다. 테이블에서 사용 중인 암호화 프로파일을 삭제하게 되면, 해당 테이블은 읽기 및 쓰기가 불가능하게 되므로 주의하시기 바랍니다. 실수로 암호화 프로파일을 삭제한 경우에는 동일하게 다시 설정하여 복구할 수 있습니다.

3.4.4. 테이블 암호화 설정

아래 예시와 같이 테이블 생성 시 암호화 프로파일 이름을 crypto 옵션으로 지정합니다:

```
araqne> logstorage.createTable iis v3p crypto=aes
table created
```

3.4.5. 무결성 검증

logcheck 쿼리 커맨드를 사용하여 데이터 원본의 무결성을 검증할 수 있습니다. from과 to 옵션을 사용하여 검사 대상 기간을 지정할 수 있습니다:

```
logcheck from=yyyyMMdd to=yyyyMMdd table1 [, table2, table3, ... ]
```

```
araqne@logdb> query logcheck iis
{block_id=999, day=Tue Aug 06 00:00:00 KST 2013,
hash=29ae97b40f7008283e7c6f0be665621b489d53862dec326ba4c3ab0a25b3a2f9,
signature=29077f4653adb072120bd620edce149c7e362b2d9f00c1c6eab29d824cb08d87, table=iis}
total 1 rows, elapsed 0.9s
```

데이터 원본이 위조 혹은 변조된 경우, 위와 같이 무결성이 깨진 데이터 블럭에 대한 정보가 표시됩니다. signature는 초기 데이터 기록 시의 해시 값이며, hash는 현재 데이터를 기준으로 계산된 해시 값입니다.

3.5. 데이터 임포트

아래의 명령어를 사용하여 테이블에 텍스트 로그 파일을 임포트 할 수 있습니다.

3.5.1. 텍스트 파일 임포트

```
araqne> logstorage.importTextFile
Description

import text log file

Arguments

1. table name: table name (required)
2. file path: text log file path (required)
3. offset: skip offset (optional)
4. limit: load limit count (optional)
```

지정된 경로의 텍스트 로그 파일을 줄 단위로 읽어서 지정된 테이블에 입력합니다. offset을 지정하면 해당 갯수만큼 건너뛰고, limit을 지정하면 해당 갯수만큼만 입력합니다. .gz 형식의 스트림 압축된 로그 파일도 이 명령어를 이용하여 로그를 읽어들일 수 있습니다.

예시)

```
araqne> logstorage.importTextFile sample iis.txt
…
loaded 1980000
loaded 1990000
loaded 1999194 logs in 11959 ms, 167170 logs/sec
```

3.5.2. ZIP 파일 임포트

```
araqne> logstorage.importZipFile
Description

import zipped text log file
```

```
Arguments

1. table name: table name (required)
2. zip file path: zip file path (required)
3. entry path: zip entry of text log file path (required)
4. offset: skip offset (optional)
5. limit: load limit count (optional)
```

ZIP 파일의 특정 텍스트 로그 파일 엔트리를 지정하여 줄 단위로 로그를 임포트합니다.

예시) iis.txt를 압축한 iis.zip 파일의 경우

```
araqne> logstorage.importZipFile sample iis.zip iis.txt
…
loaded 1990000
loaded 1999194 logs in 9368 ms, 213406 logs/sec
```

3.6. 파서 설정

로그프레소는 원본 로그를 그대로 저장하고 쿼리 시간에 파싱을 수행합니다.

3.6.1. 파서 유형 목록 조회

설치되는 번들에 따라서 내장된 파서 구성이 달라질 수 있지만 기본적으로는 아래의 파서를 지원합니다. 각 파서는 서로 다른 설정을 요구합니다.

```
araqne> logapi.parserFactories
Log Parser Factories
--------
regex
fortigate
delimiter
mf2
netscreen-isg
secureworks
paloalto
tippingpoint-sms
```

```
srx
spectraguard
weguardia
defensepro
nxg
trusguard
snort
neobox
```

3.6.2. 파서 관리

(이하 내용은 araqne-log-api 2.2.0부터 유효합니다.)

파서는 각기 다른 설정으로 구성할 수 있습니다. 가령 구분자 파서의 경우 구분자, 필드 이름 목록을 다르게 구성하게 됩니다.

```
araqne> logapi.createParser
Description

create parser profile

Arguments

1. profile name: parser profile name (required)
2. factory name: parser factory name (required)
```

예시) 세미콜론으로 구분되고 순차적으로 A, B, C 필드 이름을 부여하는 delimiter-example 파서 생성

```
araqne> logapi.createParser delimiter-example delimiter
delimiter (required)? ;
column headers (optional)? A,B,C
delimiter target field (optional)? line
include delimiter target (optional)?
created
```

생성된 파서 목록은 아래와 같이 조회할 수 있습니다.

```
araqne> logapi.parsers
Log Parser Profiles
-------
name=delimiter-example, factory=delimiter, configs={delimiter=;,
delimiter_target=line, column_headers=A,B,C}
```

파서를 삭제하려면 아래와 같이 합니다.

```
araqne> logapi.removeParser
Description

remove parser profile

Arguments

1. profile name: profile name (required)
```

예시) delimiter-example 이름의 파서를 삭제

```
araqne> logapi.removeParser delimiter-example
removed
```

3.6.3. 테이블 파서 설정

테이블을 조회할 때 항상 기본으로 지정된 파서를 이용하여 원본 로그를 파싱한 결과를 출력하도록 설정할 수 있습니다.

예시) mf2 파서를 mf2_syslogs 테이블의 기본 파서로 지정

```
araqne> logstorage.table mf2_syslogs parser mf2
```

3.6.4. 구분자 파서 설정

구분자(delimiter)는 가장 흔히 사용되는 파서입니다. 구분자는 4개의 설정 옵션이 있습니다:

- [필수] delimiter: 구분자로 사용될 문자
- [선택] delimiter_target: 파싱 대상 필드, 지정하지 않으면 line 필드를 파싱합니다.

- [선택] column_headers: 구분자로 파싱된 각 항목에 대해 부여할 필드 이름 목록 (쉼표로 구분). 지정하지 않으면 column0, column1, … 식으로 필드 이름이 부여됩니다.
- [선택] include_delimiter_target: 파싱 대상 필드를 파싱 결과에 포함할 것인지 설정합니다. true 혹은 false 지정. 기본값은 false 입니다.

예시) 공백 기준으로 구분된 iis 로그 파싱

```
araqne@logdb> query table limit=1 iis
{_id=1999194, _table=iis, _time=Wed Jun 05 10:34:51 KST 2013,
 line=2007-10-29 00:16:27 W3SVC1 123.223.21.233 GET /solution/1.982
 /asp/strawlv01982_msg.asp t=1&m=0019D1EFEDD4 80 - 125.283.236.257
 UtilMind+HTTPGet 200 0 0}
total 1 rows, elapsed 0.1s

araqne> logapi.createParser iis_parser delimiter
delimiter (required)?
column headers (optional)? date,time,s-sitename,s-ip,cs-method,cs-uri-stem,
cs-uri-query,s-port,cs-username,c-ip,user-agent,sc-status,sc-su status,sc-win32-status
delimiter target field (optional)? line
include delimiter target (optional)? false
created

araqne> logstorage.table iis parser iis_parser
set parser to iis_parser

araqne@logdb> query table limit=1 iis
{_id=1999194, _table=iis, _time=Wed Jun 05 10:34:51 KST 2013, c-ip=125.283.236.257,
cs-method=GET, cs-uri-query=t=1&m=0019D1EFEDD4, cs-uri-stem=/solution/1.982/asp
/strawlv01982_msg.asp, cs-username=-, date=2007-10-29, s-ip=123.223.21.233,
s-port=80, s-sitename=W3SVC1, sc-status=200, sc-substatus=0, sc-win32-status=0,
time=00:16:27, user-agent=UtilMind+HTTPGet}
total 1 rows, elapsed 0.1s
```

3.6.5. 정규표현식 파서 설정

정규표현식(regex) 파서는 정교한 필드 추출이 가능합니다. 정규표현식 파서는 2개의 설정 옵션이 있습니다:

- [필수] regex: 필드 추출에 사용될 정규표현식
- [선택] field: 정규표현식을 적용할 필드. 미설정 시 기본값은 line 입니다.
- [선택] include_original_field: 원본 필드 포함 여부. 미설정 시 원본 필드 값은 포함되지 않습니다. (araqne-log-api 2.4.1부터 지원)

정규표현식은 필드 이름을 포함하도록 만들어진 확장 문법을 사용합니다. 정규표현식 그룹에 ? 형식으로 끼워넣으면, 해당 정규표현식 그룹에 맞는 문자열이 지정된 필드 이름으로 추출됩니다.

예시) IIS 웹 로그에서 정규표현식으로 URL 및 쿼리스트링을 추출하는 설정

```
araqne> logapi.createParser iis_regex regex
regex (required)? (GET|POST) (?<url>[^ ]*) (?<querystring>[^ ]*)
field (optional)? line
created

araqne> logstorage.table iis parser iis_regex
set parser to iis_regex

araqne@logdb> query table limit=1 iis
{_id=1999194, _table=iis, _time=Wed Jun 05 10:34:51 KST 2013,
 querystring=t=1&m=0019D1EFEDD4, url=/solution/1.982/asp/strawlv01982_msg.asp}
```

3.6.6. WELF 파서

WELF (welf) 파서는 WELF (WebTrends Enhanced Log Format) 형식의 데이터를 파싱합니다. 별도의 설정 매개변수는 없으며 아래와 같은 키/값 쌍의 나열을 파싱할 수 있습니다 (araqne-log-api 2.6.4 버전부터 지원):

- key=value
- key="value with whitespace"
- key value

예시) TESS TMS WELF 로그 파싱

```
araqne> logapi.createParser welf welf
created

araqne@logdb> query table limit=1 tms
```

```
{_host=kaits, _id=5716, _table=tms, _time=Sat Oct 05 14:07:32 KST 2013,
facility=-1, line=Health info SensorName="demo" SensorIp="192.168.0.1"
Connection=1 Time="2013/10/05 14:10:20" CPU_Usage="4 %" MEMORY_Usage="19 %"
HDD_Usage="0 %" PROCESS_Cnt=96 EventPerSecond="0.00 " SessionPerSecond="5.87 K"
PacketLossRate="0.00 %" TotalTraffic="6.46 M" MaliciousTraffic="0.00 (0.00)"
TotalTrafficPps="2.87 K" MaliciousTrafficPps="0.00 (0.00)", severity=-1}

araqne@logdb> query table limit=1 tms | parse welf
{CPU_Usage=4 %, Connection=1, EventPerSecond=0.00 , HDD_Usage=0 %,
Health=info, MEMORY_Usage=19 %, MaliciousTraffic=0.00 (0.00),
MaliciousTrafficPps=0.00 (0.00), PROCESS_Cnt=96, PacketLossRate=0.00 %,
SensorIp=192.168.0.1, SensorName=demo, SessionPerSecond=5.87 K,
Time=2013/10/05 14:10:20, TotalTraffic=6.46 M, TotalTrafficPps=2.87 K,
_id=5716, _table=tms, _time=Sat Oct 05 14:07:32 KST 2013}
total 1 rows, elapsed 0.1s
```

3.6.7. 필드 변환 파서

필드 변환 파서는 필드 이름을 변경하려고 할 때 사용합니다. 1개의 설정 옵션이 있습니다:

- [필수] mappings: 필드 이름 변환 설정. "원본필드이름1=변경필드이름1,원본필드이름2=변경필드이름2,…" 같은 형식으로 쉼표로 구분해서 입력합니다.

예시) line 필드 이름을 text 이름으로 변경

```
araqne> logapi.createParser line2text fieldmapper
field mappings (required)? line=text
created

araqne> logstorage.table iis parser line2text
set parser to line2text

araqne@logdb> query table limit=1 iis
{_id=1999194, _table=iis, _time=Fri Jun 14 15:33:47 KST 2013,
text=2007-10-29 00:16:27 W3SVC1 123.223.21.233 GET /solution
/1.982/asp/strawlv01982_msg.asp t=1&m=0019D1EFEDD4 80 -
125.283.236.257 UtilMind+HTTPGet 200 0 0}
```

3.6.8. 체인 파서

체인 파서는 여러 개의 파서를 조합해서 파싱하려고 할 때 사용합니다. 보통 필드 변환 파서를 사용할 때는 이전 단계에 다른 파서가 붙게 됩니다. 1개의 설정 옵션이 있습니다:

- [필수] parsers: 파서 이름 목록. "파서1,파서2,파서3,.."와 같이 쉼표로 구분해서 입력합니다.

예시) weguardia 파서와 fieldmapper 파서를 조합하여 sip 필드 이름을 src_ip로, dip 필드 이름을 dst_ip로 바꿈

```
# weguardia
araqne> logapi.createParser weguardia weguardia
created

#weguardia_mapper
araqne> logapi.createParser weguardia_mapper fieldmapper
field mappings (required)? sip=src_ip,dip=dst_ip
created

#   2     weguardia_chain
araqne> logapi.createParser weguardia_chain chain
parser names (required)? weguardia,weguardia_mapper
created

# weguardia         sip, dip
araqne> logstorage.table weguardia_table parser weguardia
set parser to weguardia

araqne@logdb> query table weguardia_table
{_id=1, _table=weguardia_table, _time=Fri Jun 14 15:57:54 KST 2013,
act=537133067, category= , column24=null, count=1, date=Wed Dec 05
11:46:09 KST 2012, dip=121.189.14.140, dport=80, group_id= , iface=eth3,
logtype=1, nat_dip=121.189.14.140, nat_dport=80, nat_sip=110.92.155.254,
nat_sport=5318, note=Close[00:00:20. SF. FIN] NAT[313] R[16], oip= ,
product=arko-guro, protocol=6, rule=1, severity=4, sip=172.16.0.97,
sport=5318, type=0, usage=1698, user=null}
total 1 rows, elapsed 0.1s
```

```
# weguardia_chain       sip, dip    src_ip, dst_ip
araqne> logstorage.table weguardia_table parser weguardia_chain
set parser to weguardia_chain

araqne@logdb> query table weguardia_table
{_id=1, _table=weguardia_table, _time=Fri Jun 14 15:57:54 KST 2013,
act=537133067, category= , column24=null, count=1, date=Wed Dec 05
11:46:09 KST 2012, dport=80, dst_ip=121.189.14.140, group_id= ,
iface=eth3, logtype=1, nat_dip=121.189.14.140, nat_dport=80,
nat_sip=110.92.155.254, nat_sport=5318, note=Close[00:00:20. SF. FIN]
NAT[313] R[16], oip= , product=arko-guro, protocol=6, rule=1, severity=4,
sport=5318, src_ip=172.16.0.97, type=0, usage=1698, user=null}
```

3.6.9. 쿼리 파서

(araqne-logdb 1.6.5 버전부터 지원)

쿼리 파서는 로그 쿼리를 이용하여 로그를 파싱하려고 할 때 사용합니다. 쿼리 문법을 그대로 사용할 수 있으므로 다른 파서에 비해 상대적으로 복잡하고 정교한 파싱을 수행할 수 있습니다. 설정 항목은 다음과 같습니다:

- [필수] query: 쿼리 문자열, 입력 로그와 쿼리 결과가 일대일로 매핑되는 쿼리 커맨드를 파이프로 연결한 쿼리를 설정할 수 있습니다. 주로 eval 커맨드를 연결하여 사용합니다. stats 커맨드처럼 입력과 쿼리 결과가 일대일로 매핑되지 않는 커맨드는 사용할 수 없습니다.

예시) line의 길이를 len 필드로 추가하도록 쿼리 파서 설정

```
# addlen
araqne> logapi.createParser addlen query
Query string (required)? eval len = len(line)
created

# iis    addlen
araqne> logstorage.table iis parser addlen
set parser to addlen

#               len
```

```
araqne> logdb.console araqne
password?
Araqne LogDB Console
Type "help" for more information
araqne@logdb> query table limit=1 iis
{_id=1999194, _table=iis, _time=Fri Aug 30 22:20:39 KST 2013, len=151,
line=2007-10-29 00:16:27 W3SVC1 123.223.21.233 GET /solution/1.982/asp
/strawlv01982_msg.asp t=1&m=0019D1EFEDD4 80 - 125.283.236.257
UtilMind+HTTPGet 200 0 0}
total 1 rows, elapsed 0.1s
```

3.7. 인덱스 관리

로그프레소는 다양한 풀텍스트 인덱스 설정을 지원합니다.

3.7.1. 인덱스 토크나이저 목록 조회

인덱스 토크나이저는 원본 로그를 풀텍스트 인덱싱할 때 사용될 풀텍스트 토큰을 만드는 역할을 수행합니다.

```
araqne> logpresso.indexTokenizers
Index Tokenizers
------
[regex] token extraction using regular expression
[delimiter] delimiter index tokenizer factory
[default] delimiter ;'"`~!@#$%^&*()_+-=[]{}|\<> and whitespaces, prefix delimiter ./
[jython] jython index tokenizer script
[fixedlength] fixed length tokenizer
```

3.7.2. 인덱스 목록 조회

현재 설정되어 있는 인덱스 목록을 조회할 수 있습니다.

```
araqne> logpresso.indexes
id=4, table=iis, index=idx, period (unbound), bloomfilter=true
[lv0: 1250000, 0.0010000000474974513, lv1: 10000000,
0.019999999552965164], tokenizer=default, base path=null
```

특정 테이블에 대한 인덱스 목록을 조회하려면 테이블 이름을 매개변수로 넘깁니다.

```
araqne> logpresso.indexes iis
Index for table [iis]
-----------
id=4, table=iis, index=idx, period (unbound), bloomfilter=true
[lv0: 1250000, 0.0010000000474974513, lv1: 10000000,
0.019999999552965164], tokenizer=default, base path=null
```

3.7.3. 인덱스 상세 조회

특정 인덱스의 설정 및 현황을 조회할 수 있습니다.

예시) iis 테이블의 idx 인덱스 상세 조회

```
araqne> logpresso.index iis idx
Index Detail
------
Table Name: iis
Index Name (ID): idx (4)
Indexed Days: 2013-06-05 ~ 2013-06-05
Data Path: D:\logpresso\data\araqne-logstorage\index\14\4
Min index day: unlimited
Max index day: unlimited
Bloom Filter: true
Level 0: Capacity 1250000, Error Rate 0.0010000000474974513
Level 1: Capacity 10000000, Error Rate 0.019999999552965164
Tokenizer: default

Tokenizer Config
------
target_columns: line

Storage Consumption
-------
217,537,705 bytes
```

3.7.4. 인덱스 생성

아래 명령을 사용하여 특정 테이블에 대한 인덱스를 생성할 수 있습니다.

```
araqne> logpresso.createIndex
Description

create index

Arguments

1. table name: table name (required)
2. index name: index name (required)
```

이 때 아래와 같은 항목들을 공통적으로 설정하게 됩니다.

- 토크나이저: 원본 로그에서 인덱싱할 대상 토큰들을 추출하는 토크나이저의 이름을 지정합니다. 토크나이저에 따라 다른 설정을 추가로 입력받게 됩니다.
- 블룸필터 사용 여부: 로그 량이 많은 경우 블룸필터를 생성하면 디스크 공간을 추가로 소모하지만 높은 검색 성능을 얻을 수 있습니다.
- 인덱스 디스크 파티션: 인덱스 루트 경로를 입력합니다. 기본적으로는 $araqne.data.dir 경로 하에 생성됩니다.
- 인덱싱 대상 기간: 아래에서 과거 데이터에 대해 인덱스 빌드를 선택한 경우 어느 일자부터 인덱싱할 것인지 지정할 수 있습니다. 미래의 날짜를 지정할 경우 해당 일자 이하에 대해서는 인덱싱을 수행하지 않습니다.
- 과거 데이터에 대한 인덱스 빌드 여부: 인덱스 생성 시점 이후에 들어오는 로그만 인덱싱할 것인지, 이전에 쌓인 로그도 인덱싱 할 것인지 설정합니다. 만약 과거 데이터에 대하여 인덱싱을 선택하면 배치로 인덱스가 빌드된 다음 실시간 인덱스와 자동으로 병합됩니다.

구분자 토크나이저 구분자를 명시적으로 지정할 수 있는 경우에는 구분자 토크나이저를 사용합니다. 인덱스 되는 토큰 수가 줄어들면 더 작은 디스크 사용량과 빠른 검색을 기대할 수 있습니다.

예시) iis 테이블에서 delimiter 토크나이저를 사용하여 인덱스 생성

```
araqne> logpresso.createIndex iis delimiter_idx
tokenizer? delimiter
index tokenizer configurations..
delimiters? ,;|/=?[]&
prefix_delimiters (optional, enter to skip)?
```

```
target columns (optional, enter to skip)? line
use bloom filter (y/n)? y
index disk partition path (enter to skip)?
min day (yyyymmdd or enter to skip)?
build index for past log (y/n)? y
created index delimiter_idx for table iis
```

구분자 및 IP 토크나이저 구분자 토크나이저의 기능과 동일하나, IPv4 포맷과 일치하는 토큰을 추가로 인덱싱합니다. 가령 1.2.3.4 문자열을 구분자 및 IP 토크나이저로 자르게 되면 1, 2, 3, 4, 1.2.3.4 토큰을 인덱싱하게 됩니다. IPv4 포맷의 토큰을 추가로 인덱싱하게 되면 디스크 공간을 더 소모하지만, 블룸필터 정확도가 크게 증가하므로 고속 IP 검색을 기대할 수 있습니다. 예를 들어, "1" and "2" and "3" and "4" 로 검색할 때는 IP와 관계없는 다른 숫자도 검색될 수 있으나, "1.2.3.4" 로 검색할 때는 별도의 필터링 없이 한 번에 의도한대로 검색할 수 있습니다.

Add C class token 옵션을 true로 설정하는 경우, C 클래스 IP 토큰을 추가로 인덱싱합니다. 가령, 1.2.3.4 문자열은 1, 2, 3, 4, 1.2.3., 1.2.3.4 토큰을 인덱싱하게 됩니다. C 클래스 토큰의 마지막 .을 유의하세요.

예시) iis 테이블에서 delimiter-with-ip 토크나이저를 사용하여 인덱스 생성

```
araqne> logpresso.createIndex iis delimiter_with_ip_idx
tokenizer? delimiter-with-ip
index tokenizer configurations..
Delimiters (optional, enter to skip)?
Target fields (optional, enter to skip)? line
Add C class token (optional, enter to skip)?
use bloom filter (y/n)? y
index disk partition path (enter to skip)?
min day (yyyymmdd or enter to skip)?
build index for past log (y/n)?
created index delimiter_with_ip_idx for table iis
```

정규표현식 토크나이저 정규표현식을 사용하여 특정한 패턴의 토큰만 인덱싱 할 수 있습니다.

예시)

```
araqne> logpresso.createIndex iis regex_idx
tokenizer? regex
index tokenizer configurations..
```

```
regex? (\d+\.\d+\.\d+\.\d+)
use bloom filter (y/n)? y
index disk partition path (enter to skip)?
min day (yyyymmdd or enter to skip)?
build index for past log (y/n)? y
created index regex_idx for table iis
```

고정 길이 토크나이저 고정 길이 토크나이저는 글자 수로 된 필드 별 길이와 구분자를 입력 받아 풀텍스트 인덱싱을 수행합니다. 입력 받은 길이대로 필드를 자른 후, 구분자를 이용해 각각의 필드를 토크나이징 합니다. 로그의 길이가 입력 받은 길이의 총 합 보다 길 경우, 남은 길이를 추가 필드로 취급합니다.

예시) iis 테이블에 fixedlength 토크나이저를 사용하여 인덱스 생성

```
araqne> logpresso.createIndex iis fixedlen_idx
tokenizer? fixedlength
index tokenizer configurations..
lenghths? 10,8,4,2
delimiters (optional, enter to skip)?
prefix_delimiters (optional, enter to skip)?
target columns (optional, enter to skip)? line
use bloom filter (y/n)? y
index disk partition path (enter to skip)?
min day (yyyymmdd or enter to skip)?
build index for past log (y/n)? y
created index default_idx for table iis
```

3.7.5. 인덱스 배치 생성 모니터링

과거 데이터에 대한 배치 인덱스 빌드가 시작된 경우 아래의 명령어로 진행 상태를 확인할 수 있습니다.

예시) 전체 배치 인덱스 작업 목록 조회

```
araqne> logpresso.batchIndexTasks
Batch Indexing Tasks
--------
 table [iis] index [default_idx] indexed log count [120011] since 2013-06-05 17:29:58 (elapsed 4sec)
```

예시) iis 테이블의 default_idx 배치 인덱스 생성 작업에 대한 일자별 현황 조회

```
araqne> logpresso.batchIndexTask iis default_idx
Batch Indexing Task
-------
day=2013-06-05, logs=590058, done=false
```

3.7.6. 인덱스 삭제

기존의 인덱스를 삭제합니다. 배치 빌드 중인 인덱스 작업이 있으면 취소됩니다.

```
araqne> logpresso.dropIndex
Description

drop index

Arguments

1. table name: table name (required)
2. index name: index name (required)
```

예시) iis 테이블의 default_idx 인덱스 삭제

```
araqne> logpresso.dropIndex iis default_idx
dropped
```

3.7.7. 인덱스 기간 지정

logpresso-index 2.3.1 버전 이상 지원

특정 인덱스가 지정한 기간에 대해서만 인덱싱을 수행하도록 지정할 수 있습니다. 현재 입력이 수행중인 테이블에 대해 데이터를 보존하면서, 앞으로의 데이터에 대해 다른 토크나이저 설정을 사용하는 인덱스를 추가하고 싶을 때 사용합니다.

```
araqne> logpresso.setIndexRange idxtesttable idx1
Description

set date range of index

Arguments
```

```
1. table name: table name (required)
2. index name: index name (required)
3. min day(inclusive): min day (inclusive) (required)
4. max day(inclusive): max day (inclusive) (required)
araqne> logpresso.setIndexRange idxtesttable idx1 unlimited 2013-10-28
table name: idxtesttable, index name: idx1: setting range to unlimited ~ 2013-10-28
done.
araqne> logpresso.setIndexRange idxtesttable idx2 2013-10-29 unlimited
table name: idxtesttable, index name: idx2: setting range to 2013-10-29 ~ unlimited
done.
araqne> logpresso.indexes idxtesttable
Index for table [idxtesttable]
-----------
id=1, table=idxtesttable, index=idx1, period (unlimited ~ 2013-10-28),
bloomfilter=true, tokenizer=delimiter, base path=null
id=2, table=idxtesttable, index=idx2, period (2013-10-29 ~ unlimited),
bloomfilter=true, tokenizer=delimiter-with-ip, base path=null
```

idx1 과 idx2 의 토크나이저가 서로 다르고, 기간이 겹치지 않음에 주목하십시오. 또한 새 인덱스의 min day 값에 작업 당일 날짜 (현재 로거가 입력중인 날짜)를 설정하지 않도록 주의하십시오.

3.8. 캐시 설정

로그프레소의 검색 성능을 극대화하려면 캐시를 설정해야 합니다. 모든 캐시는 기본값으로 100MB가 지정되어 있습니다. 캐시는 다이렉트 버퍼를 사용하므로 JVM 실행 시 반드시 전체 캐시 용량보다 크게 최대 다이렉트 버퍼 크기를 설정해야 합니다. (-XX:MaxDirectMemorySize 설정 참고)

3.8.1. 인덱스 캐시 설정 조회

`logpresso.indexCacheConfigs` 명령을 사용하여 특정 인덱스 유형에 대한 설정을 조회할 수 있습니다.

```
araqne> logpresso.indexCacheConfigs bloomfilter0
Cache Configs
-----
bloomfilter0_cache_size: 15000000000
```

3.8.2. 인덱스 캐시 설정

`logpresso.indexCacheConfig type key value` 명령을 사용하여 특정 인덱스 유형에 대해 매개변수를 설정하거나 삭제할 수 있습니다.

가령, 블룸필터 레벨 0 인덱스가 최대 10,000,000,000 바이트를 캐시하게 하려면 아래와 같이 명령합니다. max_weight 를 설정하면 기존에 캐시된 모든 항목은 버려지고, 캐시가 다시 생성됩니다.

```
araqne> logpresso.indexCacheConfig bloomfilter0 max_weight 10000000000
```

inverted 인덱스가 최대 90,000,000,000 바이트를 캐시하게 하려면 아래와 같이 명령합니다. max_weight 를 설정하면 기존에 캐시된 모든 항목은 버려지고, 캐시가 다시 생성됩니다.

```
araqne> logpresso.indexCacheConfig inverted max_weight 90000000000
```

블룸필터 레벨 0 인덱스가 캐시 활동을 기록해 두도록 하려면 아래와 같이 명령합니다.

```
araqne> logpresso.indexCacheConfig bloomfilter0 record_stats true
```

따로 캐시 활동에 대한 설정을 하지 않으면 성능 저하를 막기 위해 캐시 활동을 기록해 두지 않습니다. 블룸필터 레벨 0 인덱스의 캐시 활동 기록을 끄려면 아래와 같이 명령합니다.

```
araqne> logpresso.indexCacheConfig bloomfilter0 record_stats false
```

record_stats 를 설정하면 기존에 캐시된 모든 항목은 버려지고, 캐시가 다시 생성됩니다.

3.8.3. 인덱스 캐시 상태 조회

`logpresso.indexCacheStats type` 명령을 사용하여 특정 인덱스 캐시의 상태를 확인할 수 있습니다. 역 인덱스의 캐시 상태를 확인하는 명령은 아래와 같습니다.

```
araqne> logpresso.indexCacheStats inverted
Index Cache Stats
-------
evictionCount : 7592
```

이 상태를 확인하기 위해서는, logpresso.indexCacheConfig type record_stats true 명령이 실행되어 있어야 합니다. 성능 저하를 막기 위해서 기본 설정은 캐시 활동을 기록하지 않도록 되어 있습니다.

3.8.4. 인덱스 캐시 항목 조회

logpresso.indexCacheItems type 명령으로 현재 캐시된 전체 항목을 확인할 수 있습니다.

```
araqne> logpresso.indexCacheItems inverted
Cache Items
-----
…
[weight=4604] inverted cache [index id=9, day=2013-02-06, segment=0, posting=false]
[weight=2163031] inverted cache [index id=4, day=2013-01-09, segment=6, posting=true]
```

logpresso.indexCacheItemStats type 명령으로 현재 캐시된 전체 항목의 수를 용량 별로 확인할 수 있습니다.

```
araqne> logpresso.indexCacheItemStats inverted
Index Cache Items Weight Histogram (unit: 1000000)
-------
3000000 : 111
4000000 : 860
5000000 : 1054

Total count : 2025
Total weight : 9989246654
```

어느 정도의 크기로 자를 것인지 추가 인자를 주어 지정할 수 있으며, 단위는 byte 입니다. 지정하지 않을 경우 1MB (1000000B) 단위로 잘라 통계를 계산합니다. * ~~ araqne〉 logpresso.indexCacheItemStats inverted 500000 Index Cache Items Weight Histogram (unit: 500000) ———————– 0 : 1312 500000 : 5 …

```
Total count : 5025
Total weight : 9993352746
```

```
### 3.8.5.        ###

`logpresso.indexCacheClear type`                        :

araqne> logpresso.indexCacheClear bloomfilter0
cleared
```

3.9.

.

3.9.1.

`logstorage.setParameter key value` . :

* min_free_disk_space_type: . Percentage Megabyte Percentage . Percentage
* min_free_disk_space_value: . 10 . (, 10%)
* disk_lack_action: . StopLogging RemoveOldLog StopLogging . StopLo
* log_flush_interval: . .
* log_max_idle_time: . , .

  araqne-logstorage DB global_settings .

3.9.2.

`logstorage.setEngine engine key value` . .

3.9.3. v3p

CPU . cpu .

`logstorage.setEngine v3p key value` .

* slot_count : . batch writer, heavy thread pool . , 2 .
* heavy_pool_size : heavy thread pool . CPU 75% , 2 .
* light_pool_size : light thread pool . CPU 75% , 2 .
* light_pool_queue_size : light thread pool . light_pool_size 10000 , 2 .
* light_pool_max_size : , . . light_pool_size 3 , 2
* reader_pool_size : reader thread pool . CPU 75% , 2 .

```
 * reader_pool_max_size : reader thread pool         ,                                    .    reader_pool_
 * light_reader_pool_size : light reader thread pool                .    CPU    75% ,    2  .
 * light_reader_pool_queue_size : light reader thread pool            .    light_reader_pool_size  10000
 * light_reader_pool_max_size :         ,              .    light_reader_pool_size  3  ,    2  .

### 3.9.4.        ###

`logpresso.indexGlobals`                  :

araqne> logpresso.indexGlobals
Index Global Configs
--------
index_flush_threshold: 50000
```

~~

`logpresso.setIndexGlobal key value` 명령을 사용하여 각 설정을 변경할 수 있으며, 의미는 아래와 같습니다:

- index_flush_threshold: 5만개 단위로 풀텍스트 인덱스를 빌드합니다. 단위가 클수록 전체 디스크 사용 용량은 줄어들고 메모리는 많이 사용합니다. 인덱스 수가 적고 하드웨어가 고사양인 경우 크게 설정하고, 인덱스 수가 많고 하드웨어가 저사양인 경우 작게 설정하는 것을 권장합니다.

3.10. 데이터 백업 및 복원

로그프레소가 설치된 하드웨어의 장애 발생 시 복구하려면 사전에 아래와 같이 설정, 데이터 원본, 인덱스를 각각 백업해야 합니다. 백업 시에는 인덱스 백업을 먼저 수행하고 데이터 원본을 백업하는 것을 권장합니다. 데이터가 기록되고 있는 최신 일자의 데이터를 백업하려고 시도할 경우 불완전한 파일이 백업될 수 있으며, 데이터와 인덱스의 불일치가 발생할 수 있습니다. 이 경우 복원 후에 시동 복구 기능을 이용하여 불완전한 파일 손상부를 제거하고 정상 파일로 복원해야 합니다.

3.10.1. 설정 백업

로그프레소가 사용하는 아라크네 설정 DB는 설정의 형상관리를 지원하며, 설정 스냅샷의 익스포트 및 임포트를 지원합니다. 아래의 절차를 거쳐서 일일이 백업하기 번거로운 경우, 아라크네 데이터 디렉터리 ($araqne.data.dir) 아래의 araqne-confdb 디렉터리를 그대로 압축하여 보관하고 이후 압축 해제하여 복원할 수도 있습니다.

1) 설정 데이터베이스 목록 조회

```
araqne> conf.databases
Databases
----
araqne-core
araqne-cron
logpresso
..    ..
```

2) 데이터베이스 익스포트

conf.exportFile 명령을 사용하여 특정 리비전의 설정을 익스포트 할 수 있습니다. 명령어 설명은 다음과 같습니다.

```
araqne> conf.exportFile
 Description
       export db data
 Arguments
       1. database name: database name (required)
       2. file path: export file path (required)
       3. export revision: export revision id (optional)
araqne>
```

- [필수] 데이터베이스 이름
- [필수] 익스포트 파일 경로 (이름만 지정 시 아라크네 코어가 위치한 디렉터리에 덤프)
- [선택] 익스포트 대상 리비전, 미 입력시 가장 최신 버전을 덤프합니다. 덤프 파일은 JSON 형식으로 기록됩니다.

3.10.2. 설정 복원

기존 데이터베이스 혹은 새로 생성한 데이터베이스에 백업했던 데이터를 임포트합니다.

```
araqne> conf.importFile
 Description

import db data

Arguments
```

```
1. database name: database name (required)
2. file path: target file path (required)
```

- [필수] 데이터베이스 이름: 데이터베이스 이름을 입력합니다. 데이터베이스가 존재하지 않으면 오류를 내고 중단합니다.
- [필수] 파일 경로: conf.exportFile 명령으로 백업했던 파일의 경로를 입력합니다.

예시) logpresso 데이터베이스에 logpresso.cdb 파일을 임포트

```
araqne> conf.importFile logpresso logpresso.cdb
imported logpresso from D:\20131025\logpresso.cdb
```

설정 복원 후 아라크네 코어 데몬의 재시작이 필요합니다. 아키텍처를 잘 알고 있는 경우, 번들 단위의 재시작으로 설정을 다시 읽도록 할 수 있습니다.

3.10.3 데이터 원본 백업

araqne-logstorage 2.3.0 버전 이상 지원

로그스토리지에 저장된 데이터 원본을 기간별로 외부에 백업할 수 있습니다. 선택한 백업 매체에 따라 다른 설정을 요구합니다. 테이블 목록은 와일드카드를 지원하며, 쉼표로 구분하여 입력할 수 있습니다. 기간을 비워두면 무제한으로 지정되고, yyyyMMdd 형식으로 입력할 수 있습니다. 백업이 진행되는 동안 실시간으로 어떤 파일을 백업하고 있는지 진행상황이 출력됩니다. Ctrl-C를 누르면 콘솔 출력이 중단되지만 백그라운드에서 이후 작업이 계속 진행됩니다.

백업 완료 시 $araqne.log.dir 디렉터리에 백업 리포트가 텍스트 파일로 생성됩니다. 리포트 파일을 열어보면 어떤 파일의 백업이 성공하고 실패했는지 전체 내역을 확인할 수 있습니다. 백업하려는 위치에 동일한 파일이 존재하는 경우 해당 파일의 백업이 실패합니다.

예시) 로컬 파일시스템에 모든 테이블 데이터를 백업

```
araqne> logstorage.backup
Available backup media types [local]
 Select media type: local
 Backup Path: d:/backup
 Table names (enter to backup all tables):
 Range from (yyyyMMdd, enter to unlimited):
 Range to (yyyyMMdd, enter to unlimited):
 Total 4 tables
```

```
Requires 600 bytes
Proceed? (y/N): y
started backup job
[2013-10-27 22:30:37] backup started table [logpresso-alert-trend, araqne_query_logs,
  logpresso-log-trend, alerts], [600] bytes
[2013-10-27 22:30:37] >> backup table [logpresso-alert-trend]
[2013-10-27 22:30:37] << backup table [logpresso-alert-trend]
[2013-10-27 22:30:37] >> backup table [araqne_query_logs]
[2013-10-27 22:30:37] << backup table [araqne_query_logs]
[2013-10-27 22:30:37] >> backup table [logpresso-log-trend]
[2013-10-27 22:30:37] > backup file [logpresso-log-trend:2013-10-27.idx]
[2013-10-27 22:30:37] < backup file [logpresso-log-trend:2013-10-27.idx]
[2013-10-27 22:30:37] > backup file [logpresso-log-trend:2013-10-27.dat]
[2013-10-27 22:30:37] < backup file [logpresso-log-trend:2013-10-27.dat]
[2013-10-27 22:30:37] << backup table [logpresso-log-trend]
[2013-10-27 22:30:37] >> backup table [alerts]
[2013-10-27 22:30:37] << backup table [alerts]
[2013-10-27 22:30:37] backup completed, table [logpresso-alert-trend, araqne_query_logs,
  logpresso-log-trend, alerts], [600] bytes
Press ctrl-c to exit monitor
```

로컬 백업 매체의 경우 지정된 백업 경로 아래에 table 디렉터리를 생성하고, 그 아래에 원본의 테이블 ID로 디렉터리를 생성한 후 일자별 데이터 파일을 복사합니다.

3.10.4 데이터 원본 복원

araqne-logstorage 2.3.0 버전 이상 지원

백업 매체에 저장된 데이터 원본을 이용하여 로그스토리지 테이블 데이터를 복원할 수 있습니다. 백업 시와 마찬가지로 백업 매체에 따라 다른 설정을 요구합니다. 테이블 목록은 와일드카드를 지원하며, 쉼표로 구분하여 입력할 수 있습니다. 복원이 진행되는 동안 실시간으로 어떤 파일을 복원하고 있는지 진행상황이 출력됩니다. Ctrl-C를 누르면 콘솔 출력이 중단되지만 백그라운드에서 이후 복원 작업이 계속 진행됩니다.

테이블이 존재하지 않는 경우 자동으로 생성되고 백업 시점의 테이블 메타데이터가 복원됩니다. 테이블이 이미 존재하는 경우, 데이터만 복원됩니다. 복원 완료 시 $araqne.log.dir 디렉터리에 복원 리포트가 텍스트 파일로 생성됩니다. 리포트 파일을 열어보면 어떤 파일의 복원이 성공하고 실패했는지 전체 내역을 확인할 수 있습니다. 복원하려는 위치에 동일한 파일이 존재하는 경우 해당 파일의 복원이 실패합니다.

예시) 로컬 파일시스템 백업으로 모든 테이블 데이터를 복원

```
araqne> logstorage.restore
 Available backup media types [local]
 Select media type: local
 Backup Path: d:/backup
 Table names (enter to restore all tables):
 Total 4 tables
 Restore 600 bytes
 Proceed? (y/N): y
 started restore job
 [2013-10-27 22:47:20] restore started table [logpresso-alert-trend, araqne_query_logs,
   logpresso-log-trend, alerts], [600] bytes
 [2013-10-27 22:47:20] >> restore table [logpresso-alert-trend]
 [2013-10-27 22:47:20] << restore table [logpresso-alert-trend]
 [2013-10-27 22:47:20] >> restore table [araqne_query_logs]
 [2013-10-27 22:47:20] << restore table [araqne_query_logs]
 [2013-10-27 22:47:20] >> restore table [logpresso-log-trend]
 [2013-10-27 22:47:20] > restore file [logpresso-log-trend:2013-10-27.dat]
 [2013-10-27 22:47:20] < restore file [logpresso-log-trend:2013-10-27.dat]
 [2013-10-27 22:47:20] > restore file [logpresso-log-trend:2013-10-27.idx]
 [2013-10-27 22:47:20] < restore file [logpresso-log-trend:2013-10-27.idx]
 [2013-10-27 22:47:20] << restore table [logpresso-log-trend]
 [2013-10-27 22:47:20] >> restore table [alerts]
 [2013-10-27 22:47:20] << restore table [alerts]
[2013-10-27 22:47:20] restore completed, table [logpresso-alert-trend, araqne_query_logs,
   logpresso-log-trend, alerts], [600] bytes
 Press ctrl-c to exit monitor
```

3.10.5 인덱스 백업

logpresso-index 2.2.4 버전 이상 지원

로그프레소 인덱스 파일을 기간별로 외부에 백업할 수 있습니다. 백업 명령 시 선택한 백업 매체에 따라 다른 설정을 요구합니다. 인덱스 목록 지정 시, 테이블 혹은 테이블.인덱스 형식으로 이름을 입력합니다. 테이블 이름만 입력하면 해당 테이블의 모든 인덱스를 백업 대상으로 선택하고, 테이블.인덱스 형식으로 입력하면 해당 인덱스만 백업 대상으로 선택합니다. 와일드카드를 지원하므로 . 혹은 *.idx 같은 형식으로 백업 대상 인덱스 목록을 지정할 수 있습니다. 기간을

비워두면 무제한으로 지정되고, yyyyMMdd 형식으로 입력할 수 있습니다. 백업이 진행되는 동안 실시간으로 어떤 파일을 백업하고 있는지 진행상황이 출력됩니다. Ctrl-C를 누르면 콘솔 출력이 중단되지만 백그라운드에서 이후 작업이 계속 진행됩니다.

백업 완료 시 $araqne.log.dir 디렉터리에 백업 리포트가 텍스트 파일로 생성됩니다. 리포트 파일을 열어보면 어떤 파일의 백업이 성공하고 실패했는지 전체 내역을 확인할 수 있습니다. 백업하려는 위치에 동일한 파일이 존재하는 경우 해당 파일의 백업이 실패합니다.

예시) 로컬 파일시스템에 iis 테이블의 모든 인덱스 백업

```
araqne> logpresso.backupIndex
 Backup path: d:/backup
 Table/Index Names (enter to all): iis
 Range from (yyyyMMdd, enter to unlimited):
 Range to (yyyyMMdd, enter to unlimited):
 Total 1 index
 Requires 73,585,622 bytes, free space of media is 993,690,304,512 bytes
 Proceed? (y/N): y
 started backup job
 [2013-10-27 23:02:23] backup started, index [iis:idx], total [73,585,622] bytes
 [2013-10-27 23:02:23] >> backup index [iis:idx]
 [2013-10-27 23:02:23] > backup file [iis:idx:2013-10-27.1.bpos]
 [2013-10-27 23:02:23] < backup file [iis:idx:2013-10-27.1.bpos]
 [2013-10-27 23:02:23] > backup file [iis:idx:2013-10-27.1.bseg]
 [2013-10-27 23:02:23] < backup file [iis:idx:2013-10-27.1.bseg]
 [2013-10-27 23:02:23] > backup file [iis:idx:2013-10-27.bpos]
 [2013-10-27 23:02:23] < backup file [iis:idx:2013-10-27.bpos]
 [2013-10-27 23:02:23] > backup file [iis:idx:2013-10-27.bseg]
 [2013-10-27 23:02:23] < backup file [iis:idx:2013-10-27.bseg]
 [2013-10-27 23:02:23] > backup file [iis:idx:2013-10-27.seg]
 [2013-10-27 23:02:23] < backup file [iis:idx:2013-10-27.seg]
 [2013-10-27 23:02:23] > backup file [iis:idx:2013-10-27.pos]
 [2013-10-27 23:02:23] < backup file [iis:idx:2013-10-27.pos]
 [2013-10-27 23:02:23] << backup index [iis:idx]
 [2013-10-27 23:02:23] backup completed, index [iis:idx], total [73,585,622] bytes
 Press ctrl-c to exit monitor
```

3.10.6 인덱스 복원

logpresso-index 2.2.4 버전 이상 지원

백업 매체에 저장된 파일을 이용하여 인덱스 데이터를 복원할 수 있습니다. 백업 시와 마찬가지로 백업 매체에 따라 다른 설정을 요구합니다. 인덱스 목록 지정 시, 테이블 혹은 테이블.인덱스 형식으로 이름을 입력합니다. 테이블 이름만 입력하면 해당 테이블의 모든 인덱스를 백업 대상으로 선택하고, 테이블.인덱스 형식으로 입력하면 해당 인덱스만 백업 대상으로 선택합니다. 와일드카드를 지원하므로 . 혹은 *.idx 같은 형식으로 백업 대상 인덱스 목록을 지정할 수 있습니다. 복원이 진행되는 동안 실시간으로 어떤 파일을 복원하고 있는지 진행상황이 출력됩니다. Ctrl-C를 누르면 콘솔 출력이 중단되지만 백그라운드에서 이후 복원 작업이 계속 진행됩니다.

테이블이 존재하지 않는 경우 복원이 실패합니다. 해당 테이블의 인덱스가 존재하지 않는 경우 백업 시점의 인덱스 설정으로 인덱스가 생성됩니다. 복원 완료 시 $araqne.log.dir 디렉터리에 복원 리포트가 텍스트 파일로 생성됩니다. 리포트 파일을 열어보면 어떤 파일의 복원이 성공하고 실패했는지 전체 내역을 확인할 수 있습니다. 복원하려는 위치에 동일한 파일이 존재하는 경우 해당 파일의 복원이 실패합니다.

예시) 로컬 파일시스템 백업으로 iis 인덱스를 복원

```
araqne> logpresso.restoreIndex
 Backup path: d:/backup
 Table/Index Names (enter to all): iis
 Range from (yyyyMMdd, enter to unlimited):
 Range to (yyyyMMdd, enter to unlimited):
 Total 1 index
 Restore 73,585,622 bytes
 Proceed? (y/N): y
 started restore job
 [2013-10-27 23:09:07] restore started, index [iis:idx], total [73,585,622] bytes
 [2013-10-27 23:09:07] >> restore index [iis:idx]
 [2013-10-27 23:09:07] > restore file [iis:idx:2013-10-27.1.bpos]
 [2013-10-27 23:09:07] < restore file [iis:idx:2013-10-27.1.bpos]
 [2013-10-27 23:09:07] > restore file [iis:idx:2013-10-27.1.bseg]
 [2013-10-27 23:09:07] < restore file [iis:idx:2013-10-27.1.bseg]
 [2013-10-27 23:09:07] > restore file [iis:idx:2013-10-27.bpos]
 [2013-10-27 23:09:07] < restore file [iis:idx:2013-10-27.bpos]
 [2013-10-27 23:09:07] > restore file [iis:idx:2013-10-27.bseg]
 [2013-10-27 23:09:07] < restore file [iis:idx:2013-10-27.bseg]
 [2013-10-27 23:09:07] > restore file [iis:idx:2013-10-27.pos]
```

```
[2013-10-27 23:09:07] < restore file [iis:idx:2013-10-27.pos]
[2013-10-27 23:09:07] > restore file [iis:idx:2013-10-27.seg]
[2013-10-27 23:09:07] < restore file [iis:idx:2013-10-27.seg]
[2013-10-27 23:09:07] << restore index [iis:idx]
[2013-10-27 23:09:09] restore completed, index [iis:idx], total [73,585,622] bytes
Press ctrl-c to exit monitor
```

3.11. FTP 연동 설정

로그프레소에서 FTP를 통해 로그를 수집하려는 경우 미리 FTP 프로파일을 설정해두어야 합니다.

3.11.1. FTP 프로파일 목록 조회

`logpresso.ftpProfiles` 명령을 사용하여 기존에 설정된 프로파일 목록을 조회할 수 있습니다.

```
araqne> logpresso.ftpProfiles
FTP Profiles
--------------
name=local, host=localhost:21, username=xeraph, active=false
```

3.11.2. FTP 프로파일 생성

`logpresso.createFtpProfile` 명령을 사용하여 프로파일을 추가할 수 있습니다. 설정 항목은 다음과 같습니다.

- [필수] FTP 프로파일 이름: 기존의 FTP 프로파일과 중복되지 않는 유일한 이름을 부여합니다.
- [필수] 도메인 혹은 IP 주소: FTP로 접속할 호스트 주소를 입력합니다.
- [선택] 포트: FTP 포트 번호를 입력합니다. 기본값은 21입니다.
- [선택] 계정: 계정 이름을 입력합니다. 익명 로그인의 경우 건너뜁니다.
- [선택] 암호: 계정 암호를 입력합니다. 익명 로그인의 경우 건너뜁니다.
- [선택] 액티브 모드: 접속하는 클라이언트 측 방화벽이 없는 경우 y를 입력하면 액티브 모드로 설정됩니다. 최근 대부분 클라이언트 측 방화벽이 존재하므로 기본적으로 FTP 접속 후에 패시브 모드로 동작합니다.

3.11.3. FTP 프로파일 삭제

`logpresso.removeFtpProfile` 명령을 사용하여 FTP 프로파일을 삭제할 수 있습니다. 매개변수는 다음과 같습니다:
* [필수] FTP 프로파일 이름: 삭제할 FTP 프로파일 이름을 입력합니다.

기존에 설정된 로거의 경우 프로파일이 삭제되면 로그 수집 동작이 실패하게 됩니다.

3.12. JDBC 연동 설정

로그프레소에서 SQL을 사용하여 데이터베이스에 질의하려면 JDBC 프로파일을 미리 설정해두어야 합니다.

(dbquery 설명 바로가기)

3.12.1. JDBC 프로파일 목록 조회

현재 설정되어 있는 JDBC 접속 프로파일 목록을 조회합니다.

```
araqne@bombom demo> logpresso.jdbcProfiles
JDBC Profiles
---------------
name=worldcup, con=jdbc:oracle:thin:@wood:1521:worldcup, user=worldcup, password=worldcup,
readonly=false
```

3.12.2. JDBC 프로파일 생성

logpresso.createJdbcProfile 명령을 이용하여 JDBC 접속 프로파일을 생성합니다. 매개변수는 아래와 같습니다: * [필수] 프로파일 이름: 프로파일을 식별하는데 사용할 유일한 이름을 부여합니다. * [필수] 접속 문자열: JDBC 스키마 형식으로 접속 문자열을 입력합니다. * [선택] 읽기 전용 모드: JDBC 연결 생성 시 읽기 전용 모드로 설정합니다. true나 false를 설정할 수 있으며, 기본값은 false입니다. * [선택] 계정: DB 계정 이름을 입력합니다. 기본값은 null입니다. * [선택] 암호: 암호를 입력합니다. 기본값은 null입니다.

```
araqne@bombom demo> logpresso.createJdbcProfile worldcup jdbc:oracle:thin:@wood:1521:worldcup
true worldcup worldcup
created
```

3.12.3. JDBC 프로파일 삭제

logpresso.removeJdbcProfile 명령을 이용하여 기존의 JDBC 접속 프로파일을 삭제합니다. 매개변수는 아래와 같습니다: * [필수] 프로파일 이름: 삭제할 프로파일 이름을 입력합니다. ~ araqne@bombom demo> log-presso.removeJdbcProfile worldcup removed ~

3.12.4. JDBC 접속 문자열 템플릿

- 오라클 DB: jdbc:oracle:thin:@hostname:1521:dbname
- 마이크로소프트 SQL 서버: jdbc:sqlserver://hostname:1433;DatabaseName=dbname
- 마리아DB: jdbc:mariadb://hostname:3306/dbname
- MySQL: jdbc:mysql://hostname:3306/dbname
- 테라데이터 Aster: jdbc:ncluster://hostname:2406/dbname
- 티베로: jdbc:tibero:thin:@hostname:8629:dbname
- 아파치 하이브: jdbc:hive2://hostname:10000

3.13. SSH 연동 설정

로그프레소에서 sshfs 커맨드를 사용하여 원격지 텍스트 파일에 대한 쿼리를 수행하거나, SSH 혹은 SFTP를 통해 로그를 수집하려는 경우에는 SSH 프로파일을 미리 설정해두어야 합니다.

3.13.1. SSH 프로파일 목록 조회

`logpresso.sshProfiles` 명령을 사용하여 기존에 설정된 프로파일 목록을 조회할 수 있습니다.

```
araqne> logpresso.sshProfiles
SSH Profiles
--------------
name=wood, host=wood.example.org:22, user=sshbot
```

3.13.2. SSH 프로파일 생성

`logpresso.createSshProfile` 명령을 사용하여 프로파일을 추가할 수 있습니다. 설정 항목은 다음과 같습니다: * [필수] SSH 프로파일 이름: 기존의 SSH 프로파일과 중복되지 않는 유일한 이름을 부여합니다. * [필수] 도메인 혹은 IP 주소: SSH로 접속할 호스트 주소를 입력합니다. * [선택] 포트: SSH 포트 번호를 입력합니다. 기본값은 22입니다. * [필수] 계정: 계정 이름을 입력합니다. * [필수] 암호: 계정의 암호를 입력합니다.

3.13.3. SSH 프로파일 삭제

`logpresso.removeSshProfile` 명령을 사용하여 프로파일을 삭제할 수 있습니다. 매개변수는 다음과 같습니다.

- [필수] SSH 프로파일 이름: 삭제할 SSH 프로파일 이름을 입력합니다.

기존에 설정된 로거의 경우 프로파일이 삭제되면 로그 수집 동작이 실패하게 됩니다.

3.14. 페더레이션 구성

logpresso-query 0.1.0 버전부터 지원

페더레이션 구성은 독립적인 다수의 로그프레소 서버를 하나로 묶어서 쿼리할 수 있도록 지원합니다. 페더레이션을 구성하는 각 노드는 설정한 이름으로 된 고유한 이름공간(namespace)을 가지게 되고, 각 노드의 테이블을 대상으로 쿼리할 때 콜론(:)으로 구분되는 한정자를 사용할 수 있습니다.

가령, 검색을 전담하는 노드에 n1 이름의 원격 노드를 등록한다면, n1 노드의 weblogs 테이블을 n1:weblogs 라는 이름으로 지칭할 수 있습니다. 이름공간에 와일드카드()를 사용하면 일치하는 모든 노드에서 검색 및 조회를 수행할 수 있습니다. 가령, table :weblogs 쿼리를 실행하면 쿼리를 실행하는 로컬 노드를 포함하여 등록된 모든 원격 노드의 weblogs 테이블에서 로그를 조회하게 됩니다.

페더레이션 구성은 각기 독립적인 로그프레소 서버를 연계하므로 DB 계정이나 권한 역시 별도로 관리됩니다. 따라서 노드 간 계정 위임 및 가장(impersonation)은 지원되지 않으며, 등록된 계정과 암호로 로그프레소 서버에 로그인하여 원격 쿼리를 실행하게 됩니다.

table 혹은 fulltext 쿼리를 시도할 때 지정된 대상 테이블의 이름공간에 와일드카드(*)가 포함되어 있으면, 분산 쿼리 플래너가 쿼리 실행 계획을 재작성하고 원격 쿼리가 포함된 쿼리 커맨드 파이프라인을 실행하게 됩니다. 분산 쿼리 플래너는 eval, search, rex 같은 쿼리 커맨드 뿐 아니라, stats나 timechart 처럼 그룹 함수가 포함된 쿼리 커맨드를 최대한 각 원격 노드에서 병렬로 실행하게 하여 전체 수행 시간이 최소화되도록 실행 계획을 재작성합니다. 또한 원격 노드 쿼리 시 등록된 노드 이름을 _node 필드로 추가하여 노드별 통계 쿼리를 수행할 수 있도록 합니다.

3.14.1. 원격 노드 목록 조회

현재 등록된 모든 원격 노드를 설정과 함께 보여줍니다.

```
araqne> logpresso.nodes
```

3.14.2. 원격 노드 등록

새로운 원격 노드를 등록합니다.

```
araqne> logpresso.registerNode [ ] [  ] [ ] [    ]
```

3.14.3. 원격 노드 삭제

기존의 원격 노드 구성을 삭제합니다.

```
araqne> logpresso.unregisterNode [ ]
```

3.14.4. 원격 노드 접속 테스트

현재 설정된 주소, 계정, 암호를 이용하여 정상적으로 접속이 가능한지 시험합니다.

```
araqne> logpresso.pingNode [ ]
```

4장. 실시간 로그 수집 설정

4.1. 로그 수집 설정

로그프레소에 내장된 여러가지 유형의 로그 수집기를 이용하여 데이터 원본을 구성할 수 있는데, 이를 로거(Logger)라고 부릅니다. 로그프레소는 로그 수집과 로그 저장 단계가 분리되어 있습니다. 따라서 로그에 대한 실시간 수집 및 분석만 수행하고 저장을 하지 않거나, 수집된 모든 로그를 특정한 테이블에 저장하도록 구성할 수 있습니다.

로거는 크게 액티브 로거와 패시브 로거로 구분됩니다. 액티브 로거는 설정된 주기에 따라 일정한 간격으로 동일한 동작을 반복 수행하는 로거를 의미합니다. 가령, 파일에서 로그를 수집하는 경우, 설정된 주기마다 파일의 증가분을 확인하여 로그를 읽어들입니다. 혹은 데이터베이스에서 일정 주기마다 테이블의 데이터를 읽어오는 예를 생각할 수 있습니다. 패시브 로거는 별도의 주기 없이 수동적으로 입력을 받아들이는 로거입니다. 가령 UDP 패킷을 수신해서 로깅하는 경우에는 별도의 실행 주기가 필요하지 않습니다.

이 절에서는 일반적인 로그 수집 설정에 대해서 다룹니다. 각 로그 수집 방법에 대해서는 별도의 절에서 사용 예와 함께 상세하게 설명합니다.

4.1.1. 로거 팩토리 목록 조회

로거 팩토리는 특정한 유형의 로거를 만들거나 삭제할 수 있도록 지원하는 구성요소입니다. 초기 설치 방식에 따라 다르지만, 보통은 아래와 같은 로거 팩토리가 기본으로 내장됩니다. `logapi.loggerFactories` 명령을 사용하여 전체 로거 팩토리 목록을 조회할 수 있습니다.

```
araqne> logapi.loggerFactories
 Logger Factories
```

```
+-----+----------+
| name | display name |
+-----+----------+
| dirwatch | Directory watcher |
| netflow | netFlow |
| pcap | pcap |
| selector | selector |
| syslog | syslog logger |
| syslog-relay | syslogger for relayed syslog |
| textfile | Text file logger |
+-----+----------+
```

4.1.2. 로거 목록 조회

로거는 특정한 설정으로 생성된 데이터 원본에 해당됩니다. 가령, 특정한 원격지 IP에서 전송되는 시스로그를 수신하도록 설정된 syslog 로거를 구성할 수 있습니다. `logapi.loggers` 명령을 이용하여 전체 로거 목록을 조회할 수 있습니다.

```
araqne> logapi.loggers
 Loggers
 +------+---+---+----+----+----+
 | name | factory | status | intvl.(ms) | log count | last log |
 +------+---+---+----+----+----+
 | local\flowtest | netflow | running | 0 | 0 | null |
 | local\syslog-all | syslog | running | 0 | 0 | null |
 +------+---+---+----+----+----+
```

만약 특정한 문자열을 포함하는 로거만 검색하려고 한다면, 아래와 같이 명령합니다:

예시) flow를 포함하는 로거 목록 조회

```
araqne> logapi.loggers flow
```

4.1.3. 로거 생성

로거 생성은 수집 방식과 관계없이 모두 동일한 명령어를 통해서 이루어집니다. `logapi.createLogger` 명령어의 사용법은 아래와 같습니다:

```
araqne> logapi.createLogger
 Description

create new logger

Arguments

1. logger factory name: logger factory name. try logapi.loggerFactories command. (required)
 2. logger namespace: new logger namespace (required)
 3. logger name: new logger name (required)
 4. description: the description of new logger (optional)
```

각 인자는 아래와 같습니다.

- 1.[필수] 로거 팩토리 이름: 이전에 logapi.loggerFactories를 사용하여 조회되는 로거 팩토리의 이름을 의미합니다.
- 2.[필수] 로거 이름공간: 이름이 겹치지 않도록 이름공간을 사용합니다. 로컬에서 만드는 로거는 관례적으로 "local" 문자열을 입력합니다. 원격지의 에이전트를 통해 연동되는 로거의 경우 해당 호스트의 식별자가 이름공간으로 사용됩니다.
- 3.[필수] 로거 이름: 로거 이름공간 안에서 겹치지 않는 이름을 임의로 부여할 수 있습니다.
- 4.[선택] 로거 설명: 로거에 대한 임의의 설명을 입력할 수 있습니다.

위의 인자를 모두 입력하여 명령을 실행하게 되면, 아래와 같이 로거 팩토리에 따라 추가적인 로거 설정 인자를 입력받습니다.

예시) syslog 로거 팩토리를 사용하여 nxg 로거 생성

```
araqne> logapi.createLogger syslog local nxg
 remote ip (required)? 192.168.0.10
 syslog facility (optional)?
 transformer (optional, enter to skip)?
 logger created: name=local\nxg, factory=local\syslog, status=stopped (passive),
   log count=0, last start=null, last run=null, last log=null
```

위의 예에서 remote ip나 syslog facility는 syslog 로거 팩토리에 대해서만 입력받는 항목입니다. 로거 팩토리에 따라 다른 설정 입력을 요구받게 됩니다.

transformer는 수집되는 원본 로그에 대해 추가적인 변형이 필요할 때 사용합니다. 가령, 원본 로그에 대해 특정한 태깅을 하려는 경우에는 keyvalue 트랜스포머를 사용해서 트랜스포머 인스턴스를 생성하고 태깅을 수행할 수 있습니다. 트랜스포머에 대해서는 별도의 절에서 설명합니다.

4.1.4. 로거 시작

액티브 로거, 패시브 로거에 관계 없이 로거를 명시적으로 시작시켜야만 로그 수집 동작이 활성화됩니다. 다만, 패시브 로거의 경우 별도의 수집 주기 설정이 불필요하다는 차이가 있습니다.

로거를 시작시키려면 `logapi.startLogger` 명령을 사용합니다:

```
araqne> logapi.startLogger
 Description

start the logger

Arguments

 1. logger fullname: the logger fullname to start (required)
 2. interval: sleep time of active logger thread in milliseconds. 60000ms by default.
     passive logger will ignore interval (optional)
```

- 1.[필수] 로거의 전체 이름: 이름공간₩이름 형식으로 로거의 전체 이름을 입력합니다.
- 2.[선택] 수집 동작 주기: 액티브 로거의 경우 몇 밀리초마다 동작할지 지정합니다. 기본값은 60초입니다.

4.1.5. 로거 정지

로거를 정지하려면 `logapi.stopLogger` 명령을 사용합니다:

```
araqne> logapi.stopLogger
 Description

stop the logger

Arguments

 1. logger name: the logger name to stop (required)
```

```
2. max wait time: max wait time in milliseconds (optional)
```

- 1.[필수] 로거의 전체 이름: 이름공간₩이름 형식으로 로거의 전체 이름을 입력합니다.
- 2.[선택] 로거가 정지할 때까지 최대 몇 밀리초를 기다릴지 지정합니다. 기본값은 5초입니다.

4.1.6. 로거 삭제

`logapi.removeLogger` 명령을 사용하여 로거를 삭제할 수 있습니다. 만약 로거가 동작하고 있으면 정지시킨 후에 삭제합니다:

```
araqne> logapi.removeLogger
 Description

remove logger

Arguments

1. logger fullname: the logger fullname (required)
```

- 1.[필수] 로거의 전체 이름: 이름공간₩이름 형식으로 로거의 전체 이름을 입력합니다.

4.1.7. 마지막 로그 조회

(araqne-log-api 2.6.0부터 지원)

`logapi.lastLogs` 명령을 사용하여 로거별로 시스템 부팅 후 수집된 마지막 로그를 조회할 수 있습니다. 첫번째 인자를 주면 로거 이름에 대하여 필터링을 수행합니다.

```
araqne@bombom demo> logapi.lastLogs
--
Logger [local\wtmp] Last Timestamp [2013-09-02 22:38:25+0900]
 {host=182.209.194.63, session=0, pid=8917, type=UserProcess, user=xeraph}
--
Logger [local\iis] Last Timestamp [2013-09-19 13:44:02+0900]
{line=2007-10-22 01:47:53 W3SVC1 123.223.21.233 GET /solution/1.982/asp/strawlv01982_msg.asp
   t=1&m=0013D4E55911 80 - 111.217.245.234 UtilMind+HTTPGet 200 0 0}
```

4.2. 디렉터리 로그 파일 수집 설정

디렉토리 와치 (dirwatch) 로거는 롤링되지 않는 텍스트 로그 파일을 일정 주기마다 수집하려고 할 때 사용합니다. 가령 일자별 혹은 시간대별로 순차 생성되는 로그 파일을 수집할 때 사용합니다. 디렉토리 와치 로거는 아래와 같은 설정을 입력받습니다: * [필수] 디렉터리 경로: 로그 파일이 위치하는 파일시스템 경로를 의미합니다. * [필수] 파일이름 정규표현식 패턴: 디렉터리 경로에 존재하는 파일 중 이름이 정규표현식 패턴에 일치하는 경우에만 수집합니다. 정규표현식 그룹을 쓰는 경우 파일 이름에서 날짜 문자열을 추출합니다. * [선택] 날짜 추출 정규표현식 패턴: 로그에서 날짜 문자열을 추출합니다. 정규표현식 그룹으로 묶인 모든 부분을 이어붙여서 하나의 날짜 문자열을 만들어냅니다. 파일이름 정규표현식의 그룹으로 추출된 날짜문자열은 가장 앞 부분에 위치합니다. * [선택] 날짜 파싱 포맷: 날짜 문자열을 파싱하는데 사용할 날짜 포맷을 설정합니다. (예: yyyy-MM-dd HH:mm:ss) * [선택] 날짜 로케일: 날짜 문자열의 로케일. 가령 날짜 파싱 포맷의 지시자 중 MMM의 해석은 로케일에 따라 "Jan" 혹은 "1월"로 해석됩니다. 기본값은 en입니다. * [선택] 로그 시작 정규식: 로그의 시작 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다. * [선택] 로그 끝 정규식: 로그의 끝 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다. * [선택] 문자집합: 텍스트 파일 해석에 사용할 문자집합 코드를 입력합니다. 기본값은 utf-8입니다.

여러 개의 파일이 수집 대상인 경우, 파일의 절대 경로를 사전순으로 정렬하여 순서대로 읽어들입니다. 이하에서는 여러가지 사용 예를 살펴보도록 하겠습니다.

예시 1) /var/log/examples 디렉터리에 example.log.yyyy-MM-dd-HH 로그 파일이 순차 생성되는 경우

0;20130615 141618;192.168.0.10;E002;20130615 141618;192.168.0.11; * 디렉터리 경로: /var/log/examples * 파일이름 정규표현식 패턴: example.log. 날짜 추출 정규표현식 패턴: □+[0-8-6]: * 날짜 파싱 포맷: yyyyMMdd HHmmss

예시 2) /logdata/c6509 디렉터리에 c6509.log.yyyyMMdd 로그 파일이 쌓이는 경우

May 01 23:59:59: %LINK-SP-3-UPDOWN: Interface GigabitEthernet8/15, changed state to down * 디렉터리 경로: /logdata/c6509 * 파일이름 정규표현식 패턴: c6509.log.(8□* 날짜 추출 정규표현식 패턴: 「]* 2[2]:[2]:[2]* 날짜 파싱 포맷: yyyyMMddHHmmss (파일이름 정규표현식으로 추출된 yyyyMMdd 그룹 1개와 날짜 추출 정규표현식에서 추출된 HH, mm, ss 그룹 3개가 하나로 병합되어 파싱됩니다.)

예시 3) /araqne/log 디렉터리에 araqne.log.yyyy-MM-dd 파일이 순차 생성되는 경우

```
[2013-06-14 13:55:02,186] INFO (LogIndexerEngine) - logpresso index: counter reset thread started
 [2013-06-14 13:55:40,647] WARN (ScriptRunner) - script runner:
 java.lang.reflect.InvocationTargetException
     at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
     at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
     at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
```

```
        at java.lang.reflect.Method.invoke(Unknown Source)
        at org.araqne.console.ScriptRunner.invokeScript(ScriptRunner.java:209)
        at org.araqne.console.ScriptRunner.run(ScriptRunner.java:190)
        at java.lang.Thread.run(Unknown Source)
          Caused by: java.lang.InterruptedException
        at org.araqne.console.ReadLineHandler.read(ReadLineHandler.java:156)
        at org.araqne.console.ReadLineHandler.getLine(ReadLineHandler.java:46)
        at org.araqne.console.ConsoleInputStream.readLine(ConsoleInputStream.java:56)
        at org.araqne.script.ScriptContextImpl.readLine(ScriptContextImpl.java:194)
        at org.araqne.log.api.impl.LogApiScript.setOption(LogApiScript.java:629)
        at org.araqne.log.api.impl.LogApiScript.createParser(LogApiScript.java:99)
        ... 7 more
```

- 디렉터리 경로: /araqne/log
- 파일이름 정규표현식 패턴: araqne.log.*
- 날짜 추출 정규표현식 패턴: [(.*),
- 날짜 파싱 포맷: yyyy-MM-dd HH:mm:ss
- 로그 구분자 정규식: [4□2□2□2□2□2□3□

4.3. 선택자 로그 수집 설정

선택자 (selector) 로거는 다른 로거에서 수집하는 데이터 중 일부만 선택적으로 가져오려고 할 때 사용합니다. 선택자 로거는 아래와 같은 설정을 입력받습니다: * [필수] 원본 로거 이름: 네임스페이스를 포함한 원본 로거 이름을 입력합니다. * [필수] 텍스트 패턴: 원본 데이터의 line 필드 값과 비교할 접두어 문자열을 입력합니다. 가령, 0; 으로 입력하면 line 필드 값이 0; 으로 시작하는 로그만 수집하게 됩니다.

예시) local₩apache 이름의 아파치 로그 수집기로부터 10.x 대역 IP 로그만 수집하는 local₩filtered 로거 생성

```
araqne@bombom demo> logapi.createLogger selector local filtered
 Source logger name (required)? local\apache
 Text pattern (required)? 10.
 transformer (optional, enter to skip)?
 logger created: name=local\filtered, factory=local\selector, status=stopped (passive),
   log count=0, last start=null, last run=null, last log=null
~~~
```

4.4.

araqne-log-api 2.8.4

(regex-selector) . :

* [ ] : .
* [ ] : line .
* [ ] : true . . (araqne-log-api 2.9.4)

) local\netscreen NETBIOS local\filtered

Aug 09 09:00:20 [121.133.51.213] SSG140: NetScreen device_id=SSG140 [Root]system-notification-00257(traffic): start_time="2012-08-09 09:00:20" duration=0 policy_id=16 service=NETBIOS (NS) proto=17 src zone=V1-Untrust dst zone=V1-Trust action=Deny sent=0 rcvd=0 src=21.133.10.111 dst=121.113.40.255 src_port=137 dst_port=137 session_id=0 ~

설정 예시

```
araqne@bombom demo> logapi.createLogger regex-selector local filtered
 Source logger name (required)? local\netscreen
 Regex pattern (required)? service=NETBIOS(.*?)action=Deny
 transformer (optional, enter to skip)?
logger created: name=local\filtered, factory=local\regex-selector, status=stopped (passive), log count=0,
 ~~~
```

4.5.

(rotation) . logrotate .

* [ ] : .
* [ ] : . utf-8 .
* [ ] : . .
* [ ] : . (: yyyy-MM-dd HH:mm:ss)
* [ ] : . MMM "Jan" "1 " . en .

```
* [ ]       :                    .            ,              .
* [ ]      :                    .            ,              .

  1) /var/log/httpd/access_log
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
*    : /var/log/httpd/access_log
*          : \[(.*?)\]
*      : dd/MMM/yyyy:HH:mm:ss Z
*     : en

  2) /araqne/log/araqne.log

[2013-06-14 13:55:02,186]  INFO (LogIndexerEngine) - logpresso index: counter reset thread started
[2013-06-14 13:55:40,647]  WARN (ScriptRunner) - script runner:
java.lang.reflect.InvocationTargetException
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
    at org.araqne.console.ScriptRunner.invokeScript(ScriptRunner.java:209)
    at org.araqne.console.ScriptRunner.run(ScriptRunner.java:190)
    at java.lang.Thread.run(Unknown Source)
Caused by: java.lang.InterruptedException
    at org.araqne.console.ReadLineHandler.read(ReadLineHandler.java:156)
    at org.araqne.console.ReadLineHandler.getLine(ReadLineHandler.java:46)
    at org.araqne.console.ConsoleInputStream.readLine(ConsoleInputStream.java:56)
    at org.araqne.script.ScriptContextImpl.readLine(ScriptContextImpl.java:194)
    at org.araqne.log.api.impl.LogApiScript.setOption(LogApiScript.java:629)
    at org.araqne.log.api.impl.LogApiScript.createParser(LogApiScript.java:99)
    ... 7 more


*    : /araqne/log/araqne.log
*          : \[(.*),
*      : yyyy-MM-dd HH:mm:ss
*       : \[\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2},\d{3}\]
```

4.6.

(exec) . . :

* [ ] : .

) load average

* : cat /proc/loadavg

```
araqne> logapi.trace local\loadavg
tracing logger: name=local\loadavg, factory=local\exec, status=running (interval=1000ms),
  log count=0, last start=2013-08-18 20:07:39, last run=null, last log=null
local\loadavg: date=2013-08-18 20:07:56, logger=local\loadavg, data={line=0.38 0.17 0.11 1/358 18042}
local\loadavg: date=2013-08-18 20:08:14, logger=local\loadavg, data={line=0.27 0.16 0.11 1/357 18101}
local\loadavg: date=2013-08-18 20:08:32, logger=local\loadavg, data={line=0.21 0
```

4.7. WTMP

araqne-log-api 2.5.1

WTMP wtmp . WTMP :

* [ ] : wtmp . /var/log/wtmp .

:

* type: Unknown, RunLevel, BootTime, NewTime, OldTime, InitProcess, LoginProcess, UserProcess, DeadProcess,
* host:
* pid:
* session:
* user:

wtmp .

```
type=BootTime, date=2013-05-13 16:07:17, pid=0, user=reboot, host=2.6.18-274.7.1.el5xen
type=RunLevel, date=2013-05-13 16:07:17, pid=20019, user=runlevel, host=2.6.18-274.7.1.el5xen
type=InitProcess, date=2013-05-13 16:07:17, pid=717, user=, host=2.6.18-274.7.1.el5xen
type=DeadProcess, date=2013-05-13 16:07:20, pid=717, user=, host=2.6.18-274.7.1.el5xen
type=InitProcess, date=2013-05-13 16:07:20, pid=1368, user=, host=2.6.18-274.7.1.el5xen
type=LoginProcess, date=2013-05-13 16:07:20, pid=1368, user=LOGIN, host=
type=UserProcess, date=2013-05-13 16:44:25, pid=2927, user=8con, host=120.130.206.219
type=UserProcess, date=2013-05-13 17:19:07, pid=4265, user=8con, host=120.161.231
```

```
## 4.8.
```

.

```
### 4.8.1.

#### 4.8.1.1
```

.

```
araqne> syslog.open
 Description

open persistent syslog server

Arguments

1. server name: unique server name (required)
 2. port: syslog port number (optional)
 3. address: syslog bind address. 0.0.0.0 by default (optional)
 4. charset: character set name. utf-8 by default (optional)
 5. queue size: buffering queue size. 20000 by default (optional)
 6. buffer size: os receive buffer size (optional)
```

514 . utf-8 . . UDP .

UDP . JVM GC .

4.8.1.2.

.

```
araqne> syslog.close
 Description

close syslog server

Arguments

1. server name: the name of syslog server instance (required)


 ) "default"

araqne> syslog.close default
 closed
```

4.8.1.3.

.

```
araqne> syslog.servers
 Syslog Servers
------
[default] 0.0.0.0:514, charset=utf-8, capacity=20000, rx_buf_size=0, since=2013-06-05 02:25:53, received=0
```

4.8.1.4.

. Ctrl-C .

```
araqne> syslog.trace
 Description

trace a syslog receiver.

Arguments

1. server name: the name of syslog server instance (required)


 )

araqne> syslog.trace default
 press ctrl-c to stop
--------
[2013-06-05 02:28:13.536+0900] (/127.0.0.1:51245) => [fc:16, sv:5] FW-A: NetScreen device_id=FW-A [Root]sys


### 4.8.2.

#### 4.8.2.1

                    .                    :

 ) 1.2.3.4  PRI

araqne> logapi.createLogger syslog local name
 remote ip (required)? 1.2.3.4
 syslog facility (required)? -1
 transformer (optional, enter to skip)?
 logger created: name=local\name, factory=local\syslog, status=stopped (passive),
   log count=0, last start=null, last run=null, last log=null


name             . remote ip             IP      . NAT                  ,  , NAT         . syslog facility

                .
```

```
araqne> logapi.startLogger local\name
 logger started
```

:

```
araqne> logapi.logger local\name
 Logger [local\name]
-------
* Description: null
 * Logger Factory: local\syslog
 * Status: Running
 * Interval: 0ms
 * Last Log: N/A
 * Last Run: N/A
 * Log Count: 0

Configuration
-----
* facility: -1
 * remote_ip: 1.2.3.4
```

4.8.2.2.

, .

```
araqne> logpresso.createLogger
 Description

create managed logger

Arguments

 1. org domain: org domain (required)
 2. logger fullname: logger fullname (required)
```

3. table name: destination table name, use logger fullname if not specified (optional)
4. host: host name will be recorded to log data if specified (optional)

localhost , . . _host . ,

)

araqne> logpresso.createLogger localhost local\name syslogs
created

`syslog.servers` `logapi.logger` , `logdb.console` DB .

4.9. SNMP GET

logpresso-snmpmon 0.5.0

SNMP GET (snmpget) SNMP . OID- :
* [ ] SNMP : v1, v2c .
* [ ] IP : SNMP IP .
* [ ] SNMP : SNMP UDP . 161 .
* [ ] SNMP : SNMP . public .
* [ ] OID : Object ID . OID .
* [ ] : . 5 .
* [ ] : . 2 .

) NET-SNMP CPU
* SNMP : v2c
* IP : hostname
* SNMP : 161
* SNMP : public
* OID : .1.3.6.1.4.1.2021.11.10.0=system, .1.3.6.1.4.1.2021.11.9.0=user

:

```
tracing logger: name=local\snmp_cpu, factory=local\snmpget, status=running (interval=1000ms),
  log count=8, last start=2014-01-10 19:56:08, last run=2014-01-10 19:56:15,
  last log=2014-01-10 19:56:15
 local\snmp_cpu: date=2014-01-10 19:56:16, logger=local\snmp_cpu, data={system=0, user=25}
 local\snmp_cpu: date=2014-01-10 19:56:17, logger=local\snmp_cpu, data={system=0, user=25}
```

4.10. SNMP

SNMP SNMP .

4.10.1. SNMP

4.10.1.1. SNMP

SNMP .

```
araqne> snmp.openTrapPort
 Description

open trap port

Arguments

 1. name: trap binding name (required)
 2. port: trap port (162 by default) (optional)
 3. address: trap address (0.0.0.0 by default) (optional)
```

* [ ]
* [ ] UDP , 162
* [ ] IP , 0.0.0.0

4.10.1.2. SNMP

```
araqne> snmp.closeTrapPort
```

```
 Description

close trap port

Arguments

1. name: trap binding name (required)

                .

#### 4.10.1.3.

     SNMP          .

araqne> snmp.trapBindings
 Trap Bindings
-----
default => name=default, listen=0.0.0.0/0.0.0.0:162, thread=1


#### 4.10.1.4.    SNMP

SNMP                                .       Ctrl-C    .
```

araqne〉 snmp.trace press ctrl-c to stop ————————— 127.0.0.1:62454 {1.3.6.1.4.1.33957.1.2.4=test, 1.3.6.1.4.1.33957.1.2.5=1, 1.3.6.1.4.1.33957.1.2.6=testrulegroup, 1.3.6.1.4.1.33957.1.2.20=http, 1.3.6.1.4.1.33957.1.2.7=eth0, 1.3.6.1.4.1.33957.1.2.1=1101, 1.3.6.1.4.1.33957.1.2.2=20111205225000+0900, 1.3.6.1.4.1.33957.1.2.3=31, 1.3.6.1.4.1.33957.1.1.1=ML 1.3.6.1.4.1.33957.1.1.2=1, 1.3.6.1.4.1.33957.1.2.8=eth1, 1.3.6.1.4.1.33957.1.2.9=/172.20.2.25, 1.3.6.1.4.1.33957.1.2.19=http, 1.3.6.1.4.1.33957.1.2.13=60018, 1.3.6.1.4.1.33957.1.2.14=80, 1.3.6.1.4.1.33957.1.2.11=/74.125.71.106, 1.3.6.1.4.1.33957.1.2.22=http, 1.3.6.1.4.1.33957.1.2.23=soul, 1.3.6.1.4.1.33957.1.2.18=172.20.2.25_60018_74.125.71.106_80_1234.pcap, 1.3.6.1.4.1.33957.1.2.21.1=r 1.3.6.1.4.1.33957.1.2.15=001122334455, 1.3.6.1.4.1.33957.1.2.16=001122334455} ~

4.10.2. SNMP 트랩 로거 설정

4.10.2.1. SNMP 트랩 로거 생성 및 시작 로거 생성은 로거의 유형과 관계없이 공통 인터페이스를 따릅니다. 아래의 명령을 사용하여 SNMP 트랩 로거를 생성할 수 있습니다:

예시) 1.2.3.4에서 전송된 SNMP 트랩 로그를 수신하는 설정

```
araqne> logapi.createLogger snmptrap local trapsample
 remote ip (required)? 1.2.3.4
 transformer (optional, enter to skip)?
 logger created: name=local\trapsample, factory=local\snmptrap, status=stopped (passive), log count=0, last
```

remote ip 설정 항목은 SNMP 트랩을 전송하는 장비의 IP 주소를 기입합니다. NAT 환경인 경우 로그프레소 서버 측에서 보이는 원격지 주소, 즉 NAT된 주소를 입력하면 됩니다.

로거를 생성했으면 아래의 명령으로 시작시킵니다.

```
araqne> logapi.startLogger local\trapsample
 logger started
```

4.10.2.2. SNMP 트랩 로그 저장 설정

수집되는 SNMP 트랩 로그를 저장하려면, 명시적으로 로그 저장 설정을 해야합니다.

```
araqne> logpresso.createLogger
 Description

create log archive

Arguments

 1. org domain: org domain (required)
 2. logger fullname: logger fullname (required)
 3. table name: destination table name, use logger fullname if not specified (optional)
 4. host: host name will be recorded to log data if specified (optional)
```

첫번째 매개변수는 localhost 도메인을, 두번째 매개변수는 로거의 전체 이름을 입력합니다. 세번째 매개변수는 저장할 테이블 이름을 입력합니다. 네번째 항목을 설정하면 해당 문자열을 로그에 _host 필드 이름으로 태깅합니다. 즉, 여러 장비의 로그를 한 테이블에 저장했을 때 태그로 각 호스트를 구분할 수 있도록 지원하는 것입니다.

예시) local₩trapsample 로거로 수집되는 모든 로그를 traps 테이블에 저장

```
araqne> logpresso.createLogger localhost local\trapsample traps
```

이제 `logapi.logger` 명령으로 로그 수신 상태를 확인한 후, `logdb.console` 명령으로 DB 콘솔에 접속하여 올바르게 SNMP 트랩이 저장되었는지 쿼리하여 확인합니다.

4.11. SNMP 트랩 로거 설정넷플로우 로그 수집 설정

넷플로우 서버와 로거를 이용하여 실시간으로 넷플로우 로그를 수집하고 테이블에 저장할 수 있습니다. netflow v5와 v9 패킷을 지원합니다.

4.11.1. 넷플로우 서버 관리

4.11.1.1. 넷플로우 포트 열기

```
araqne> flowmon.open
 Description

open netflow port

Arguments

 1. name: binding name (required)
 2. port: udp port for netflow collector (required)
```

- [필수] 바인딩 이름. 임의의 이름을 부여합니다.
- [필수] 넷플로우를 수신할 UDP 포트 번호

예시) default 라는 이름으로 9090 포트 개방

```
araqne> flowmon.open default 9090
 port opened
```

4.11.1.2. **넷플로우 포트 목록 조회** 현재 열려있는 넷플로우 포트 목록을 조회하려면 flowmon.bindings 명령을 실행합니다:

```
araqne> flowmon.bindings
 Port Bindings
-----
name=default, port=9090
```

4.11.1.3. 실시간 넷플로우 트레이스 넷플로우 패킷이 정상적으로 수신되고 있는지 확인하려고 할 때 실시간 트레이스 명령을 사용할 수 있습니다. 실시간 트레이스를 중지하려면 Ctrl-C를 입력합니다.

```
araqne> flowmon.trace default
 press ctrl-c to stop
--------
[127.0.0.1:60772] ver=5, count=30, sysuptime=223215, unixsecs=1369017127, unixnsecs=26, seq=696, engine_ty
 [127.0.0.1:60772] ver=9, count=3, sys_uptime=42212, unixsecs=1369122709, seq=0, source=106
 interrupted
```

4.11.1.4. 넷플로우 포트 닫기 지정된 이름의 넷플로우 포트를 닫습니다.

예시) default 넷플로우 포트 닫기

```
araqne> flowmon.close default
 port closed
```

4.11.2. 넷플로우 로거 설정

넷플로우 로거는 넷플로우 패킷을 수신한 후 개별 플로우 레코드에 대하여 로그를 발생시킵니다. 1개의 넷플로우 패킷이 수십 개의 플로우 레코드를 포함할 수 있습니다.

4.11.2.1. 넷플로우 로거 생성 및 시작 로거 생성은 로거의 유형에 관계없이 공통 인터페이스를 따릅니다. 아래의 명령을 사용하여 넷플로우 로거를 생성할 수 있습니다:

예시) 모든 넷플로우 패킷 수신

```
araqne> logapi.createLogger netflow local netflow-logger
 Source ID List (optional)?
 Field Name Filter (optional)?
 transformer (optional, enter to skip)?
 logger created: name=local\netflow-logger, factory=local\netflow, status=stopped (passive), log count=0, l
```

소스 ID 목록은 넷플로우 패킷을 전송하는 측에서 설정한 ID를 의미합니다. 입력하지 않으면 필터링 없이 모든 패킷을 수신합니다. 쉼표로 구분하여 여러 개의 ID를 입력할 수 있습니다. 또한 필드 이름 필터를 사용해서 전송된 플로우 레코드 중 특정한 필드만 저장하도록 설정할 수 있습니다. 마찬가지로 쉼표로 구분된 필드 이름들을 입력할 수 있습니다.

로거를 생성했으면 아래의 명령으로 시작시킵니다.

```
araqne> logapi.startLogger local\netflow-logger
 logger started
```

로거 상태 조회 명령으로 넷플로우 패킷이 서버에서 수신된 후 로거까지 도달했는지 확인할 수 있습니다:

```
araqne> logapi.logger local\netflow-logger
 Logger [local\netflow-logger]
----------
* Description: null
 * Logger Factory: local\netflow
 * Status: Running
 * Interval: 0ms
 * Last Log: 2013-06-15 17:54:47
 * Last Run: N/A
 * Log Count: 31

Configuration
-----
```

4.11.2.2. **넷플로우 로그 저장 설정** 이전 단계에서 생성한 로거가 넷플로우 데이터소스로서 기능한다고 하더라도, 명시적으로 로그 저장 설정을 해야만 수신된 넷플로우 로그가 테이블에 저장됩니다.

```
araqne> logpresso.createLogger
 Description

create managed logger

Arguments

 1. org domain: org domain (required)
 2. logger fullname: logger fullname (required)
 3. table name: destination table name, use logger fullname if not specified (optional)
 4. host: host name will be recorded to log data if specified (optional)
```

첫번째 매개변수는 localhost 도메인을, 두번째 매개변수는 로거의 전체 이름을 입력합니다. 세번째 매개변수는 저장할 테이블 이름을 입력합니다. 네번째 항목을 설정하면 해당 문자열을 로그에 _host 필드 이름으로 태깅합니다. 즉, 여러 장비의 로그를 한 테이블에 저장했을 때 태그로 각 호스트를 구분할 수 있도록 지원하는 것입니다.

예시) local₩netflow-logger에서 발생한 모든 로그를 netflows 테이블에 저장

```
araqne> logpresso.createLogger localhost local\netflow-logger netflows
 created
```

이제 logapi.logger 명령으로 수신 상태를 확인한 후, logdb.console 명령으로 DB 콘솔에 접속하여 올바르게 넷플로우 로그가 저장되었는지 쿼리하여 확인합니다.

```
araqne@logdb> query table netflows | stats count, sum(octet_count), sum(packet_count)
 {count=62, sum(octet_count)=298138, sum(packet_count)=408}
 total 1 rows, elapsed 0.1s
```

4.12. JDBC 수집 설정

logpresso-jdbc 0.1.0 버전부터 지원

JDBC 로거는 SQL 쿼리를 사용하여 데이터베이스의 테이블나 뷰에서 데이터를 수집하려고 할 때 사용합니다. JDBC 로거는 아래와 같은 설정을 입력받습니다: * [필수] JDBC 프로파일: JDBC 접속 프로파일의 이름을 입력합니다. (JDBC 연동 설정 바로가기) * [필수] SQL: 데이터 수집에 사용할 SQL 문장을 입력합니다. $where.*[]$:where 매크로에 삽입될 조건절을 입력합니다. 물음표(?)를 위치 지정자 (place holder)로 사용할 수 있습니다. 위치 지정자는 마지막 기준 컬럼 값으로 대치됩니다. 입력할 때 where 문자열까지 포함해야 합니다. * [필수] 기준 컬럼: 매 조회 시 마지막으로 수집했던 행 이후부터 가져올 수 있도록, 검색 기준이 되는 컬럼 이름을 입력합니다. 가령 시퀀스, IDENTITY, auto_increment로 지정된 컬럼, 혹은 증가하는 타임스탬프 컬럼의 이름을 입력합니다. * [선택] 날짜 컬럼 이름: 데이터가 수집된 날짜를 가지고 있는 컬럼의 이름을 지정합니다. 지정되지 않을 경우 로거가 수집한 날짜를 저장합니다. * [선택] 날짜 형식: 날짜 컬럼이 SQL 시간 형식이 아닌 문자열이라면 날짜의 형태를 지정할 수 있습니다(예. yyyyMMdd HH:mm:ss).

SQL은 아래의 사항들을 고려하여 작성합니다: * 기준 컬럼으로 검색할 때 인덱스를 타는지 확인합니다. 인덱스를 타지 않는다면 수천만건 이상 들어있는 테이블의 경우 지속적으로 테이블 풀스캔 부하가 걸릴 수 있습니다. * 한 번에 가져오는 갯수를 제한합니다. 가령, 오라클의 경우 rownum을 사용하여 가져올 행 갯수를 제한할 수 있습니다. 이를 고려하지 않으면 초기 적재 시에 너무 많은 데이터를 한 번에 가져오려고 시도하면서 문제가 발생할 수 있습니다. JDBC 로거는 지정된 주기별로 쿼리를 수행하지만, 한 번 수집할 때 더 이상 새로운 값이 없을 때까지 쿼리를 반복 수행하므로 가져오는 행 갯수를 제한하는 것이 좋습니다. * 필요한 컬럼만 SELECT 절에 명시적으로 지정합니다. JDBC 로거는 조회되는 모든 컬럼 값을 키/값 형태로 수집합니다. 불필요한 컬럼을 제외하면 더 나은 성능을 기대할 수 있습니다.

오라클 테이블 데이터 수집 설정 예시

```
worldcup-schema
araqne@bombom demo> logapi.createLogger jdbc local dblog
```

```
JDBC Profile (required)? worldcup
SQL (required)? select * from (select * from worldcup_weblogs $where order by id) t where rownum < 10000
Where clause (required)? where id > ?
Column name (required)? id
transformer (optional, enter to skip)?
logger created: name=local\dblog, factory=local\jdbc, status=stopped (interval=0ms), log count=0, last star
```

이와 같이 설정된 상태에서 JDBC 로거는 다음과 같이 동작합니다: * 1.처음에는 $where 절이 빈 문자열로 치환되어 아래와 같은 쿼리를 수행합니다. ~ select * from (select * from worldcup_weblogs order by id) t where rownum 〈 10000 ~

- 2.두번째 쿼리부터는 저장된 마지막 id 컬럼 값을 기준으로 다음과 같은 쿼리를 수행합니다. ~ select * from (select * from worldcup_weblogs where id 〉 9999 order by id) t where rownum 〈 10000 ~
- 3.더 이상 데이터가 조회되지 않으면, 다음 실행 주기가 올 때까지 대기합니다.

아래는 JDBC를 통해 수집된 로그가 저장된 테이블을 쿼리한 결과입니다.

```
araqne@logdb> query table limit=3 worldcup
{ID=13737450, LINE=10.0.1.130 - - [14/May/1998:14:30:00 +0000] "GET /english/history/past_cups/images/poste
{ID=13737449, LINE=10.3.188.218 - - [14/May/1998:14:30:00 +0000] "GET /images/102325s.gif HTTP/1.1" 200 776
{ID=13737448, LINE=10.0.4.254 - - [14/May/1998:14:30:00 +0000] "GET /images/home_bg_stars.gif HTTP/1.0" 304
total 3 rows, elapsed 0.1s
```

4.13. FTP 디렉터리 로그 파일 수집 설정

FTP 디렉토리 와치 (ftp-dirwatch) 로거는 FTP를 통해 원격지에서 롤링되지 않는 텍스트 로그 파일을 일정 주기마다 수집하려고 할 때 사용합니다. 가령 일자별 혹은 시간대별로 순차 생성되는 로그 파일을 수집할 때 사용합니다. FTP 디렉토리 와치 로거는 아래와 같은 설정을 입력받습니다: * [필수] FTP 프로파일: FTP 접속 설정 프로파일의 이름을 입력합니다. (FTP 연동 설정 바로가기) * [필수] 디렉터리 경로: 로그 파일이 위치하는 파일시스템 경로를 입력합니다. * [필수] 파일이름 정규표현식 패턴: 디렉터리 경로에 존재하는 파일 중 이름이 정규표현식 패턴에 일치하는 경우에만 수집합니다. 정규표현식 그룹을 쓰는 경우 파일 이름에서 날짜 문자열을 추출합니다. * [선택] 날짜 추출 정규표현식 패턴: 로그에서 날짜 문자열을 추출합니다. 정규표현식 그룹으로 묶인 모든 부분을 이어붙여서 하나의 날짜 문자열을 만들어냅니다. 파일이름 정규표현식의 그룹으로 추출된 날짜문자열은 가장 앞 부분에 위치합니다. * [선택] 날짜 파싱 포맷: 날짜 문자열을 파싱하는데 사용할 날짜 포맷을 설정합니다. (예: yyyy-MM-dd HH:mm:ss) * [선택] 로그 시작 정규식: 로그의 시작 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄

단위로 읽어들입니다. * [선택] 로그 끝 정규식: 로그의 끝 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다. * [선택] 문자집합: 문자집합 코드를 입력합니다. 기본값은 utf-8입니다.

여러 개의 파일이 수집 대상인 경우, 파일의 절대 경로를 사전순으로 정렬하여 순서대로 읽어들입니다. 이하에서는 여러가지 사용 예를 살펴보도록 하겠습니다.

예시 1) /var/log/examples 디렉터리에 example.log.yyyy-MM-dd-HH 로그 파일이 순차 생성되는 경우

0;20130615 141618;192.168.0.10;E002;20130615 141618;192.168.0.11; * 디렉터리 경로: /var/log/examples * 파일이름 정규표현식 패턴: example.log. 날짜 추출 정규표현식 패턴: □+[(8[6]: * 날짜 파싱 포맷: yyyyMMdd HHmmss

예시 2) /logdata/c6509 디렉터리에 c6509.log.yyyyMMdd 로그 파일이 쌓이는 경우

May 01 23:59:59: %LINK-SP-3-UPDOWN: Interface GigabitEthernet8/15, changed state to down * 디렉터리 경로: /logdata/c6509 * 파일이름 정규표현식 패턴: c6509.log.(8□* 날짜 추출 정규표현식 패턴: [ ]* 2[(2)(2)(2□* 날짜 파싱 포맷: yyyyMMddHHmmss(파일이름 정규표현식으로 추출된 yyyyMMdd 그룹 1개와 날짜 추출 정규표현식에서 추출된 HH, mm, ss 그룹 3개가 하나로 병합되어 파싱됩니다.)

4.14. FTP 로테이션 로그 파일 수집 설정

FTP 로테이션 (ftp-rotation) 로거는 FTP 서버를 통해서 주기적으로 로테이션 되는 텍스트 로그 파일을 일정 주기마다 수집하려고 할 때 사용합니다. 흔히 리눅스 서버에서는 logrotate 프로그램을 사용하여 일정 주기로 기존 파일의 이름을 변경하여 백업 보관하고 로그 파일을 새로 생성합니다. FTP 로테이션 로거는 이런 시나리오에서 사용되며, 다음과 같은 설정을 입력받습니다: * [필수] FTP 프로파일: FTP 접속 설정 프로파일의 이름을 입력합니다. (FTP 연동 설정 바로가기) * [필수] 파일 경로: 주기적으로 로테이션 되는 텍스트 로그 파일의 절대 경로를 입력합니다. * [선택] 문자집합: 텍스트 파일 해석에 사용할 문자집합 코드를 입력합니다. 기본값은 utf-8입니다. * [선택] 날짜 추출 정규표현식 패턴: 로그에서 날짜 문자열을 추출합니다. 정규표현식 그룹으로 묶인 모든 부분을 이어붙여서 하나의 날짜 문자열을 만들어냅니다. * [선택] 날짜 파싱 포맷: 날짜 문자열을 파싱하는데 사용할 날짜 포맷을 설정합니다. (예: yyyy-MM-dd HH:mm:ss) * [선택] 날짜 로케일: 날짜 문자열의 로케일. 가령 날짜 파싱 포맷의 지시자 중 MMM의 해석은 로케일에 따라 “Jan” 혹은 “1월”로 해석됩니다. 기본값은 en입니다. * [선택] 로그 시작 정규식: 로그의 시작 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다. * [선택] 로그 끝 정규식: 로그의 끝 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다.

예시 1) /var/log/httpd/access_log 파일이 로테이션 되는 경우

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

- 파일 경로: /var/log/httpd/access_log

- 날짜 추출 정규표현식 패턴: [(.*?)]
- 날짜 파싱 포맷: dd/MMM/yyyy:HH:mm:ss Z
- 날짜 로케일: en

예시 2) /araqne/log/araqne.log 파일이 로테이션 되는 경우

```
[2013-06-14 13:55:02,186] INFO (LogIndexerEngine) - logpresso index: counter reset thread started
[2013-06-14 13:55:40,647]  WARN (ScriptRunner) - script runner:
java.lang.reflect.InvocationTargetException
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
    at org.araqne.console.ScriptRunner.invokeScript(ScriptRunner.java:209)
    at org.araqne.console.ScriptRunner.run(ScriptRunner.java:190)
    at java.lang.Thread.run(Unknown Source)
Caused by: java.lang.InterruptedException
    at org.araqne.console.ReadLineHandler.read(ReadLineHandler.java:156)
    at org.araqne.console.ReadLineHandler.getLine(ReadLineHandler.java:46)
    at org.araqne.console.ConsoleInputStream.readLine(ConsoleInputStream.java:56)
    at org.araqne.script.ScriptContextImpl.readLine(ScriptContextImpl.java:194)
    at org.araqne.log.api.impl.LogApiScript.setOption(LogApiScript.java:629)
    at org.araqne.log.api.impl.LogApiScript.createParser(LogApiScript.java:99)
    ... 7 more
```

- 파일 경로: /araqne/log/araqne.log
- 날짜 추출 정규표현식 패턴: [(.*),
- 날짜 파싱 포맷: yyyy-MM-dd HH:mm:ss
- 로그 시작 정규식: [4□2□2□2□2□2□3□

4.15. SFTP 디렉터리 로그 파일 수집 설정

SFTP 디렉토리 와치 (sftp-dirwatch) 로거는 SFTP를 통해 원격지에서 롤링되지 않는 텍스트 로그 파일을 일정 주기마다 수집하려고 할 때 사용합니다. 가령 일자별 혹은 시간대별로 순차 생성되는 로그 파일을 수집할 때 사용합니다. SFTP 디렉토리 와치 로거는 아래와 같은 설정을 입력받습니다: * [필수] SSH 프로파일: SSH 접속 설정 프로파일의 이름을 입력합니다. (SSH 연동 설정 바로가기) * [필수] 디렉터리 경로: 로그 파일이 위치하는 파일시스템 경로를 입력합니다.

* [필수] 파일이름 정규표현식 패턴: 디렉터리 경로에 존재하는 파일 중 이름이 정규표현식 패턴에 일치하는 경우에만 수집합니다. 정규표현식 그룹을 쓰는 경우 파일 이름에서 날짜 문자열을 추출합니다. * [선택] 날짜 추출 정규표현식 패턴: 로그에서 날짜 문자열을 추출합니다. 정규표현식 그룹으로 묶인 모든 부분을 이어붙여서 하나의 날짜 문자열을 만들어냅니다. 파일이름 정규표현식의 그룹으로 추출된 날짜문자열은 가장 앞 부분에 위치합니다. * [선택] 날짜 파싱 포맷: 날짜 문자열을 파싱하는데 사용할 날짜 포맷을 설정합니다. (예: yyyy-MM-dd HH:mm:ss) * [선택] 로그 시작 정규식: 로그의 시작 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다. * [선택] 로그 끝 정규식: 로그의 끝 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다. * [선택] 문자집합: 문자집합 코드를 입력합니다. 기본값은 utf-8입니다.

여러 개의 파일이 수집 대상인 경우, 파일의 절대 경로를 사전순으로 정렬하여 순서대로 읽어들입니다. 이하에서는 여러가지 사용 예를 살펴보도록 하겠습니다.

예시 1) /var/log/examples 디렉터리에 example.log.yyyy-MM-dd-HH 로그 파일이 순차 생성되는 경우

0;20130615 141618;192.168.0.10;E002;20130615 141618;192.168.0.11; * 디렉터리 경로: /var/log/examples * 파일이름 정규표현식 패턴: example.log. 날짜 추출 정규표현식 패턴: □+[0-8]{6}; * 날짜 파싱 포맷: yyyyMMdd HHmmss

예시 2) /logdata/c6509 디렉터리에 c6509.log.yyyyMMdd 로그 파일이 쌓이는 경우

May 01 23:59:59: %LINK-SP-3-UPDOWN: Interface GigabitEthernet8/15, changed state to down * 디렉터리 경로: /logdata/c6509 * 파일이름 정규표현식 패턴: c6509.log.(8□* 날짜 추출 정규표현식 패턴: [ ]* 2(2):(2):(2)□* 날짜 파싱 포맷: yyyyMMddHHmmss (파일이름 정규표현식으로 추출된 yyyyMMdd 그룹 1개와 날짜 추출 정규표현식에서 추출된 HH, mm, ss 그룹 3개가 하나로 병합되어 파싱됩니다.)

4.16. SFTP 로테이션 로그 파일 수집 설정

SFTP 로테이션 (sftp-rotation) 로거는 SFTP 서버를 통해서 주기적으로 로테이션 되는 텍스트 로그 파일을 일정 주기마다 수집하려고 할 때 사용합니다. 흔히 리눅스 서버에서는 logrotate 프로그램을 사용하여 일정 주기로 기존 파일의 이름을 변경하여 백업 보관하고 로그 파일을 새로 생성합니다. SFTP 로테이션 로거는 이런 시나리오에서 사용되며, 다음과 같은 설정을 입력받습니다: * [필수] SSH 프로파일: SSH 접속 설정 프로파일의 이름을 입력합니다. (SSH 연동 설정 바로가기) * [필수] 파일 경로: 주기적으로 로테이션 되는 텍스트 로그 파일의 절대 경로를 입력합니다. * [선택] 문자집합: 텍스트 파일 해석에 사용할 문자집합 코드를 입력합니다. 기본값은 utf-8입니다. * [선택] 날짜 추출 정규표현식 패턴: 로그에서 날짜 문자열을 추출합니다. 정규표현식 그룹으로 묶인 모든 부분을 이어붙여서 하나의 날짜 문자열을 만들어냅니다. * [선택] 날짜 파싱 포맷: 날짜 문자열을 파싱하는데 사용할 날짜 포맷을 설정합니다. (예: yyyy-MM-dd HH:mm:ss) * [선택] 날짜 로케일: 날짜 문자열의 로케일. 가령 날짜 파싱 포맷의 지시자 중 MMM의 해석은 로케일에 따라 “Jan” 혹은 “1월”로 해석됩니다. 기본값은 en입니다. * [선택] 로그 시작 정규식: 로그의 시작 부분을 인식하는 정규표현식을 지정합니다.

멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다. * [선택] 로그 끝 정규식: 로그의 끝 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다.

예시 1) /var/log/httpd/access_log 파일이 로테이션 되는 경우

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

- 파일 경로: /var/log/httpd/access_log
- 날짜 추출 정규표현식 패턴: [(.*?)]
- 날짜 파싱 포맷: dd/MMM/yyyy:HH:mm:ss Z
- 날짜 로케일: en

예시 2) /araqne/log/araqne.log 파일이 로테이션 되는 경우

```
[2013-06-14 13:55:02,186] INFO (LogIndexerEngine) - logpresso index: counter reset thread started
[2013-06-14 13:55:40,647]  WARN (ScriptRunner) - script runner:
java.lang.reflect.InvocationTargetException
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
    at org.araqne.console.ScriptRunner.invokeScript(ScriptRunner.java:209)
    at org.araqne.console.ScriptRunner.run(ScriptRunner.java:190)
    at java.lang.Thread.run(Unknown Source)
Caused by: java.lang.InterruptedException
    at org.araqne.console.ReadLineHandler.read(ReadLineHandler.java:156)
    at org.araqne.console.ReadLineHandler.getLine(ReadLineHandler.java:46)
    at org.araqne.console.ConsoleInputStream.readLine(ConsoleInputStream.java:56)
    at org.araqne.script.ScriptContextImpl.readLine(ScriptContextImpl.java:194)
    at org.araqne.log.api.impl.LogApiScript.setOption(LogApiScript.java:629)
    at org.araqne.log.api.impl.LogApiScript.createParser(LogApiScript.java:99)
    ... 7 more
```

- 파일 경로: /araqne/log/araqne.log
- 날짜 추출 정규표현식 패턴: [(.*),
- 날짜 파싱 포맷: yyyy-MM-dd HH:mm:ss
- 로그 시작 정규식: [4E2E2[2[2[2[2,3[

4.17. SSH 표준 출력 수집 설정

SSH 표준 출력 수집 (ssh-exec) 로거는 SSH를 통해 원격으로 지정된 명령어의 실행 결과를 수집합니다. 원격지에 설치된 프로그램 혹은 스크립트 등을 실행하여 표준 출력으로 나오는 문자열 전체를 하나의 로그로 수집합니다. SSH 표준 출력 수집 로거는 아래와 같은 설정을 입력받습니다: * [필수] SSH 프로파일: SSH 접속 설정 프로파일의 이름을 입력합니다. (SSH 연동 설정 바로가기) * [필수] 명령어: SSH 쉘 접속 후 실행할 명령어를 입력합니다.

로그 수집 시 매번 접속을 다시 수행하므로, 의도된 간격보다 접속 지연 시간에 따라 수집 간격이 길어질 수 있습니다.

예시) 리눅스 호스트의 load average 로그 수집 * SSH 프로파일 이름: 수집 대상 호스트에 대한 SSH 접속 프로파일 이름을 입력합니다. * 명령어: cat /proc/loadavg

실시간 로그 트레이스 결과

```
araqne> logapi.trace local\loadavg
tracing logger: name=local\loadavg, factory=local\ssh-exec, status=running (interval=1000ms), log count=0,
local\loadavg: date=2013-08-18 20:07:56, logger=local\loadavg, data={line=0.38 0.17 0.11 1/358 18042}
local\loadavg: date=2013-08-18 20:08:14, logger=local\loadavg, data={line=0.27 0.16 0.11 1/357 18101}
local\loadavg: date=2013-08-18 20:08:32, logger=local\loadavg, data={line=0.21 0.
```

4.18. JMX 수집 설정

araqne-core 2.6.3, araqne-logdb-jmx 0.1.0 버전부터 지원

원격 JMX 로거는 자바 RMI (Remote Method Invocation) 프로토콜을 사용하여 원격 JVM의 상태 정보를 수집합니다. 모니터링 대상 JVM을 기동할 때 미리 아래와 같은 설정으로 RMI를 활성화시키고, 방화벽에서 RMI 포트를 허용해야 합니다.

4.18.1. 모니터링 대상의 원격 JMX 허용과 관련된 JVM 실행 옵션 예시

- -Dcom.sun.management.jmxremote
- -Dcom.sun.management.jmxremote.port=8999
- -Dcom.sun.management.jmxremote.ssl=false
- -Dcom.sun.management.jmxremote.password.file=jmxremote.password
- -Dcom.sun.management.jmxremote.access.file=jmxremote.access

jmxremote.password 파일과 jmxremote.access 파일은 $JAVA_HOME/lib/management/jmxremote.password.template$JAVA_H
파일을 각각 복사하여 수정 후 사용하시면 됩니다. jmxremote.password 파일 최하단에는 아래와 같은 내용이 있습니다.

```
# monitorRole QED
# controlRole R&D
```

monitorRole의 주석을 해제하고 QED를 적절한 강도의 암호로 변경합니다. 또한, jmxremote.password 파일과 jmxremote.access 파일의 읽기 권한을 제한적으로 설정해야 JVM이 정상적으로 부팅됩니다. 읽기 권한 설정은 아래와 같이 합니다. * 유닉스 계열 ~ chmod 600 jmxremote.password chmod 600 jmxremote.access ~

- 윈도우즈 ~ cacls jmxremote.password /P 계정이름:R cacls jmxremote.access /P 계정이름:R ~

4.18.2. 원격 JMX 로거 설정

- [필수] 호스트 주소: 모니터링 대상의 도메인 이름 혹은 IP 주소를 입력합니다.
- [필수] 포트: 모니터링 대상의 JMX 포트 번호를 입력합니다. 위의 예시에서는 8999에 해당됩니다.
- [필수] 사용자 계정: 모니터링 대상의 JMX 계정을 입력합니다. 위의 예시에서는 monitorRole에 해당됩니다.
- [필수] 암호: 모니터링 대상의 JMX 암호를 입력합니다.
- [필수] 개체 이름: JMX 개체 이름(Object Name)을 입력합니다. 예를 들어, 적재된 클래스 갯수를 보려면 java.lang:type=Classloading을 입력합니다.
- [선택] 속성 이름 목록: 로그로 수집할 속성 이름을 쉼표로 구분하여 명시적으로 지정할 수 있습니다. 지정하지 않으면 전체 속성 정보가 수집됩니다.

4.18.3. 원격 JMX 조회 테스트

쉘에서 jmx 명령어를 사용하여 간단하게 사용 가능한 개체 이름의 목록과 속성 정보를 확인할 수 있습니다.

1) 개체 이름 목록 조회 예시

```
araqne@bombom demo> jmx.objectNames localhost:8999 monitorRole QED
 Connecting to host…
java.lang:type=Memory
 java.lang:type=MemoryPool,name=PS Eden Space
 java.lang:type=MemoryPool,name=PS Survivor Space
 java.lang:type=GarbageCollector,name=PS MarkSweep
 java.lang:type=MemoryPool,name=Code Cache
 java.lang:type=Runtime
 java.lang:type=ClassLoading
 java.nio:type=BufferPool,name=direct
```

```
java.lang:type=Threading
java.nio:type=BufferPool,name=mapped
java.util.logging:type=Logging
java.lang:type=Compilation
com.sun.management:type=HotSpotDiagnostic
java.lang:type=MemoryPool,name=PS Perm Gen
java.lang:type=GarbageCollector,name=PS Scavenge
java.lang:type=OperatingSystem
java.lang:type=MemoryPool,name=PS Old Gen
java.lang:type=MemoryManager,name=CodeCacheManager
JMImplementation:type=MBeanServerDelegate
```

2) 스레딩 속성 정보 조회 예시

```
araqne@bombom demo> jmx.attrs localhost:8999 monitorRole QED java.lang:type=Threading
 Connecting to host…
ThreadAllocatedMemoryEnabled = true
 ThreadAllocatedMemorySupported = true
 ThreadContentionMonitoringEnabled = false
 CurrentThreadCpuTimeSupported = true
 ObjectMonitorUsageSupported = true
 ThreadContentionMonitoringSupported = true
 AllThreadIds = [J@1f49f731
 CurrentThreadCpuTime = 78000500
 CurrentThreadUserTime = 78000500
 ThreadCount = 72
 TotalStartedThreadCount = 147
 ThreadCpuTimeSupported = true
 ThreadCpuTimeEnabled = true
 DaemonThreadCount = 17
 PeakThreadCount = 83
 SynchronizerUsageSupported = true
 ObjectName = java.lang:type=Threading
```

5장. 실시간 로그 필터링, 태깅

5.1. 로그 트랜스포머 설정

원본 로그에 추가적인 데이터를 태깅하거나 원본 로그에 특정한 변형을 가하고 싶은 경우에 트랜스포머를 사용할 수 있습니다.

5.1.1. 트랜스포머 유형 목록 조회

```
araqne> logapi.transformerFactories
 Log Transformer Factories
---------
keyvalue: add key=value pairs to original log
```

5.1.2. 트랜스포머 생성

```
araqne> logapi.createTransformer
 Description

create transformer profile

Arguments

 1. profile name: transformer profile name (required)
 2. factory name: transformer factory name (required)
```

- [필수] 임의의 유일한 트랜스포머 이름
- [필수] 트랜스포머 유형 이름 (예: keyvalue)

예시) keyvalue 트랜스포머를 이용하여 원본 로그에 origin=1.2.3.4와 device_type=firewall 필드 추가

```
araqne> logapi.createTransformer sample keyvalue
 tags (required)? origin=1.2.3.4,device_type=firewall
 created
```

5.1.3. 트랜스포머 목록 조회

아래와 같이 생성된 트랜스포머 목록을 조회할 수 있습니다:

```
araqne> logapi.transformers
 Log Transformer Profiles
---------
name=sample, factory=keyvalue, configs={tags=origin=1.2.3.4,device_type=firewall}
```

5.1.4. 트랜스포머 삭제

지정된 트랜스포머를 삭제합니다:

예시) sample 이름의 트랜스포머 삭제

```
araqne> logapi.removeTransformer sample
 removed
```

5.2. 로그 트랜스포머 설정정규표현식을 이용한 로그 필터링

(araqne-log-api 2.3.1부터 지원)

정규표현식 필터링을 이용하여 원하는 로그만 선택적으로 수집할 수 있습니다. 설정된 정규표현식으로 매칭되지 않는 경우 로그를 버립니다. regex-filter 트랜스포머의 설정은 아래와 같습니다: * [필수] 정규표현식: line 필드 문자열에서 검색할 정규표현식을 입력합니다. * [선택] 결과 반전: true 혹은 false. true를 입력하면 정규표현식 매칭 결과를 반전합니다.

아래 예시는 10.26. 문자열을 포함하는 로그만 선별적으로 수집하는 트랜스포머를 설정합니다:

```
araqne> logapi.createTransformer 1026filter regex-filter
 regex (required)? 10\.26\.
 inverse match (optional)?
 created
```

이후 로거 설정 시 10.26.을 포함하는 로그만 수집하도록 설정할 수 있습니다:

```
araqne> logapi.createLogger dirwatch local sample
 Directory path (required)? d:/
```

```
 Filename pattern (required)? sample.txt
 Date pattern (optional)?
 Date format (optional)?
 Date locale (optional)?
 New log designator (Regex) (optional)?
 Charset (optional)?
 transformer (optional, enter to skip)? 1026filter
logger created: name=local\sample, factory=local\dirwatch, status=stopped (interval=0ms),
    log count=0, last start=null, last run=null, last log=null
```

버려진 로그 갯수는 아래와 같이 확인할 수 있습니다:

```
araqne> logapi.logger local\sample
 Logger [local\sample]
--------
* Description: null
 * Logger Factory: local\dirwatch
 * Status: Running
 * Interval: 1000ms
 * Last Log: 2013-07-09 23:45:40
 * Last Run: 2013-07-10 00:05:23
 * Log Count: 10
 * Drop Count: 8
```

위의 예시는 총 10건의 로그를 수집했지만 8건의 로그를 버렸다는 의미입니다.

6장. 관리되는 로거

6.1. 로그 저장 설정

로그 수집 설정을 통해 시스템에 데이터가 공급되면, 관리되는 로거 설정을 통해 로그를 테이블에 저장하도록 설정할 수 있습니다.

테이블에 저장하도록 설정할 필요가 없더라도, 패시브 및 액티브 로거들의 시작 순서를 맞추려면 관리되는 로거로 설정해야 합니다. 가령, selector 로거에 대하여 테이블 저장이 설정된 경우, 데이터 원본에 해당되는 액티브 로거를 관리되는 로거로 설정하지 않으면 로거 시작 순서로 인하여 부팅 시 일부 로그가 유실될 수 있습니다.

6.2. 관리되는 로거 설정

6.2.1. 관리되는 로거 목록 조회

```
araqne> logpresso.loggers
```

- [선택] 이름 필터: 필터 텍스트 입력 시 해당 텍스트를 포함한 관리되는 로거들만 표시합니다.

현재 설정된 모든 관리되는 로거를 표시합니다. 각 로거별로 매핑된 테이블, 테이블에 저장할 때 사용할 호스트 태그, 프라이머리 및 백업 로거 속성을 확인할 수 있습니다.

6.2.2. 관리되는 로거 생성

관리되는 로거를 생성합니다.

```
araqne> logpresso.createLogger
```

- [필수] 로거 이름: 생성할 관리되는 로거 이름을 입력합니다. 이름공간₩이름 형식을 사용합니다.
- [선택] 테이블 이름: 로거에서 수집되는 로그들을 테이블에 저장하려면 테이블 이름을 입력합니다. 테이블이 존재하지 않으면 기본 설정으로 테이블이 자동 생성됩니다.
- [선택] 호스트: 테이블 저장 시 _host 필드로 추가할 호스트 태그 값을 입력합니다.

6.2.3. 관리되는 로거 삭제

지정된 이름의 관리되는 로거를 삭제합니다.

```
araqne> logpresso.removeLogger
```

- [필수] 로거 이름: 삭제할 관리되는 로거 이름

6.3. 로그 수집 HA 설정

페더레이션 설정된 노드의 관리되는 로거와 묶어서 액티브-스탠바이 구성을 만들 수 있습니다. 프라이머리 로거의 노드가 일정 시간동안 무응답 시 백업 로거가 시작됩니다. 반대로, 백업 로거가 동작하는 상태에서 프라이머리 로거가 복원되면 백업 로거가 즉시 정지하고 프라이머리 로거가 다시 시작합니다.

백업 로거는 초 단위로 프라이머리 로거의 마지막 상태 정보를 복제합니다. 또한 프라이머리 로거가 복구될 때에 백업 로거의 마지막 상태를 복제하여 동작을 재개합니다. 이하에서는 시스로그를 양쪽 노드로 전송하는 상황에서 액티브-스탠바이로 수집 HA를 구성하는 가장 단순한 시나리오를 설명합니다.

6.3.1. 프라이머리 로거 설정

로컬의 관리되는 로거를 프라이머리 역할로 돌립니다.

```
araqne> logpresso.runAsPrimaryLogger
```

- 로컬 로거: 이름공간₩이름 형식으로 로컬의 관리되는 로거 이름을 입력합니다.
- 백업 로거: 노드₩이름공간₩이름 형식으로 백업 역할의 관리되는 로거 이름을 입력합니다.

로그프레소 재기동 시 프라이머리 로거와 매핑된 백업 로거의 마지막 상태 정보를 복제합니다.

6.3.2. 백업 로거 설정

로컬의 관리되는 로거를 백업 역할로 돌립니다.

```
araqne> logpresso.runAsBackupLogger
```

- 로컬 로거: 이름공간₩이름 형식으로 로컬의 관리되는 로거 이름을 입력합니다.
- 프라이머리 로거: 노드₩이름공간₩이름 형식으로 프라이머리 역할의 관리되는 로거 이름을 입력합니다.

백업 로거는 상시 프라이머리 로거의 마지막 상태 정보를 복제하고 모니터링 하다가, 프라이머리 로거의 노드 다운 시 시작하며 로그 수집을 이어갑니다.

6.3.3. 스탠드얼론 로거 설정

기존의 프라이머리, 백업 로거 설정을 삭제하고 단독으로 동작하는 관리되는 로거로 되돌립니다.

```
araqne> logpresso.runAsStandaloneLogger
```

- 로컬 로거: 이름공간₩이름 형식으로 로컬의 관리되는 로거 이름을 입력합니다.

7장. 실시간 로그 출력 설정

7.1. 롤링 로그 파일 쓰기 설정

(araqne-log-api 2.2.6부터 지원)

로그프레소에서 수집되는 로그를 특정 로그 파일에 실시간으로 출력할 수 있습니다. 또한 일정 파일 크기에 도달하면 자동으로 번호를 붙여서 백업 파일을 생성하고, 설정한 갯수만큼의 백업 파일만 유지하도록 할 수 있습니다. tofile 로거는 아래와 같은 설정을 입력받습니다: * [필수] 원본 로거: 파일에 쓸 로그를 수집하는 원본 로거 이름 (namespace₩name 형식) * [필수] 파일 경로: 파일의 경로 * [필수] 최대 파일 크기: 백업본 파일을 생성하는 기준 파일 크기 (바이트 단위) * [선택] 최대 백업 갯수: 백업본 파일 유지 갯수, 기본값은 1 * [선택] 파일 문자집합: 기본값은 utf-8

가령 최대 파일 크기가 10485760인 경우 (10MB), 파일 크기가 10MB를 넘기 직전에 확장자에 .N이 붙은 백업본 파일로 이름을 변경한 후 새로운 파일을 열고 쓰기 시작합니다. 만약, 기존 백업본 .1이 있는 상태에서 파일 크기를 넘게 되면, 기존의 .1은 .2로 이름이 바뀌고 쓰고 있던 파일 이름이 .1로 변경됩니다. 이 때 만약 최대 백업 갯수를 초과하게 되면 가장 오래된 백업본 파일이 삭제됩니다. 가령 최대 백업 갯수가 1개인 경우, .1 파일은 삭제되고 쓰고 있던 파일 이름이 .1로 변경됩니다.

예시) 최대 파일 크기가 10485760이고 최대 백업 갯수가 1인 경우

그림 2: 롤링로그 파일 스위칭 화면

7.2. 시간대별 파일 쓰기 설정

(araqne-log-api 2.3.2부터 지원)

totimefile 로거는 로그프레소에서 수집되는 로그를 특정 로그 파일에 실시간으로 시간 또는 일 단위로 출력할 수 있습니다. totimefile 로거는 아래와 같은 설정을 입력받습니다: * [필수] 원본 로거: 파일에 쓸 로그를 수집하는 원본 로거 이름 (namespace₩name 형식) * [필수] 파일 경로: 로그 파일 경로 (파일 이름 끝에 시간을 추가하여 생성하게 됨) * [필수] 파일 교체 주기: day (일) 혹은 hour (시간) * [선택] 문자집합: 출력할 파일 인코딩, 기본값 utf-8

line 필드가 있는 경우에는 line 필드 값만 줄 단위로 기록하며, 그렇지 않은 경우에는 key1="value1" , key2="value2" 와 같은 방식으로 모든 키/값 쌍을 기록합니다. 운영체제가 윈도우인 경우에는 개행 시 캐리지 리턴(CR)을 추가합니다.

예시) 일 단위로 d:/output 디렉터리 위치에 sample.log.yyyy-MM-dd 파일을 생성하는 로거 생성

```
araqne> logapi.createLogger totimefile local timefile-sample
 Source logger name (required)? local\source
 file path (required)? d:/output/sample.log
 rotate interval (required)? day
 charset (optional)? utf-8
 transformer (optional, enter to skip)?
```

```
logger created: name=local\timefile-sample, factory=local\totimefile, status=stopped (passive), log count
```

로거를 시작하면, 그 때부터 원본 로거로 수집되는 모든 로그에 대한 파일 쓰기를 시작합니다.

7.3. 시스로그 전송자 로거

시스로그 전송자 (syslog-send) 로거는 다른 로거에서 수집하는 데이터를 시스로그 서버로 전송합니다. 시스로그 전송자 로거는 아래와 같은 설정을 입력 받습니다. * [필수] 원본 로거 이름: 네임스페이스를 포함한 원본 로거 이름을 입력합니다. * [필수] 시스로그 서버 IP: 시스로그 서버의 IP를 입력받습니다. * [선택] 시스로그 서버 포트: 시스로그 서버의 포트를 지정합니다. 지정하지 않을 경우 514로 지정됩니다. * [선택] 인코딩: 데이터의 인코딩 형식을 지정합니다. 지정하지 않을 경우 UTF-8로 지정됩니다. * [선택] 필드: 전송할 필드 이름 목록을 쉼표로 구분하여 입력합니다. 미지정 시 모든 필드가 전달됩니다. * [선택] 형식: 데이터의 형식을 지정합니다. TXT, CSV, JSON 형식을 지원하며 미지정 시 TXT로 지정됩니다. * [선택] 태그 지정 여부: true로 설정한 경우 원본 로거의 이름을 태깅하여 전달합니다. 미지정 시 false로 지정됩니다.

예시) 이름이 SOURCE_LOGGER인 디렉토리 와치 (dirwatch) 로거로부터 받은 로그를 시스로그 서버 SYSLOG_IP로 전달하는 시스로그 전달자 로거 local₩syslog_send 생성

```
araqne@darkluster logpresso> logapi.createLogger syslog-send local syslog_send
 Source logger name (required)? SOURCE_LOGGER
 syslog remote ip (required)? SYSLOG_IP
 syslog server port (optional)?
 encoding (optional)?
 fields (optional)?
 format (optional)?
 use source logger tag (optional)?
 transformer (optional, enter to skip)?
 logger created: name=local\syslog-send, factory=local\syslog-send, status=stopped (passive), log count=0,
```

8장. 쿼리 관리

8.1. 스트림 쿼리

개요

스트림 쿼리는 데이터 스트림을 입력으로 하는 쿼리를 설정하여, 실시간으로 이벤트 처리나 통계 분석을 수행할 수 있습니다. 일정 기간마다 실행되는 스케줄 쿼리와 달리, 스트림 쿼리는 연속적으로 입력 순서를 보장하면서 쿼리를 수행하는 특징을 가지고 있습니다.

예를 들어 10분마다 스트림에 대한 통계를 생성하려는 경우, 스케줄 쿼리로는 미묘한 시간 차이로 인해 10분 마다 생성된 결과의 합이 전체와 일치하지 않을 수 있습니다. 반면, 스트림 쿼리는 연속적인 데이터 입력에 대해 쿼리가 실행되므로 순서와 정합성을 보장합니다.

스트림 쿼리를 사용하여 특정 시간 단위의 통계를 산출하여 중간 통계 테이블에 저장하고, 이 테이블을 쿼리하여 최종적인 통계 결과를 쿼리하도록 설계하면, 디스크를 거의 사용하지 않으면서 대용량 데이터 스트림에 대하여 실시간으로 통계 결과를 계산할 수 있습니다. 특히, 그루비 스크립팅을 이용하면 고도로 복잡한 실시간 분석 및 가공이 가능합니다.

스트림 쿼리는 입력으로 3가지의 스트림 유형을 지원합니다: * 로거: 로그 수집 설정을 통해 생성한 로거를 입력으로 사용합니다. 수집되는 모든 로그가 스트림 쿼리에 입력됩니다. * 테이블: 테이블에 새로운 행(row)이 쓰여질 때마다 스트림 쿼리에 입력됩니다. 관계형 데이터베이스(RDBMS)에서 사용하던 트리거의 진화된 사용 예로 생각할 수 있습니다. * 스트림 쿼리: 다른 스트림 쿼리의 출력을 입력으로 사용할 수 있습니다. 비정형 로그에 대하여 파싱을 수행하는 스트림 쿼리를 앞단에 두고, 해당 스트림 쿼리를 입력으로 사용하는 다수의 분석용 스트림 쿼리를 배치하는 시나리오를 예로 들 수 있습니다.

8.1.1. 스트림 쿼리 생성

스트림 쿼리는 생성되는 즉시 실행됩니다.

```
logpresso.createStreamQuery
```

- 이름: 다른 스트림 쿼리와 구별되는 유일한 스트림 쿼리 이름을 지정합니다.
- 입력유형: logger, table, stream 중 하나를 지정합니다.

아래는 local₩netscreen 로거를 입력으로 하여 10분마다 방화벽 출발지, 목적지별 통계를 netscreen_ip_stats 테이블에 입력하는 예입니다.

```
araqne> logpresso.createStreamQuery netscreen_ip_stats logger
 interval? 600
 query? parsekv | stats count by src, dst | import netscreen_ip_stats
 loggers? local\netscreen
 owner? root
 created
```

위의 예시에서 interval은 초 단위인데, 0으로 설정하는 경우에는 쿼리의 종료가 없으므로 스트리밍 처리가 가능한 쿼리 커맨드만 사용할 수 있습니다. 가령 eval, rex, search 같은 부류의 커맨드들은 스트리밍이 가능하지만, sort나 stats와 같이 입력 끝을 필요로 하는 커맨드들은 스트리밍 처리가 불가능합니다.

query는 데이터소스 커맨드를 제외한 쿼리를 입력합니다. loggers (입력유형에 따라 tables 혹은 streams)는 쉼표로 구분된 식별자 목록을 입력합니다. owner는 쿼리를 실행하는 DB 계정을 의미합니다.

8.1.2. 스트림 쿼리 삭제

```
logpresso.removeStreamQuery
```

지정된 이름의 스트림 쿼리를 삭제합니다.

8.1.3. 스트림 쿼리 목록 조회

```
logpresso.streamQueries
```

현재 설정된 모든 스트림 쿼리의 실행 상태를 조회합니다.

실행 예:

```
Stream Queries
 ----------------
|     name    | input count |     last refresh    | running | enabled |
|-------------|-------------|---------------------|---------|---------|
| pcap-stream |           0 | 2014-02-27 16:38:23 |    true |    true |
```

8.1.4. 스트림 쿼리 목록 상세 조회

```
logpresso.streamQueries -v
```

앞서 실행한 명령어에 “-v”를 추가해, 현재 설정된 모든 스트림 쿼리의 실행 상태를 상세 조회합니다.

실행 예:

```
Stream Queries
------
name=pcap-stream, interval=60, query_string=eval min = string(_time, "yyyyMMddHHmm") | stats count by min, s
```

8.2. 예약된 쿼리 경보

logpresso-core 0.6.5 버전부터 지원

예약된 쿼리를 통해 주기적으로 쿼리를 실행하고, 그 결과에 대해 조건을 걸어 경보 메일을 발송할 수 있습니다.

8.2.1. 예약된 쿼리 목록 조회

현재 예약된 쿼리 목록을 조회합니다. 예약된 쿼리는 경보 설정을 선택적으로 포함할 수 있습니다.

```
araqne@bombom demo> logpresso.scheduledQueries
 Scheduled Queries
------
[5df7705c-6254-4557-8474-ffce1846e55d] Too many tables * * * * *
```

8.2.2. 예약된 쿼리 상세 조회

현재 예약된 쿼리의 모든 설정을 상세하게 조회합니다.

```
araqne@bombom demo> logpresso.scheduledQuery 5df7705c-6254-4557-8474-ffce1846e55d
 Scheduled Query Details
---------
guid: 5df7705c-6254-4557-8474-ffce1846e55d
 title: Too many tables
 schedule type: cron
 cron scheudle: * * * * *
 owner: root
 query: logdb tables
 use alert: true
 alert query: stats count | search count > 17
 suppress interval: 120secs
 mail profile: googleapps
 mail from: xeraph@eediom.com
 mail to: xeraph@eediom.com
 mail subject: Logpresso Alert
 created at: Sun Sep 08 21:59:49 KST 2013
```

8.2.3. 예약된 쿼리 경보 생성

아래와 같이 예약된 쿼리 경보를 생성합니다.

```
araqne@bombom demo> logpresso.createScheduledQuery
 title? Too many tables
 cron schedule? * * * * *
 owner? root
 query? logdb tables
 use alert? true
 alert query? stats count | search count > 17
 suppress interval? 600
 mail profile? googleapps
 mail from? xeraph@eediom.com
 mail to? xeraph@eediom.com
 mail subject? Logpresso Alert
 created
```

각 설정 항목의 의미는 아래와 같습니다: * title: 예약된 쿼리의 제목, 메일 전송 시 메일 제목에도 표시됩니다. * cron schedule: CRON 문법으로 실행 주기를 설정합니다. * owner: 예약된 쿼리를 실행하는 주체가 될 DB 계정을 설정합니다. * query: 쿼리 문자열을 입력합니다. * use alert: 경보를 설정하려면 true를 입력합니다. 단순히 예약된 쿼리를 실행하고 쿼리 결과를 저장해두려면 false를 입력합니다. * alert query: 경보 조건 쿼리를 설정합니다. 쿼리 결과에 대하여 경보 조건 쿼리를 적용했을 때 결과가 존재한다면 경보가 발생합니다. * suppress interval: 동일한 경보가 반복적으로 발생하지 않도록 억제하려면 경보를 무시할 주기를 설정해야 합니다. 초 단위로 설정합니다. * mail profile: SMTP 설정 프로파일 이름을 입력합니다. * mail from: 보낸 사람 메일 주소를 입력합니다. * mail to: 받을 사람 메일 주소를 입력합니다. * mail subject: 메일 제목을 입력합니다.

8.2.4. 예약된 쿼리 삭제

인자로 예약된 쿼리의 식별자를 전달하여 삭제합니다.

```
araqne@bombom demo> logpresso.removeScheduledQuery d5e06505-791d-45bc-a65a-b82c66aab244
 removed
```

8.2.5. 예약된 쿼리 즉시 실행

경보 설정 후 정상적으로 동작하는지 확인하고 싶을 때 아래와 같이 즉시 실행할 수 있습니다.

```
araqne@bombom demo> logpresso.runScheduledQuery 5df7705c-6254-4557-8474-ffce1846e55d
 completed
```

8.2.6. 경보 메일 예시

경보 메일은 본문에 경보 설정과 최대 1000건의 쿼리 결과를 포함합니다.

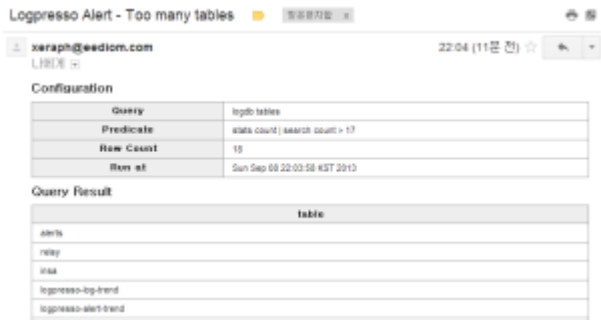

그림 3: 경보 메일 스크린샷 이미지

9장. 경보설정

9.1. SMTP 설정

경보 메일을 설정할 때 MTA 설정을 담고 있는 SMTP 프로파일을 참조하므로, 미리 SMTP 프로파일을 설정해두어야 합니다.

9.1.1. SMTP 프로파일 목록 조회

현재 SMTP 프로파일 목록을 조회합니다.

```
araqne@bombom demo> mailer.list
 Configurations
------
 smtp, smtp.gmail.com:587 user: xeraph@eediom.com
```

9.1.2. SMTP 프로파일 생성

새로운 MTA 설정을 추가합니다.

```
araqne@bombom demo> mailer.register
 Name? googleapps
 SMTP Server? smtp.gmail.com
 SMTP Port? 587
 SMTP User? xeraph@eediom.com
 SMTP Password?
 new configuration added
```

9.1.3. SMTP 프로파일 삭제

기존의 프로파일을 삭제합니다.

```
araqne@bombom demo> mailer.unregister googleapps
 smtp configuration removed
```

9.2. 예약된 쿼리 경보

logpresso-core 0.6.5 버전부터 지원

예약된 쿼리를 통해 주기적으로 쿼리를 실행하고, 그 결과에 대해 조건을 걸어 경보 메일을 발송할 수 있습니다.

9.2.1. 예약된 쿼리 목록 조회

현재 예약된 쿼리 목록을 조회합니다. 예약된 쿼리는 경보 설정을 선택적으로 포함할 수 있습니다.

```
araqne@bombom demo> logpresso.scheduledQueries
 Scheduled Queries
------
[5df7705c-6254-4557-8474-ffce1846e55d] Too many tables * * * * *
```

9.2.2. 예약된 쿼리 상세 조회

현재 예약된 쿼리의 모든 설정을 상세하게 조회합니다.

```
araqne@bombom demo> logpresso.scheduledQuery 5df7705c-6254-4557-8474-ffce1846e55d
 Scheduled Query Details
---------
```

```
guid: 5df7705c-6254-4557-8474-ffce1846e55d
 title: Too many tables
 schedule type: cron
 cron scheudle: * * * * *
 owner: root
 query: logdb tables
 use alert: true
 alert query: stats count | search count > 17
 suppress interval: 120secs
 mail profile: googleapps
 mail from: xeraph@eediom.com
 mail to: xeraph@eediom.com
 mail subject: Logpresso Alert
 created at: Sun Sep 08 21:59:49 KST 2013
```

9.2.3. 예약된 쿼리 경보 생성

아래와 같이 예약된 쿼리 경보를 생성합니다.

```
araqne@bombom demo> logpresso.createScheduledQuery
 title? Too many tables
 cron schedule? * * * * *
 owner? root
 query? logdb tables
 use alert? true
 alert query? stats count | search count > 17
 suppress interval? 600
 mail profile? googleapps
 mail from? xeraph@eediom.com
 mail to? xeraph@eediom.com
 mail subject? Logpresso Alert
 created
```

각 설정 항목의 의미는 아래와 같습니다: * title: 예약된 쿼리의 제목, 메일 전송 시 메일 제목에도 표시됩니다. * cron schedule: CRON 문법으로 실행 주기를 설정합니다. * owner: 예약된 쿼리를 실행하는 주체가 될 DB 계정을 설정합니다. * query: 쿼리 문자열을 입력합니다. * use alert: 경보를 설정하려면 true를 입력합니다. 단순히 예약된 쿼리를 실행하고

쿼리 결과를 저장해두려면 false를 입력합니다. * alert query: 경보 조건 쿼리를 설정합니다. 쿼리 결과에 대하여 경보 조건 쿼리를 적용했을 때 결과가 존재한다면 경보가 발생합니다. * suppress interval: 동일한 경보가 반복적으로 발생하지 않도록 억제하려면 경보를 무시할 주기를 설정해야 합니다. 초 단위로 설정합니다. * mail profile: SMTP 설정 프로파일 이름을 입력합니다. * mail from: 보낸 사람 메일 주소를 입력합니다. * mail to: 받을 사람 메일 주소를 입력합니다. * mail subject: 메일 제목을 입력합니다.

9.2.4. 예약된 쿼리 삭제

인자로 예약된 쿼리의 식별자를 전달하여 삭제합니다.

```
araqne@bombom demo> logpresso.removeScheduledQuery d5e06505-791d-45bc-a65a-b82c66aab244
 removed
```

9.2.5. 예약된 쿼리 즉시 실행

경보 설정 후 정상적으로 동작하는지 확인하고 싶을 때 아래와 같이 즉시 실행할 수 있습니다.

```
araqne@bombom demo> logpresso.runScheduledQuery 5df7705c-6254-4557-8474-ffce1846e55d
 completed
```

9.2.6. 경보 메일 예시

경보 메일은 본문에 경보 설정과 최대 1000건의 쿼리 결과를 포함합니다.

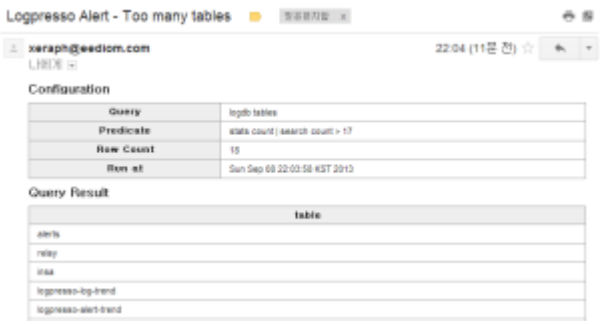

그림 4: 경보 메일 스크린샷

10장. 엑셀 및 외부 시스템 연동

10.1. API 키 관리

(logpresso-core 0.4.8과 araqne-logdb 1.2.5 버전부터 지원)

모든 기능이 갖춰진 전용 클라이언트 라이브러리를 사용하지 않더라도 HTTP를 이용하여 쉽게 연동할 수 있습니다. 간단한 스크립트, 혹은 기존의 HTTP 클라이언트 프로그램을 통하여 자동화된 쿼리를 수행하도록 구축할 수 있습니다. 이를 위해서는 먼저 API 키를 발급해야 합니다.

API 키는 임의의 랜덤한 문자열을 이용하여 권한을 부여하므로 사용자의 계정과 암호를 외부에 노출하지 않습니다. 그러나 API 키 자체가 암호와 동일한 인증 수단의 역할을 하기 때문에 주의하여 관리하여야 하고, API 키와 연관되는 계정은 별도로 구성하여 최소한의 데이터 접근 권한만을 가지도록 해야 안전합니다.

10.1.1. API 키 생성

```
araqne> logpresso.createApiKey
 Description

create api key

Arguments

 1. guid key: guid key (required)
 2. login name: db login name (required)
```

- guid (필수): 임의의 API 키 문자열
- login name (필수): API 키를 통해 인증할 계정 이름

랜덤한 guid를 생성할 때는 아라크네 쉘에서 guid 명령을 사용하면 편리합니다:

```
araqne> guid
 3530b1df-8e62-4744-9677-dfee47c5fe07
```

API 키가 등록되면 HTTP 클라이언트를 사용하여 쿼리할 수 있습니다.

10.1.2. API 키 목록 조회

```
araqne> logpresso.apiKeys
 API Keys
----
guid=sample-apikey, login_name=araqne, created=2013-06-28
```

현재 등록된 모든 API 키 목록을 조회합니다.

10.1.3. API 키 삭제

```
araqne> logpresso.removeApiKey
 Description

remove api key

Arguments

1. guid key: guid key (required)
```

지정된 API 키를 삭제합니다. 더 이상 해당 API 키를 사용하여 쿼리할 수 없게 됩니다.

10.1.4. HTTP 기반 쿼리

API 키가 등록되면 /logpresso/httpexport/query 경로를 통하여 쿼리를 수행할 수 있습니다. HTTP 쿼리스트링에 다음과 같은 매개변수를 포함해야 합니다: * apikey: 등록된 API 키 문자열 * q: 로그프레소 쿼리 문자열, URLencode 필요 (가령, 공백문자는 +로 쓰여야 합니다.) * limit: 최대 결과 행 갯수 * fields: 출력필드 순서 지정, 필드 이름들을 쉼표로 구분하여 순서대로 입력합니다. (logpresso-core 0.7.5 버전부터 지원)

또한, 경로 확장자에 따라 3종의 출력 포맷을 지원합니다. 확장자가 없는 경우 html 포맷으로 지정됩니다: * /logpresso/httpexport/query.csv: CSV 포맷으로 출력합니다. * /logpresso/httpexport/query.xml: 엑셀 XML 포맷으로 출력합니다. * /logpresso/httpexport/query.html: HTML 포맷으로 출력합니다.

HTML 및 CSV 출력 시 첫 행은 컬럼 헤더에 해당합니다.

10.1.5. 테스트 시나리오

아래와 같은 단계를 거쳐서 HTTP 쿼리 동작을 시험할 수 있습니다.

- 1.araqne> logpresso.createApiKey sample-apikey araqne ~ sample-apikey API 키를 araqne 계정으로 인증되도록 설정합니다. ~
- 2.웹 브라우저를 사용하여 아래의 주소로 접속 테스트

http://hostname:port/logpresso/httpexport/query?apikey=sample-apikey&q=logdb+tables

11장. 하둡 커넥터

11.1. 하둡 연동 설정

hadoop-core-osgi 1.1.2 버전, logpresso-hdfs 0.1.0 버전 이상 필요

로그프레소 하둡 커넥터가 설치된 경우, 하둡 클러스터를 연동하여 사용할 수 있습니다. 실시간으로 HDFS에서 원본 데이터를 수집하거나, 반대로 로그프레소를 통해 수집된 로그를 실시간으로 HDFS에 적재할 수 있습니다. 또한 쿼리를 사용하여 HDFS에 있는 파일을 즉시 조회 및 분석할 수 있으며, 로그프레소 쿼리 결과를 HDFS에 파일로 출력할 수 있습니다.

아래와 같이 로그프레소에서 하둡 연동 설정을 관리할 수 있습니다.

11.1.1. HDFS 사이트 목록 조회

`logpresso.hdfsSites` 명령어를 사용하여 기존에 설정된 HDFS 사이트 목록을 조회합니다.

```
araqne@bombom demo> logpresso.hdfsSites
 HDFS Sites
----
name=vm, filesystem=hdfs://192.168.87.2:9000
```

11.1.2. HDFS 사이트 추가

`logpresso.createHdfsSite [ ] [ ]` 명령을 이용하여 HDFS 사이트를 추가합니다. 중복된 이름이 존재하는 경우 명령이 실패합니다.

```
araqne@bombom demo> logpresso.createHdfsSite vm hdfs://192.168.87.2:9000
 created
```

11.1.3. HDFS 사이트 삭제

`logpresso.removeHdfsSite [ ]` 명령을 이용하여 기존의 HDFS 사이트 설정을 삭제합니다. 존재하지 않는 이름의 경우 명령이 실패합니다.

```
araqne@bombom demo> logpresso.removeHdfsSite vm
 removed
```

11.2. 실시간 HDFS 로그 수집 설정

HDFS 사이트 설정이 완료된 후, 로그 수집 설정을 통해 HDFS에 적재되는 파일을 실시간으로 수집할 수 있습니다. 다른 프로세스에서 HDFS 파일을 아직 닫지 않았거나 마지막 HDFS 블럭이 디스크로 플러시되지 않은 상태이더라도, 해당 데이터를 읽어들일 수 있습니다. (이 때 네임노드에서 조회되는 파일 크기는 실제 읽히는 크기보다 작습니다.)

11.2.1. HDFS 텍스트 파일 수집 설정

HDFS 텍스트 파일 수집 (hdfs-text) 로거는 HDFS에서 읽어온다는 점을 제외하면 디렉터리 로그 파일 수집 설정과 동일합니다. * [필수] HDFS 사이트: 미리 설정한 HDFS 사이트 이름을 지정합니다. * [필수] 디렉터리 경로: 로그 파일이 위치하는 HDFS 경로를 의미합니다. * [필수] 파일이름 정규표현식 패턴: HDFS 디렉터리 경로에 존재하는 파일 중 파일 이름이 정규표현식 패턴에 일치하는 경우에만 수집합니다. 정규표현식 그룹을 쓰는 경우 파일 이름에서 날짜 문자열을 추출합니다. * [선택] 날짜 추출 정규표현식 패턴: 로그에서 날짜 문자열을 추출합니다. 정규표현식 그룹으로 묶인 모든 부분을 이어붙여서 하나의 날짜 문자열을 만들어냅니다. 파일이름 정규표현식의 그룹으로 추출된 날짜문자열은 가장 앞 부분에 위치합니다. * [선택] 날짜 파싱 포맷: 날짜 문자열을 파싱하는데 사용할 날짜 포맷을 설정합니다. (예: yyyy-MM-dd HH:mm:ss) * [선택] 날짜 로케일: 날짜 문자열의 로케일. 가령 날짜 파싱 포맷의 지시자 중 MMM의 해석은 로케일에 따라 “Jan” 혹은 “1월”로 해석됩니다. 기본값은 en입니다. * [선택] 로그 시작 정규식: 로그의 시작 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다. * [선택] 로그 끝 정규식: 로그의 끝 부분을 인식하는 정규표현식을 지정합니다. 멀티라인 로그의 경우에 사용되며, 지정하지 않으면 줄 단위로 읽어들입니다. * [선택] 문자집합: 텍스트 파일 해석에 사용할 문자집합 코드를 입력합니다. 기본값은 utf-8입니다.

여러 개의 파일이 수집 대상인 경우, 파일 이름을 사전순으로 정렬하여 순서대로 읽어들입니다. 설정 예시는 디렉터리 로그 파일 수집 설정을 참고하시기 바랍니다.

11.3. 실시간 HDFS 로그 파일 출력 설정

로그프레소에서 수집하는 로그를 HDFS 파일로 실시간 출력할 수 있습니다. 지정된 파일 경로가 존재하지 않으면 자동으로 생성합니다. 또한 일정 파일 크기, 로그 갯수, 유휴 시간에 도달하면 자동으로 파일을 닫고 새로운 파일에 쓰도록 설정할 수 있습니다. 동작 주기마다 HDFS 플러시를 수행합니다.

HDFS 경로, 파일 이름 접두어, 파일 이름 접미어에 아래와 같은 지시자를 사용하면 시간 값으로 치환됩니다. 시간 값은 HDFS에 파일이 생성되는 시점의 로그프레소 서버 시각을 기준으로 합니다. 모든 월, 일, 시, 분, 초 값은 0 패딩을 포함한 문자열로 표시됩니다: * %Y: 년도 * %m: 월 * %d: 일 * %H: 시 * %M: 분 * %S: 초

만약 권한 문제로 디렉터리 생성이나 파일 생성이 실패하면, 즉시 로그 파일 출력 로거가 정지됩니다. 이는 반복적인 예외 발생과 이로 인한 성능 저하를 방지하고, HDFS 설정을 다시 점검하도록 하기 위한 것입니다.

11.3.1. HDFS 텍스트 파일 출력 설정

HDFS 텍스트 파일 출력 로거 (hdfs-totext)는 다음과 같은 설정을 입력받습니다: * [필수] 원본 로거 이름 목록: 출력할 원본 수집 설정 이름 목록을 쉼표로 구분하여 입력합니다. HDFS 텍스트 파일 출력 로거는 원본으로 지정한 수집기에서 수집되는 모든 로그를 HDFS 파일로 출력합니다. * [필수] HDFS 사이트: 미리 설정한 HDFS 사이트 이름을 지정합니다. * [필수] HDFS 경로: 출력할 HDFS 디렉터리 경로를 지정합니다. * [필수] 파일 이름 접두어: 파일 이름 타임스탬프 앞에 붙일 문자열을 지정합니다. * [필수] 파일 이름 접미어: 파일 이름 타임스탬프 뒤에 붙일 문자열을 지정합니다. 주로 .을 포함한 확장자를 설정합니다. * [선택] 롤링 기준 파일 크기: 파일을 닫고 새로 여는 기준 파일 크기를 바이트 단위로 설정합니다. 미설정 시 파일 크기 기준으로는 롤링하지 않습니다. * [선택] 롤링 기준 건수: 파일을 닫고 새로 여는 기준 로그 건수를 설정합니다. 미설정 시 로그건수 기준으로는 롤링하지 않습니다. * [선택] 유휴 기준 시간 (초): 로그가 들어오지 않는 상태로 기준 시간이 지나면 파일을 닫고 새로 열도록 설정합니다. 미설정 시 유휴 시간 기준으로는 롤링하지 않습니다.

예시) IIS 원본 로그를 수집하여 HDFS에 텍스트 로그 파일로 적재 * 원본 로거 이름 목록: local₩source_iis * HDFS 사이트: vm * HDFS 경로: /tmp/text_iis/%Y-%m-%d * 파일 이름 접두어: iis- * 파일 이름 접미어: .log * 롤링 기준 크기: 671088640 * 유휴 시간 기준 (초): 600

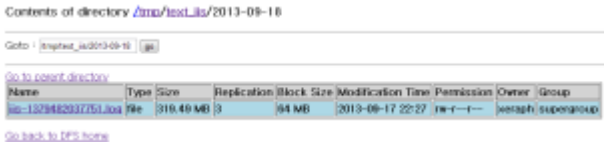

그림 5: HDFS Text Writer 이미지

11.3.2. HDFS 시퀀스 파일 출력 설정

HDFS 시퀀스 파일 포맷은 하둡의 맵리듀스 작업 시 기본으로 사용되는 바이너리 파일입니다. 개체 타입을 보존하기 때문에 쉽게 자바 개체 스트림을 쓰고 읽을 수 있으며, 레코드 단위, 블럭 단위 압축을 지원합니다.

HDFS 시퀀스 파일 출력 로거 (hdfs-toseq)는 다음과 같은 설정을 입력받습니다: * [필수] 원본 로거 이름 목록: 출력할 원본 수집 설정 이름 목록을 쉼표로 구분하여 입력합니다. HDFS 텍스트 파일 출력 로거는 원본으로 지정한 수집기에서 수집되는 모든 로그를 HDFS 파일로 출력합니다. * [필수] HDFS 사이트: 미리 설정한 HDFS 사이트 이름을 지정합니다. * [필수] HDFS 경로: 출력할 HDFS 디렉터리 경로를 지정합니다. * [필수] 파일 이름 접두어: 파일 이름 타임스탬프

앞에 붙일 문자열을 지정합니다. * [필수] 파일 이름 접미어: 파일 이름 타임스탬프 뒤에 붙일 문자열을 지정합니다. 주로 .을 포함한 확장자를 설정합니다. * [선택] 키 타입: 시퀀스 파일의 키 타입을 지정합니다. 미설정 시 LongWritable로 기록됩니다. * [선택] 키 필드 이름: 시퀀스 파일에 기록할 키 필드 이름을 지정합니다. 미설정 시 1부터 증가하는 카운터가 키 값으로 기록됩니다. * [선택] 값 타입: 시퀀스 파일의 값 타입을 지정합니다. 미설정 시 MapWritable로 기록됩니다. * [선택] 값 필드 이름: 시퀀스 파일에 기록할 값 필드 이름을 지정합니다. 미설정 시 데이터 전체가 MapWritable로 묶여 기록됩니다. * [선택] 압축 유형: record 혹은 block을 설정합니다. 미설정 시 압축하지 않습니다. * [선택] 롤링 기준 건수: 파일을 닫고 새로 여는 기준 로그 건수를 설정합니다. 미설정 시 로그 건수 기준으로는 롤링하지 않습니다. * [선택] 유휴 기준 시간 (초): 로그가 들어오지 않는 상태로 기준 시간이 지나면 파일을 닫고 새로 열도록 설정합니다. 미설정 시 유휴 시간 기준으로는 롤링하지 않습니다.