로그프레소 쿼리

# 개요

# 시작하기 전에

## 사용자 인터페이스의 표기

사용자 인터페이스(GUI, Graphic User Interface)를 구성하는 요소들은 다음과 같이 표기합니다.

|  |  |
| --- | --- |
| 표기 형식 | 의미 |
| **메뉴 1 > 메뉴 2** | 여러 단계의 메뉴를 거치는 경우, ">"를 사용해 메뉴 경로를 표시합니다. |
| **[탭]** | 화면에서 탭의 이름을 표시합니다. |
| **버튼** | 버튼 이름을 굵은 글씨로 표시합니다. |

## 명령어의 표기

다음 표는 문서에서 명령어 및 옵션, 입력값 표기에 사용하는 표기 규칙입니다.

|  |  |
| --- | --- |
| 표기 예시 | 의미 |
| table araqne\_query\_logs | 문자 그대로 입력하는 내용을 고정폭 소문자로 표기합니다. |
| VALUE, TABLE, TABLE.INDEX, FIELD | 사용자의 환경에 맞춰 입력해야 하는 내용을 고정폭 대문자로 표기합니다. |
| opt=value, [opt=value] | 명령어의 옵션을 '옵션=값' 형태로 표기합니다. 대괄호 쌍([ ])이 있으면 선택 옵션으로, 생략할 수 있습니다. |
| opt={true|false}, opt=INT{s|m|d|w|mon} | 여러 개 중 하나를 선택해서 지정해야 하는 값은 중괄호 쌍({ })으로 감싸서 표기합니다. |

이 문서에서 명령어 문법을 서술할 때 위에 있는 표기법을 사용합니다. 예를 들어 [stream](https://docs.logpresso.com/ko/query/stream-command) 명령어의 문법은 다음과 같이 표시합니다.

stream [forward=BOOL] [window=INT{y|mon|w|d|h|m|s}] STREAM[, ...]

## 약어 및 용어

이 문서에서 다음과 같은 용어를 사용합니다.

**ENT**

로그프레소 엔터프라이즈(Logpresso Enterprise)

**FRS**

로그프레소 포렌식(Logpresso Forensic)

**GUID**

16진수로 구성되는 고유 ID(Globally Unique Identity)의 약어

**SNR**

로그프레소 소나(Logpresso Sonar)

**STD**

로그프레소 스탠다드(Logpresso Standard)

**웹 콘솔**

로그프레소 제품군이 제공하는 웹 기반 사용자 인터페이스

**테이블**

로그를 저장하는 파일의 논리적 이름

### 웹 콘솔에서 쿼리 사용하기

표준 SQL은 데이터 처리의 상세한 과정을 명시하지 않고도 사용자가 원하는 데이터를 얻어낼 수 있는 선언적인 문법 특징을 가지고 있지만, 표준 SQL은 비정형적인 데이터를 처리하는데 많은 제약이 있을 뿐 아니라 스트리밍 처리를 기술하기에 자연스럽지 않은 단점이 있습니다.

로그프레소는 단순성, 응집성, 재사용성, 유연성을 극대화하는 유닉스 계열 운영체제의 설계 철학을 계승합니다. 명령어 하나는 가장 작고 단순한 기능을 수행하지만, 여러개의 명령어를 조합하는 것만으로 복잡하고 비정형적인 데이터를 효과적으로 처리할 수 있습니다.

여기서는 로그프레소 웹 콘솔에서 쿼리를 사용하는 방법과 쿼리문의 기본적인 구조를 설명합니다.

### 제품별 쿼리 메뉴 경로

로그프레소의 웹 콘솔에서 쿼리를 사용할 수 있습니다. 쿼리문을 입력할 수 있는 인터페이스가 곳곳에 있지만, 쿼리 수행을 위한 인터페이스를 별도로 제공합니다. 제품군별 쿼리 메뉴는 다음과 같습니다.

* ENT, STD: **쿼리** 또는 **쿼리 > 쿼리**
* MAE, SNR: **분석 > 쿼리**

### 쿼리문의 실행

쿼리문을 실행하려면 쿼리문을 입력 상자에 입력하고 **실행**을 누릅니다. 쿼리문은 하나의 명령어를 이용하는 단문일 수도 있고, 파이프(|)를 이용해 계속 데이터를 이어받는 여러 개의 명령문으로 구성될 수도 있습니다.

#### 쿼리 단축 키

쿼리 입력 상자에서 다음과 같은 단축 키를 지원합니다.

**쿼리 단축 키**

|  |  |
| --- | --- |
| 기능 | 단축 키 |
| 쿼리문의 실행 | **Ctrl+Enter** or **Shift+Enter** |
| 명령어 목록과 도움말 보기 | **Ctrl+Space** |
| 명령어 들여쓰기 및 자동 정렬 | **Ctrl+Shift+F** |

명령어 목록과 도움말 보기(**Ctrl+Space**)의 경우, 명령어 입력 여부에 따라 동작이 다릅니다.

* 명령어를 입력하지 않은 상태에서 단축 키를 누르면 명령어 목록을 보여줍니다.
* 명령어를 입력한 상태에서 단축 키를 누르면 사용할 수 있는 옵션 목록을 보여줍니다.

쿼리문 들여쓰기 자동 정렬은 들여쓰기와 줄 나눔을 자동으로 적용함으로써 여러 줄에 이어 길게 작성된 쿼리문을 이해하기 쉽도록 해줍니다. 이 단축키는 MAE, SNR에서만 지원합니다.

### 쿼리 유형

쿼리는 실행방식에 따라 크게 네 가지 유형으로 구분됩니다.

#### 애드혹 쿼리

애드혹 쿼리는 사용자가 임의의 시점에 임의의 질의문을 만들어서 실행하는 쿼리를 의미합니다. 로그프레소의 웹 콘솔의 로그 쿼리 메뉴, SSH를 통해 접속한 로그프레소 터미널, 혹은 로그프레소 클라이언트 SDK를 통해 프로그래밍 방식으로 사용자가 임의의 시점에 쿼리를 실행할 수 있습니다.

장시간 실행되는 쿼리를 백그라운드로 전환하면 현재 세션을 로그아웃하거나 접속이 끊어지더라도 계속 실행됩니다. 이후에 해당 쿼리를 포어그라운드로 다시 전환하여 쿼리 결과를 확인할 수 있습니다.

#### 실시간 쿼리

실시간 쿼리는 실행 시점부터 지정한 시간 범위만큼 대기하면서 실시간으로 수신되는 데이터를 대상으로 처리하는 쿼리를 의미합니다. 로그 수집기에서 로그를 수집할 때, 스트림 쿼리의 결과가 출력될 때, 테이블에 데이터가 입력될 때, 실시간으로 해당 데이터를 입력으로 수신하면서 쿼리할 수 있습니다. 이는 디스크에 전체 데이터를 저장하지 않으면서도 즉각 데이터를 샘플링하여 분석할 때 유용합니다.

실시간 쿼리 명령어로 [logger](https://docs.logpresso.com/ko/query/logger-command), [stream](https://docs.logpresso.com/ko/query/stream-command), [table](https://docs.logpresso.com/ko/query/table-command)이 있습니다.

#### 스트림 쿼리

스트림 쿼리는 시스템이 종료할 때까지 실시간 데이터 원본에 대하여 백그라운드에서 무한히 실행되는 쿼리를 의미합니다. 스트림 쿼리는 연속적으로 입력 순서를 보장하면서 쿼리를 수행하는 특징을 가지고 있습니다.

엔터프라이즈, 스탠다드는 **쿼리 > 스트림**에서 스트림 쿼리를 확인할 수 있습니다. 스트림 쿼리는 입력으로 3가지의 스트림 유형을 지원합니다:

로그 수집기

로거(logger)를 통해 수집한 모든 로그가 스트림 쿼리에 입력됩니다. 엔터프라이즈, 스탠다드는 **수집**에서, 마에스트로, 소나는 **수집 > 수집 설정**에서 로거를 구성할 수 있습니다.

테이블

테이블에 새로운 행(row)이 쓰여질 때마다 스트림 쿼리에 입력됩니다. 관계형 데이터베이스(RDBMS)에서 사용하던 트리거의 진화된 사용 예로 생각할 수 있습니다. 로그프레소 엔터프라이즈, 스탠다드는 **테이블 관리**에서, 로그프레소 소나는 **시스템 > 테이블 관리**에서 테이블을 구성할 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. 스트림 쿼리는 스트리밍 모드와 리프레시 모드로 구분됩니다.

**스트리밍 모드**

데이터 입력 완료에 의존하지 않는 명령어들 - 스트리밍 가능한 명령어 - 로만 구성된 경우 스트리밍 모드로 스트림 쿼리를 설정할 수 있습니다.

**리프레시 모드**

가령 통계나 정렬의 경우 입력이 완료되어야만 전체 데이터를 대상으로 작업을 수행할 수 있기 때문에, 일정한 주기로 입력 완료 신호를 전달하게 됩니다.

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.

#### 예약된 쿼리

예약된 쿼리는 사용자가 지정한 일정에 따라 실행됩니다. 선택적으로 쿼리 결과를 저장할 수 있으며, 경보 조건과 일치하는 결과를 메일로 전송할 수 있습니다.

로그프레소 엔터프라이즈와 스탠다드는 **쿼리 > 불러오기**에서 **저장된 쿼리 결과** 목록을 통해 조회할 수 있습니다.

### 쿼리 문법

#### 명령문 형식

쿼리는 1개 이상의 명령문으로 구성됩니다. 명령문을 구성하는 기본 단위는 명령어와 옵션, 대상 개체(object)입니다.

대상 개체가 있는 명령문

개체는 로그 수집기나 데이터 스트림, 로그 파서, 데이터가 저장된 테이블, 테이블 인덱스일 수 있습니다. 서브쿼리문, 함수, 프로시저 등으로 구성된 표현식을 호출해 결과 값을 대상으로 실행할 수도 있습니다. 이와 같은 개체를 대상으로 실행하는 명령문의 구성 형식은 다음과 같습니다.

command-name [opt\_1=VALUE] [opt\_2=VALUE] ... OBJECT[, ...]

가장 단순한 명령문의 예로, 기본 시스템 테이블인 araqne\_query\_logs에서 데이터를 조회하는 명령문은 다음과 같습니다.

table araqne\_query\_logs

대상 개체가 없는 명령문

개체가 없는 명령문은 다른 명령문으로부터 데이터를 전달받아 처리하는 명령어에서 주로 사용합니다. 예를 들어, [decodedns](https://docs.logpresso.com/ko/query/decodedns-command) 명령어가 그렇습니다. 구성 형식은 다음과 같습니다.

FORWADING\_STATEMENT | command-name [opt\_1=VALUE] [opt\_2=VALUE] [opt\_N=VALUE] ...

이와 같은 명령어들은 개체를 전달하는 선행 명령문(FORWARDING\_STATEMENT)이 반환하는 출력을 파이프(|)를 통해 입력으로 전달받아 처리합니다.

#### 파이프를 이용한 입력 처리

로그프레소에서 명령문은 파이프를(|)를 이용해 명령문의 출력을 다른 명령문에 입력으로 전달합니다. 예를 들어, **araqne\_query\_log** 테이블에서 **login\_name** 필드가 "root"인 로그만 조회하려면 다음과 같이 쿼리합니다.

table araqne\_query\_logs | search login\_name == "root"

이 쿼리문은 root 계정으로 실행된 쿼리문들을 보여줍니다. root 문자열을 포함하는 행이 10분 단위로 몇 건씩 발생하는지 10분 단위로 통계를 계산하려면 다음과 같이 쿼리합니다:

table araqne\_query\_logs| search login\_name == "root"| timechart span=10m count

이와 같이 첫번째 명령문의 출력을 두번째 명령문의 입력으로, 두번째 명령문의 출력을 세번째 명령문의 입력으로 전달하는 과정을 거쳐, 세번째 명령문의 출력이 전체 쿼리문의 결과로 전달됩니다. 쿼리 결과는 클라이언트의 요청 형태에 따라 디스크에 임시로 기록되거나, 네트워크를 경유하여 즉시 스트리밍할 수 있습니다.

#### 서브쿼리

어떤 쿼리 명령어는 명령문 안에 중첩된 명령문을 실행하고, 실행된 결과를 받아서 실행합니다. 중첩된 명령문을 서브쿼리(subquery)라 합니다.

서브쿼리문은 입력할 때 대괄호 쌍([ ])으로 감싸서 표현합니다. 서브쿼리는 전체 쿼리문 실행 시점에 상위 쿼리문보다 앞서 실행됩니다. 서브쿼리가 반환하는 레코드를 쿼리 명령어가 받아서 실행합니다.

서브쿼리가 있을 경우, 명령문 구조는 다음과 같습니다.

command [ SUBCOMMAND\_STATEMENT ]

#### 명령문의 주석 처리

주석 처리 명령어 '#'을 이용해 명령행에 설명을 삽입하거나, 단일 명령행 또는 연속된 명령행을 주석 처리할 수 있습니다. 쿼리 입력 상자에서 주석 처리된 명령행은 회색으로 표시됩니다.

단일 명령행의 주석 처리

명령문 첫머리에 '#'을 삽입해 주석 처리할 수 있습니다.

'#' 뒤에 공백문자가 필요합니다. 공백문자가 없으면 주석 처리가 되지 않습니다.

# 최근 1시간 동안 sys\_cpu\_log에 기록된 CPU 사용률 조회하기 | table duration=1h sys\_cpu\_logs | # eval total = kernel + user

위 예시에서는 최근 1시간 동안 sys\_cpu\_log에 기록된 CPU 사용률 조회하기, eval total = kernel + user가 주석 처리되어 table duration=1h sys\_cpu\_logs만 실행합니다.

여러줄의 명령행을 주석 처리

# [과 ]으로 주석 처리할 명령행들을 감싸서 주석 처리합니다.

table duration=1h sys\_cpu\_logs | # [ eval total = kernel + user | search total > 10 ] | sort \_time

위 예시에서는 대괄호 안의 명령문이 주석 처리되어 실제 실행 쿼리는 table duration=1h sys\_cpu\_logs | sort \_time 입니다.

주석 처리 명령어 #은 뒤에 오는 문자열 뿐만 아니라, 서브쿼리를 감싸는 대괄호 쌍([ ]) 안에 있는 파이프(|)를 무시합니다. 서브쿼리에 줄바꿈이 있어도 무시합니다. 다시 말하면, 서브쿼리문 전체를 무시하고 대괄호 쌍 밖에 파이프가 나타날 때까지 주석 처리합니다.

table sys\_cpu\_logs | # union [ table sys\_cpu\_logs | limit 30 ] | eval total = kernel + user

위 예시에서 서브쿼리인 union [ table sys\_cpu\_logs | limit 30 ]는 모두 주석 처리되어 실제 실행 쿼리는 table sys\_cpu\_logs | eval total = kernel + user 입니다.

#### 쿼리 매개변수

쿼리 매개변수에 값을 할당하고, 필요할 때 호출해 사용할 수 있습니다. 상수 대신에 함수 등을 이용한 표현식을 이용함으로써 동적으로 값을 할당해 쿼리를 실행할 때 유용합니다. 예를 들어, 예약된 쿼리를 실행할 때 현재 날짜를 기준으로 일주일 범위의 데이터를 조회해서 처리하거나, 프로시저 실행 시 사용자가 입력한 매개변수 값을 이용하여 쿼리를 실행하려고 할 때 쿼리 매개변수를 사용합니다.

매개변수의 선언

매개변수는 [set](https://docs.logpresso.com/ko/query/set-command) 혹은 [setq](https://docs.logpresso.com/ko/query/setq-command) 명령을 이용해 선언합니다.

매개변수의 참조

매개변수에 할당된 값은 매개변수 참조 함수인 [$()](https://docs.logpresso.com/ko/query/$-function)으로 참조합니다.

#### 함수

쿼리문에 함수를 사용할 수 있습니다. 함수는 표현식을 사용할 수 있는 곳이면 어디든 사용 가능합니다.

#### 프로시저

로그프레소는 사전 정의된 쿼리문을 함수처럼 호출할 수 있는 프로시저 기능을 제공합니다. DBMS의 프로시저와 유사한 기능으로, 다음과 같은 장점을 제공합니다.

**재사용성 및 유지보수 향상**

프로시저를 통해 특정한 기능을 제공하는 쿼리를 모듈화함으로써 재사용성을 높일 수 있습니다. 사용자에게는 프로시저 이름과 사용할 매개변수만 알려주면 됩니다. 사용자는 반복적인 쿼리를 쓰지 않아도 되고, 쿼리문이 간결해져 유지보수하기 쉬워집니다.

**보안성 향상**

[dbquery](https://docs.logpresso.com/ko/query/dbquery-command), [ftp](https://docs.logpresso.com/ko/query/ftp-command), [sftp](https://docs.logpresso.com/ko/query/sftp-command) 등 외부의 시스템에 접속하는 명령어는 프로파일 사용 권한이 필요합니다. 프로파일 권한을 사용자에게 직접 부여하면 외부 시스템에서 임의의 작업을 수행할 수 있으므로 안전하지 않습니다. 사용자가 특정한 관리자 권한이 필요한 명령을 실행하거나 로컬/원격 호스트에 임의의 작업을 수행할 수 있는 명령을 프로시저로 구성한 다음 사용자 권한을 관리함으로써 시스템 전체의 관리자 권한을 부여하지 않고도 원하는 작업을 사용자 계정으로 사용할 수 있게 해줍니다. 원본데이터의 일부만 조회하도록 제한하거나, 원본 데이터를 마스킹해 보여주는 방식으로 응용할 수 있습니다.

**로그프레소의 설정 정보 접근**

로그프레소의 시스템 테이블은 관리자 권한이 있어야 접근할 수 있습니다. 로그프레소의 시스템 설정 정보를 사용자가 접근해야 할 때 프로시저를 통해 접근할 수 있도록 해줍니다.

프로시저 정의하기

웹 콘솔에서 프로시저를 정의하고 관리할 수 있습니다. 프로시저를 관리하는 기능은 다음 경로에서 사용할 수 있습니다.

* (ENT, STD) **쿼리 > 프로시저**
* (SNR) **분석 > 프로시저**

프로시저로 사용할 쿼리문은 프로시저를 호출할 때 사용할 매개변수 또는 사용자 정의 필드를 포함할 수 있습니다.

프로시저 화면에서 정의하는 쿼리는 [$()](https://docs.logpresso.com/ko/query/$-function) 함수를 사용하여 사용자가 프로시저를 호출할 때 넘긴 매개변수를 참조할 수 있습니다. 예를 들면 아래와 같습니다:

table duration=1d sys\_cpu\_logs | search kernel + user >= $("threshold")

예시된 쿼리문에서 threshhold가 매개변수입니다.

프로시저 작성 시 가장 흔한 실수는 [$()](https://docs.logpresso.com/ko/query/$-function) 함수 참조가 매크로처럼 치환된다고 생각하여 쿼리를 작성하는 것입니다. [$()](https://docs.logpresso.com/ko/query/$-function) 함수는 쿼리 명령어에서 표현식을 입력할 수 있는 위치에만 지정할 수 있습니다. 예를 들어, 아래의 프로시저는 [dbquery](https://docs.logpresso.com/ko/query/dbquery-command)가 임의의 SQL 문장 입력을 표현식으로 지원하지 않으므로 올바른 쿼리가 아닙니다:

dbquery USERDB $("sql")

프로시저의 호출

프로시저를 호출해 실행하는 명령어로는 [proc](https://docs.logpresso.com/ko/query/proc-command)이 있습니다. 호출 방법은 명령어의 설명을 참조하십시오.

# 엔터프라이즈 명령어

## 매개변수

### set

매개변수에 값을 할당합니다.

#### 문법

set VAR\_NAME=EXPR

**VAR\_NAME = EXPR**

표현식을 평가한 값을 매개변수에 할당합니다. 명령어의 우변에는 쿼리 시작 시점에 레코드 없이 평가될 수 있는 모든 표현식을 사용할 수 있습니다.

* 할당 연산자(=) 전후로 공백문자를 넣어도, 넣지 않아도 동작합니다.
* 쿼리 시작 시점에 레코드 없이 평가될 수 있는 모든 표현식을 사용할 수 있습니다.
* 쿼리 매개변수는 하나의 쿼리 인스턴스가 살아있는 동안 유효합니다.
* 여러 개의 set 명령어가 있을 때는 왼쪽부터 순서대로 평가됩니다.

#### 설명

쿼리 매개변수는 하나의 쿼리 인스턴스가 살아있는 동안 유효하고, set 명령어를 사용해서 쿼리를 시작하는 시점에 매개변수 값을 평가하도록 할 수 있습니다. 아래의 쿼리 문자열은 [table](https://docs.logpresso.com/ko/query/table-command) 명령어를 사용해서 3일 전 0시부터 당일 0시 이전까지의 데이터를 동적으로 조회하는 예제입니다.

set from = string(dateadd(now(), "day", -3), "yyyyMMdd")| set to = string(now(), "yyyyMMdd")| table from=$("from") to=$("to") sys\_cpu\_logs

위와 같이 set 명령어를 사용하여 쿼리 매개변수를 설정할 수 있으며, [$()](https://docs.logpresso.com/ko/query/$-function) 함수를 사용하여 쿼리 매개변수의 값을 참조할 수 있습니다.

각 쿼리 명령어의 옵션들은 쿼리 매개변수로 대치할 수 있습니다. 예를 들어, 예약된 쿼리를 실행할 때 현재 날짜를 기준으로 일주일 범위의 데이터를 조회해서 처리한다거나, 프로시저 실행 시 사용자가 입력한 매개변수 값을 이용하여 쿼리를 실행하려고 할 때 쿼리 매개변수를 사용하게 됩니다.

프로시저 호출 시, 프로시저의 매개변수로 전달되는 값들이 쿼리 매개변수로 설정됩니다. 따라서 프로시저를 생성하거나 편집할 때 쿼리 문자열에는 프로시저 매개변수에 해당되는 값들이 이미 있다고 가정하고 [$()](https://docs.logpresso.com/ko/query/$-function) 함수로 참조하여 사용할 수 있습니다.

### setq

서브쿼리를 실행하고, 첫번째 레코드에 존재하는 키-값 쌍을 쿼리 매개변수로 설정합니다.

#### 문법

setq [ SUBQUERY ]

**SUBQUERY**

서브쿼리문을 대괄호 쌍([ ])으로 감싸서 서브쿼리문을 입력합니다.

#### 설명

서브쿼리를 실행한 결과의 첫번째 레코드에 존재하는 필드-값 쌍을 매개변수로 설정합니다. 서브쿼리가 1개 이상의 결과를 반환하더라도 두번째 레코드부터는 무시합니다.

setq 명령문으로 구성한 서브쿼리는 전체 쿼리문의 모든 명령보다 먼저 실행됩니다. 여러 개의 setq 명령문이 있으면 순차적으로 실행됩니다.

#### 사용 예

ip, port 매개변수에 setq 서브쿼리 첫 번째 레코드 값인 '1.2.3.4', 8888을 각각 세팅하고, 각 매개변수를 setq\_ip, setq\_port 필드에 할당

setq [ json "[{'ip':'1.2.3.4', 'port':8888}, {'ip':'2.3.4.5', 'port':443}]" ]| eval setq\_ip=$("ip"), setq\_port=$("port")

## 데이터 조회

### csvfile

CSV(Comma-Separated Values) 또는 TSV(Tab-Separated Values) 파일에서 데이터를 조회합니다. CSV 또는 TSV 파일 첫번째 줄에 있는 헤더 정보를 읽어와 필드명으로 사용합니다.

#### 문법

csvfile [OPTIONS] PATH

필수 매개변수

**PATH**

CSV 파일의 경로, 또는 파일 경로를 반환하는 쿼리 매개변수를 참조합니다. 파일 이름에 와일드카드(\*)를 사용해 패턴 매칭 방식으로 파일을 조회할 수 있습니다. 예를 들어, PATH에 allow-\*.csv를 지정함으로써 allow-ip.csv, allow-user.csv, allow-url.csv 등의 파일을 한꺼번에 조회할 수 있습니다. 파일을 읽어오려면 로그프레소 실행 계정에 접근 권한이 부여되어 있어야 합니다.

선택 매개변수

**cs=CHARSET**

문자열 인코딩 형식(기본값: utf-8). 이 옵션은 대소문자를 구분하지 않으며, 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 CHARSET으로 사용할 수 있습니다: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**limit=INT**

가져올 레코드의 최대 개수(기본값: 제한 없음)

**maxcol=INT**

조회할 최대 컬럼 수(기본값: 10,000 개). 조회할 데이터의 컬럼 개수가 지정한 최대 컬럼을 초과하는 경우, rest 옵션을 이용해 처리 방식을 정의합니다.

**offset=INT**

건너뛸 레코드 개수(기본값: 0)

**rest=BOOL**

maxcol 옵션으로 지정한 개수 이후의 컬럼의 표시 여부(기본값: f).

* t: maxcol 옵션으로 지정한 개수를 초과하는 컬럼의 데이터를 모두 **\_rest** 필드에 할당
* f: 지정한 개수를 초과하는 컬럼의 데이터를 모두 버림

**strict=BOOL**

RFC4180[(https://tools.ietf.org/html/rfc4180)](https://tools.ietf.org/html/rfc4180)의 준수 옵션 (default: f)

* t: 마이크로소프트 엑셀과 동일하게 RFC4180 준수. 이 옵션을 tab=t와 함께 사용할 수 없습니다.
* f: CSV 파일을 유연하게 파싱

**tab=BOOL**

탭(tab) 문자를 구분자로 사용 여부 (기본값: f).

* t: 탭(tab) 문자를 구분자로 사용
* f: 쉼표(,)를 구분자로 사용

#### 사용 예

/opt/logpresso/wp-nginx.csv 파일을 조회

# 다운로드: https://raw.githubusercontent.com/logpresso/dataset/main/wp-nginx.csv | csvfile /opt/logpresso/wp-nginx.csv

/opt/logpresso/wp-nginx.csv 파일 내용 중 첫 줄을 건너뛴 후 20건의 레코드 조회

csvfile limit=20 offset=1 /opt/logpresso/wp-nginx.csv

/opt/logpresso/wp-nginx.csv 파일에서 4개의 컬럼만 조회

csvfile maxcol=4 /opt/logpresso/wp-nginx.csv

/opt/logpresso/wp-nginx.csv 파일에서 4개의 컬럼만 조회하고, 나머지 컬럼 값을 **\_rest** 필드에 할당

csvfile maxcol=4 rest=t /opt/logpresso/wp-nginx.csv

구분자와 컬럼 사이에 공백이 있는 데이터

strict=t일 때, 구분자와 컬럼 사이에 공백이 있으면 큰 따옴표(")는 문자로 인식되어 의도한대로 파싱되지 않습니다.

# 다운로드: https://raw.githubusercontent.com/logpresso/dataset/main/csvfile-strict-option-test-1.csv | csvfile strict=t /opt/logpresso/csvfile-strict-option-test-1.csv

strict=f일 때는 큰 따옴표 쌍(" ")의 짝이 맞으면 따옴표 쌍 안에 있는 문자열만 컬럼으로 인식하기 때문에 의도한대로 파싱됩니다.

csvfile strict=f /opt/logpresso/csvfile-strict-option-test-1.csv

구분자와 컬럼 사이에 공백이 없는 데이터

strict 값에 관계없이 구분자와 컬럼 사이에 공백이 없으므로 의도한대로 파싱됩니다.

# 다운로드: https://raw.githubusercontent.com/logpresso/dataset/main/csvfile-strict-option-test-2.csv | csvfile strict=t /opt/logpresso/csvfile-strict-option-test-2.csv csvfile strict=f /opt/logpresso/csvfile-strict-option-test-2.csv

큰 따옴표(")를 이스케이프 처리(")한 데이터

strict=t일 때, 이스케이프 문자(\)를 일반 문자로 인식하므로 큰 따옴표 쌍(" ")으로 감싼 컬럼 안에서 "를 사용하여 큰 따옴표(")를 표기한 경우, 의도한대로 파싱되지 않습니다.

# 다운로드: https://raw.githubusercontent.com/logpresso/dataset/main/csvfile-strict-option-test-3.csv | csvfile strict=t /opt/logpresso/csvfile-strict-option-test-3.csv

strict=f일 때, 겹 큰 따옴표("")와 이스케이프 처리한 큰 따옴표(")가 의도한대로 컬럼 안에서 큰 따옴표로 파싱됩니다.

csvfile strict=f /opt/logpresso/csvfile-strict-option-test-1.csv csvfile strict=f /opt/logpresso/csvfile-strict-option-test-3.csv

#### 호환성

maxcol과 rest 옵션은 ENT #2246 2019-05-24\_14-58 버전부터 지원합니다.

### fulltext

지정된 인덱스에 대해 풀텍스트 검색을 수행합니다.

#### 문법

fulltext [duration=INT{mon|w|d|h|m|s}] [from=yyyyMMddHHmmss] [to=yyyyMMddHHmmss] [limit=INT] [offset=INT] [order={desc|asc}] [tt=BOOL] EXPR [from TABLE[.INDEX], ...]

필수 매개변수

**EXPR [from TABLE[.INDEX], ...]**

테이블(TABLE) 또는 테이블의 특정 인덱스(INDEX)에서 검색할 문자열 또는 표현식. TABLE을 지정하지 않으면 모든 테이블을 검색합니다. TABLE만 지정하면 해당 테이블의 모든 인덱스를 검색합니다. 만약 같은 TABLE이나 INDEX를 여러 번 지정하면, 중복해서 지정한 횟수만큼 출력됩니다.

EXPR은 검색할 데이터를 표현하는 식으로, 다음과 같은 규칙을 만족해야 합니다.

* 비교연산자를 사용할 수 있습니다. 사용할 수 있는 비교연산자는 다음과 같습니다: ==, !=, >=, >, <, <=
* 검색할 문자열은 큰 따옴표 쌍(" ")으로 감싸야 하며, 대소문자를 구분하지 않습니다.
* 논리 연산자인 and, or, not과 괄호 쌍(( ))을 조합해 입력할 수 있습니다.
* 테이블을 지정하지 않으면 모든 테이블을 검색합니다.
* 중복 지정된 테이블이나 인덱스가 있으면 중복 횟수만큼 출력됩니다.

EXPR은 표현식에 대괄호 쌍([ ])으로 감싼 서브쿼리를 인식합니다. 인덱스 검색을 수행하기 전에 서브쿼리를 먼저 수행하고, 서브쿼리 결과에 등장하는 모든 용어를 검색합니다. 서브쿼리가 반환하는 검색 대상이 많을수록 인덱스 검색 속도가 느려집니다. 서브쿼리에 [fields](https://docs.logpresso.com/ko/query/fields-command) 명령어를 사용하여 꼭 필요한 필드만 검색하는 것을 권장합니다.

EXPR에서 사용할 수 있는 전용 함수로 range(), iprange()가 있습니다. 이 두 함수는 다른 명령어에서 사용할 수 없습니다.

range() 함수는 인자로 받은 인덱스에서 지정된 범위에 포함되는 숫자를 검색합니다.

range(MIN\_INT, MAX\_INT)

**MIN\_INT**

검색할 범위의 숫자 중에서 최소값. 이 값은 검색 범위에 포함됩니다.

**MAX\_INT**

검색할 범위의 숫자 중에서 최대값. 이 값은 검색 범위에 포함됩니다.

iprange() 함수는 인덱스에서 지정된 IPv4 또는 IPv6 주소 구간에 포함되는 IP 주소를 검색합니다.

iprange(START\_IP\_EXPR, END\_IP\_EXPR)

**START\_IP\_EXPR**

검색할 IP 주소 구간의 시작 주소 문자열을 반환하는 표현식. 이 값은 검색 범위에 포함됩니다.

**END\_IP\_EXPR**

검색할 IP 주소 구간의 마지막 주소 문자열을 반환하는 표현식. 이 값은 검색 범위에 포함됩니다.

선택 매개변수

**duration=INT{mon|w|d|h|m|s}**

현재 시각을 기준으로 지정한 시간 이내의 로그만 검색합니다. mon(월), w(주), d(일), h(시), m(분), s(초) 단위로 지정할 수 있습니다. 예를 들어 10s은 현재 시각을 기준으로 “최근 10초”를 의미합니다. 이 옵션은 from, to와 함께 사용할 수 없습니다.

**from=yyyyMMddHHmmss**

검색 대상 기간의 시작 날짜와 시각. yyyyMMddHHmmss 형식으로 입력하며, 입력한 시각도 검색 범위에 포함됩니다. 앞부분만 입력하면 나머지 자리는 0으로 인식합니다. 예를 들어, 20130605를 입력하면 20130605000000(2013년 6월 5일 0시 0분 0초)으로 인식합니다. 이 옵션은 duration과 함께 사용할 수 없습니다.

**to=yyyyMMddHHmmss**

검색 대상 기간의 마지막 날짜와 시각. yyyyMMddHHmmss 형식으로 입력하며, 입력한 시각은 검색 범위에 포함되지 않습니다. 입력 방식은 from과 같습니다. 이 옵션은 duration과 함께 사용할 수 없습니다.

**limit=INT**

검색할 레코드의 최대 개수(기본값: 제한 없음)

**offset=INT**

건너뛸 겸색 결과 개수(기본값: 0)

**order={desc|asc}**

인덱스의 검색 순서(기본값: desc)

* desc: 최근 데이터부터 검색
* asc: 오래된 데이터부터 검색

**tt=BOOL**

검색 토크나이저의 사용 여부(기본값: f). tt=t일 때, 검색할 문자열을 각 인덱스에 맞는 토크나이저로 분할하여 검색합니다. 또한 EXPR에서 문자열 와일드카드(\*)는 문자열 시작이나 끝에만 넣을 수 있습니다. 예를 들어 EXPR에 "\*asp", "asp\*", "\*asp\*"은 입력 가능하지만 "a\*sp"는 입력할 수 없습니다. 분할된 검색어는 and 논리연산자로 묶여 쿼리문이 재구성됩니다. 예를 들어 fulltext tt=t dst == "10.10.130.235" 쿼리문은 fulltext dst == "10" and dst == "10" and dst == "130" and dst == "235"으로 재구성됩니다.

'duration', 'from'과 'to'를 사용하지 않으면 레코드를 기록한 모든 기간에 대해 검색합니다.

#### 사용 예

전체 테이블에서 2013년 6월 5일 로그 중 1.2.3.4를 포함한 로그 검색

fulltext from=20130605 to=20130606 "1.2.3.4"

iis 테이블에서 cmdshell을 포함한 모든 웹 로그 검색

fulltext "cmdshell" from iis

iis 테이블에서 game을 포함하면서 MSIE 혹은 Firefox 문자열이 포함된 모든 웹 로그 검색

fulltext "game" and ("MSIE" or "Firefox") from iis

iis 테이블에서 400~500 범위의 숫자를 포함한 웹 로그 검색

fulltext range(400, 500) from iis

SSLVPN으로 끝나는 모든 테이블에서 192.0.0.1 ~ 192.0.0.255 사이의 IP 주소 검색

fulltext iprange("192.0.0.1", "192.0.0.255") from \*.\*SSLVPN

iis 테이블에서 블랙리스트 DB의 IP 집합을 검색

fulltext [ dbquery black select ip from ip\_blacklist ] from iis

테이블의 파서가 openssh인 테이블 집합을 대상으로 풀텍스트 검색

fulltext "term" from meta("parser==openssh")

iis 테이블의 fidx 인덱스 데이터 중 첫 5건은 건너뛴 후 20건 조회

fulltext offset=5 limit=20 "\*" from iis.fidx

"1.2.3.4" 문자열을 fidx 인덱스의 토크나이저를 사용해 분할하여 iis 테이블의 fidx 인덱스에서 검색

fulltext tt=t "1.2.3.4" from iis.fidx

### json

JSON 문자열을 이용하여 데이터 원본을 생성합니다. 일반적으로 이후에 연결되는 쿼리 커맨드의 동작을 테스트하기 위한 용도로 사용합니다.

#### 문법

json JSON\_DATA

필수 매개변수

**JSON\_DATA**

큰 따옴표 쌍(" ")으로 감싼 JSON 문자열, 또는 JSON 형식으로 문자열을 반환하는 표현식. JSON 문자열에 큰 따옴표(")가 있으면 역슬래시(\)를 사용해 이스케이프 처리(")해야 합니다.

#### 사용 예

a => 8, b => "miles" 키-값 쌍을 가진 레코드 생성

json "{ 'a': 8, 'b':'miles' }"

a => 8, b => "miles" 레코드와 a => 2, b => "cats" 레코드 생성

json "[{ 'a': 8, 'b':'miles' }, { 'a': 2, 'b':'cats' }]"

### jsonfile

JSON 파일에서 데이터를 조회합니다. JSON 파일은 개행 문자로 줄 바꿈한 형식이어야 합니다. 키는 필드의 이름으로 사용되고, 값을 해당 필드에 할당합니다.

#### 문법

jsonfile [OPTIONS] PATH

필수 매개변수

**PATH**

JSON 파일의 경로. 파일 이름에 와일드카드(\*)를 사용해 패턴 매칭 방식으로 파일을 조회할 수 있습니다. 예를 들어, PATH에 allow-\*.json를 지정함으로써 allow-ip.json, allow-user.json, allow-url.json 등의 파일을 한꺼번에 조회할 수 있습니다. 파일을 읽어오려면 로그프레소 실행 계정에 접근 권한이 부여되어 있어야 합니다.

선택 매개변수

**limit=INT**

가져올 레코드의 최대 개수. 개행문자(CRLF 또는 LF)로 JSON 엔트리를 구분합니다.

**offset=INT**

건너뛸 레코드 개수(기본값: 0)

**overlay=BOOL**

JSON 원본 데이터의 출력 옵션(기본값: f).

* t: 파싱된 데이터를 필드에 출력하고, JSON 원본 데이터를 **line** 필드에 출력
* f: 파싱된 데이터만 필드에 출력

#### 사용 예

/opt/logpresso/wp-nginx.json 파일에서 데이터를 가져와 출력

# 다운로드: https://raw.githubusercontent.com/logpresso/dataset/main/wp-nginx.json | jsonfile /opt/logpresso/wp-nginx.json

/opt/logpresso/wp-nginx.json 파일에서 첫 줄을 건너뛴 후 20건의 레코드 조회

jsonfile offset=1 limit=20 /opt/logpresso/wp-nginx.json

/opt/logpresso/wp-nginx.json 파일에서 데이터를 가져와 출력하고, JSON 원본을 **line** 필드에 출력

jsonfile overlay=t /opt/logpresso/wp-nginx.json

### load

저장된 쿼리의 결과를 조회합니다.

#### 문법

load GUID

필수 매개변수

**GUID**

저장된 쿼리 결과에 할당된 GUID

#### 설명

사용자가 저장된 쿼리의 GUID 정보를 조회하는 방법이 없으므로 load 명령어를 사용자가 직접 실행하는 경우는 없습니다. 대신, (ENT, STD 웹 콘솔) **쿼리 > 불러오기 > 저장된 쿼리 결과 목록**에서 저장된 항목 이름을 누르면 **쿼리 > 쿼리**에서 load 명령어로 해당 저장된 쿼리 결과를 불러옵니다.

### logger

지정된 시간 동안 로그 수집기에서 수집한 로그를 실시간으로 출력합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

logger window=INT{y|mon|w|d|h|m|s} NAMESPACE\LOGGER[, ...]

**window=INT{y|mon|w|d|h|m|s}**

실시간으로 입력 데이터를 출력할 기간. y(연), mon(월), w(주), d(일), h(시), m(분), s(초) 단위로 지정할 수 있습니다. 예를 들어 10s은 "쿼리 실행 시점부터 10초"입니다.

**NAMESPACE\LOGGER, ...**

실시간으로 입력 데이터를 조회할 로그 수집기. 두 개 이상 로그 수집기를 지정하려면 구분자로 쉼표(,)를 사용합니다. LOOGER에 와일드카드(\*)를 사용해 패턴과 이름이 일치하는 수집기를 한 번에 조회할 수 있습니다.

NAMESPACE는 수집기가 속한 이름 공간입니다. 수집기의 이름은 동일 NAMESPACE 안에서 유일합니다. NAMESPACE에 local로 표시되는 경우, logger 명령어를 실행한 로그프레소 서버의 네임스페이스를 나타냅니다. 그 외에는 로그프레소 서버나 센트리의 식별자가 표시됩니다. LOGGER는 수집기의 이름을 나타냅니다.

#### 사용 예

local\sample1, local\sample2 로그 수집기에서 10초간 실시간 수신

logger window=10s local\sample1, local\sample2

### pcapfile

PCAP 파일에서 데이터를 조회합니다.

#### 문법

pcapfile FILE\_PATH

필수 매개변수

**FILE\_PATH**

pcap 파일의 경로. 파일 이름에 와일드카드(\*)를 사용하면 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. 파일을 읽어오려면 로그프레소 실행 계정에 접근권한이 부여되어 있어야 합니다.

#### 설명

페이로드 바이너리를 **payload** 필드에 출력합니다. 이 명령어는 PCAP 파일을 읽어들인 후 데이터를 가공하는 명령인 [decodedhcp](https://docs.logpresso.com/ko/query/decodedhcp-command), [decodedns](https://docs.logpresso.com/ko/query/decodedns-command), [decodehttp](https://docs.logpresso.com/ko/query/decodehttp-command), [decodesflow](https://docs.logpresso.com/ko/query/decodesflow-command), [pcapdecode](https://docs.logpresso.com/ko/query/pcapdecode-command) 등으로 전달하는 용도로 사용합니다.

#### 사용 예

예제로 사용된 nslookup.pcap 파일은 다음 경로에서 받을 수 있습니다.https://github.com/logpresso/dataset/blob/main/pcap/nslookup.pcap

nslookup.pcap 파일 조회

pcapfile nslookup.pcap

nslookup.pcap 파일 조회 결과에서 DNS 트래픽 확인

pcapfile nslookup.pcap | decodedns

### remote

원격 페더레이션 노드에서 쿼리를 실행합니다. 노드 접속 실패 시 쿼리가 실패합니다.

#### 문법

remote NODE [ SUBQUERY ]

필수 매개변수

**NODE**

쿼리를 실행할 대상 노드의 이름. [system nodes](https://docs.logpresso.com/ko/query/system-nodes-command) 쿼리 결과에서 노드 이름을 확인 후 입력합니다.

**[ SUBQUERY ]**

대괄호 쌍([ ]) 안에 서브쿼리문을 입력합니다.

#### 사용 예

n1 노드에서 system tables 쿼리 실행

remote n1 [ system tables ]

### result

현재 세션에서 실행한 쿼리의 결과를 조회합니다.

#### 문법

result [offset=INT] QUERY\_ID

필수 매개변수

**QUERY\_ID**

결과를 조회할 쿼리의 ID(기본값: 없음). 쿼리 ID는 [system queries](https://docs.logpresso.com/ko/query/system-queries-command) 명령어를 통해 확인할 수 있습니다.

선택 매개변수

**offset=INT**

건너뛸 레코드 개수(기본값: 0)

#### 사용 예

현재 세션에서 실행중인 쿼리(616)의 쿼리 결과를 조회

result 616

현재 세션에서 실행중인 쿼리(616)의 쿼리 결과 중 10개를 건너 뛰고 조회

result offset=10 616

### stream

지정된 스트림 수신 데이터를 출력하거나(window 옵션), 지정된 스트림에 입력 데이터를 전달합니다(forward 옵션). 이 커맨드를 실행하려면 관리자 권한이 필요합니다.

#### 문법

stream [forward=BOOL] [window=INT{y|mon|w|d|h|m|s}] STREAM, ...

필수 매개변수

**STREAM, ...**

쉼표(,)를 구분자로 사용하는 스트림 목록. 스트림 이름에 와일드카드(\*)를 사용해 패턴과 일치하는 모든 스트림을 한 번에 지정할 수 있습니다.

선택 매개변수

**forward=BOOL**

입력으로 전달받은 쿼리 데이터를 STREAM, ...으로 지정된 스트림으로 전송 여부를 제어하는 옵션(기본값: f).

* t: 파이프(|)를 통해 입력으로 전달받은 데이터를 STREAM, ...으로 지정한 스트림으로 전달. 명시적으로 스트림 전달 기능을 활성화할 때만 사용합니다. 이 옵션은 window와 함께 사용할 수 없습니다.
* f: STREAM, ...으로 지정한 스트림으로부터 데이터를 수신

**window=INT{y|mon|w|d|h|m|s}**

쿼리 실행 시점부터 출력을 실행할 시간을 지정합니다. y(연), mon(월), w(주), d(일), h(시), m(분), s(초) 단위로 지정할 수 있습니다. 단위가 y일 때, 1y만 허용됩니다. 예를 들어 10s은 쿼리 실행 시각을 기준으로 “앞으로 10초”입니다. 이 옵션은 forward와 함께 사용할 수 없습니다.

#### 사용 예

sample1, sample2 스트림으로부터 10초간 데이터를 실시간으로 수신

stream window=10s sample1, sample2

test 테이블에서 100 개의 레코드를 sample1, sample2 스트림의 입력으로 전달

table limit=100 test | stream forward=t sample1, sample2

### table

로그프레소 테이블에 저장된 데이터를 조회합니다.

#### 문법

table [OPTIONS] TABLE[, ... ]table [OPTIONS] meta("KEY\_VALUE\_EXPR"[, ... ])

필수 매개변수

**TABLE, ...**

검색할 테이블 목록. 여러 테이블을 지정하려면 구분자로 쉼표(,)를 사용합니다.

테이블 이름 뒤에 물음표(?)를 붙이면 해당 테이블이 존재하지 않아도 오류가 발생하지 않습니다. 가령, sys\_status 테이블이 존재하지 않을때 table sys\_status 쿼리를 실행하면 오류가 발생하지만, table sys\_status? 쿼리를 실행하면 오류 없이 쿼리가 수행되며 결과 0건이 출력됩니다.

테이블 이름에 와일드카드(\*)를 사용할 수 있습니다. 예를 들어, 쿼리문 table sys\_\*를 실행하면 sys\_로 시작하는 모든 테이블을 모두 조회합니다. 읽기 권한이 없는 테이블은 검색에서 제외됩니다. 쿼리를 실행한 다음 **\_table** 필드에서 테이블 이름을 확인할 수 있습니다.

**meta("KEY\_VALUE\_EXPR", "TABLE"[, ... ])**

조회할 테이블의 메타데이터. 지정한 KEY\_VALUE\_EXPR 속성과 일치하는 테이블을 선택할 때 사용합니다. meta()는 table 명령어에서만 사용할 수 있습니다.

**KEY\_VALUE\_EXPR**

"KEY == VALUE" 또는 "KEY != VALUE" 형태의 비교 표현식. and, or와 같은 논리 연산자나 not 단항 부정 연산자와 함께 사용할 수 있습니다.

**KEY**

테이블 메타데이터의 키 이름

**VALUE**

테이블 메타데이터의 값(와일드카드 패턴 사용 가능)

**"TABLE"[, ... ]**

조회할 테이블 이름. 아무 테이블도 지정하지 않으면 모든 테이블에 메타데이터 조건식을 시도합니다.

ENT, STD는 "**테이블 관리 > (테이블 선택) [기본정보]**"에서 테이블 메타데이터를 지정할 수 있습니다. 파서 설정의 메타데이터 키는 logparser입니다.

선택 매개변수

duration, from, to과 같은 옵션으로 검색 기간/범위를 지정하지 않으면 모든 데이터를 검색합니다.

**duration=INT{mon|w|d|h|m|s}**

현재 시각을 기준으로 지정한 시간만큼 이전 데이터만 검색. mon(월), w(주), d(일), h(시), m(분), s(초) 단위로 입력합니다. 예를 들어 10s은 쿼리 실행 시각을 기준으로 "최근 10초"를 의미합니다. 이 옵션은 from, to, window와 함께 사용할 수 없습니다.

**from=yyyyMMddHHmmss**

검색할 기간의 시작 날짜를 yyyyMMddHHmmss 형식으로 지정. 입력한 시각부터 검색을 시작합니다. 앞부분만 입력하면 나머지 자리는 0으로 인식합니다. 예를 들어 20130605를 입력하면 20130605000000(2013년 6월 5일 0시 0분 0초)으로 인식합니다. 이 옵션은 to와 함께 사용할 수 있지만, duration, window는 함께 사용할 수 없습니다.

**to=yyyyMMddHHmmss**

검색할 기간의 끝 날짜를 yyyyMMddHHmmss 형식으로 지정. 입력한 시각은 검색 범위에 포함되지 않습니다. 입력 방식은 from과 같습니다. from과 함께 사용할 수 있지만, duration, window는 함께 사용할 수 없습니다.

**window=INT{y|mon|w|d|h|m|s}**

쿼리 실행 시점부터 어느 정도의 시간 동안 테이블에 입력되는 데이터를 실시간으로 수신할 것인지 지정. y(연), mon(월), w(주), d(일), h(시), m(분), s(초) 단위로 지정할 수 있습니다. 단위가 y일 때, 1y만 허용됩니다. 예를 들어 10s은 쿼리 실행 시각을 기준으로 “앞으로 10초”입니다. 이 옵션은 duration, from, to와 함께 사용할 수 없습니다.

**limit=INT**

가져올 최대 로그 개수(기본값: 제한 없음)

**offset=INT**

건너뛸 로그 개수(기본값: 0).

**order=STR**

레코드의 정렬 순서 (기본값: desc)

* asc: 오름차순 정렬. 오래된 레코드부터 출력.
* desc: 내림차순 정렬. 최근 레코드부터 출력.

#### 사용 예

sys\_cpu\_logs 테이블에서 최근 100건의 로그를 조회

table limit=100 sys\_cpu\_logs

sys\_cpu\_logs 테이블에서 최근 10분 간의 로그를 조회

table duration=10m sys\_cpu\_logs

sys\_cpu\_logs 테이블에서 2013년 6월 5일의 전체 로그를 조회

table from=20130605 to=20130606 sys\_cpu\_logs

sys\_cpu\_logs 테이블과 sys\_mem\_logs 테이블의 모든 로그를 순서대로 조회

table sys\_cpu\_logs, sys\_mem\_logs

테이블의 파서(parser)가 openssh인 테이블만 조회

table meta("parser==openssh")

### textfile

텍스트 파일에서 데이터를 읽어옵니다.

#### 문법

textfile [OPTIONS] PATH

필수 매개변수

**PATH**

파일 경로. 절대경로로 입력하거나, 로그프레소 엔진을 기준으로 상대경로로 지정할 수 있습니다. 파일 이름에 와일드카드(\*)를 사용해 패턴 매칭 방식으로 파일을 조회할 수 있습니다. 예를 들어, PATH에 allow-\*.txt를 지정함으로써 allow-ip.txt, allow-user.txt, allow-url.txt 등의 파일을 한꺼번에 조회할 수 있습니다. 파일을 읽어오려면 로그프레소 실행 계정에 접근 권한이 부여되어 있어야 합니다.

선택 매개변수

**brex="REGEX"**

하나의 레코드가 여러 행으로 구성된 경우, 레코드의 시작 행을 찾을 때 사용할 정규표현식. 정규표현식이 일치하는 행이 나오기 전 행 혹은 erex로 지정한 마지막 행까지 하나의 레코드로 병합합니다. 지정하지 않으면 개행문자(CRLF 혹은 LF)를 기준으로 인식합니다.

레코드의 마지막 행을 찾으려면 erex 옵션을 이용합니다.

**cs=CHARSET**

문자열 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**df="TIME\_FMT"**

dp로 추출한 날짜 데이터를 파싱할 때 사용할 형식 문자열. yyyy-MM-dd HH:mm:ss.SSS와 같이 입력할 수 있습니다. dp 옵션과 함께 사용합니다. 지정하지 않으면 데이터 로딩 시점의 시각을 **\_time** 필드에 기록합니다.

형식 문자열에 다음과 같은 날짜 지시자를 사용할 수 있습니다.

|  |  |  |
| --- | --- | --- |
| 지시자 | 설명 | 예제 |
| G | 기원 전/후 | AD |
| y | 연도 | 1996; 96 |
| M | 월 | July; Jul; 07 |
| w | 연 단위 주 | 27 |
| W | 월 단위 주 | 2 |
| d | 연 단위 일 | 189 |
| D | 월 단위 일 | 10 |
| F | 요일 번호 | 2 |
| E | 요일 | Tuesday; Tue |
| u | 요일 숫자(1=월요일, …, 7=일요일 | 1 |
| a | 오전/오후 | PM |
| H | 시간(0-23) | 0 |
| k | 시간(1-24) | 24 |
| K | 오전/오후 시간 (0-11) | 0 |
| h | 오전/오후 시간 (1-12) | 12 |
| m | 분 | 30 |
| s | 초 | 55 |
| S | 밀리초 | 978 |
| z | 시간대(일반 표기) | Pacific Standard Time; PST |
| Z | 시간대(RFC 822 표기) | -0800 |
| X | 시간대(ISO 8601 표기) | -08;-0800;08:00 |

예를 들어, 읽어온 데이터에 2000-01-01 11:22:33과 같은 문자열이 있고, 이와 유사한 문자열에서 날짜 데이터를 추출할 형식 문자열이 yyyy-MM-dd HH:mm:ss 라면 **\_time** 필드에 2000년 1월 1일 11시 22분 33초가 기록됩니다. df를 지정하지 않으면 데이터를 읽어들인 시점의 시각을 **\_time** 필드에 기록합니다.

**dp="REGEX"**

**\_time** 필드에 기록할 날짜 추출 정규표현식. 예를 들어, 읽어온 데이터에 2000-01-01 11:22:33과 같은 문자열이 있고, 이와 동일한 패턴의 문자열에서 날짜 데이터를 추출하려면 dp에 정규표현식으로 (\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2})를 지정합니다. (정규표현식 메타 문자인 \가 문자열 안에 있으므로 이스케이프 문자로서 \를 추가해 \를 메타 문자로 사용합니다.) dp로 추출한 날짜 데이터는 df로 지정한 지정한 형식으로 파싱합니다. df 옵션과 함께 사용합니다. 이 옵션을 지정하지 않으면 데이터를 읽어들인 시점의 시각을 **\_time** 필드에 기록합니다.

**erex="REGEX"**

하나의 레코드가 여러 행으로 구성된 경우, 레코드의 마지막 행을 찾을 때 사용할 정규표현식. 정규표현식이 일치하는 행이 나오기 전까지, 혹은 파일의 마지막 행부터 파일 끝까지 하나의 레코드로 병합합니다. 지정하지 않으면 개행문자(CRLF 혹은 LF)를 기준으로 인식합니다.

레코드의 시작 행을 찾으려면 brex 옵션을 이용합니다.

**limit=INT**

가져올 최대 로그 개수(기본값: 제한 없음)

**offset=INT**

건너뛸 로그 개수(기본값: 0)

#### 사용 예

/var/log/secure 로그 파일 조회

textfile /var/log/secure

euc-kr로 인코딩 된 iis.txt 파일 조회

textfile cs=euc-kr iis.txt

/var/log에서 파일 이름에 syslog.가 포함된 모든 gz 파일 조회

textfile /var/log/syslog.\*.gz

test.txt 파일 내용 중 첫 5줄은 건너뛴 후 20건 조회

textfile offset=5 limit=20 test.txt

레코드 시작 행은 대괄호()로 감싸여진 문자열을 포함하고 마지막 행은 end 문자열을 포함하는 test.txt 파일 조회

textfile brex="\[.\*\]" erex="end" test.txt

test.txt 파일을 조회하면서 레코드 내에 2000-01-01 11:22:33 형태로 포함된 날짜 데이터를 \_time 필드로 추출

textfile dp="(\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2})" df="yyyy-MM-dd HH:mm:ss" test.txt

#### 호환성

읽을 파일의 확장자가 .gz이면 자동으로 gzip 파일로 인식하여 압축 해제 후 조회할 수 있습니다. 이 기능은 ENT #2241 2019-04-23\_17-20 버전부터 지원합니다.

### xmlfile

XML 파일에서 데이터를 조회합니다. 파일 이름에 와일드카드(\*)를 사용하면 특정 문자열 패턴을 포함한 모든 파일을 한 번에 조회할 수 있습니다.

#### 문법

xmlfile [OPTIONS] FILE\_PATH

필수 매개변수

**FILE\_PATH**

XML 파일의 경로. 파일 이름에 와일드카드(\*)를 사용해 패턴 매칭 방식으로 파일을 조회할 수 있습니다. 예를 들어, FILE\_PATH에 report-\*.xml을 지정함으로써 report-2022-01-01.xml, report-2022-01-02.xml 등의 파일을 한꺼번에 조회할 수 있습니다. 파일을 읽어오려면 로그프레소 실행 계정에 접근 권한이 부여되어 있어야 합니다.

선택 매개변수

**cs=CHARSET**

문자열 인코딩 형식(기본값: utf-8). 이 옵션은 대소문자를 구분하지 않으며, 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 CHARSET으로 사용할 수 있습니다: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**xpath=EXPR**

XML 노드를 선택하는데 사용할 XPath(XML Path Language) 표현식. XPath는 다음 문서를 참조하십시오: <https://www.w3.org/TR/xpath-31/>.

#### 사용 예

euc-kr로 인코딩된 report\_kr.xml 파일 조회

xmlfile cs=euc-kr report\_kr.xml

books.xml 파일에서 bookstore 노드의 하위 노드들 중에서 첫번째 책의 title 노드 정보 조회

# 다운로드: https://raw.githubusercontent.com/logpresso/dataset/main/books.xml | xmlfile xpath="/bookstore/book[1]/title" books.xml

#### 호환성

이 명령어는 ENT #2241 2019-04-23\_17-20 버전부터 지원합니다.

### zipfile

ZIP으로 압축된 텍스트 파일에서 데이터를 조회합니다.

#### 문법

zipfile [limit=INT] [offset=INT] ZIP\_PATH FILE\_IN\_ZIP

필수 매개변수

**ZIP\_PATH**

ZIP 파일의 절대 경로를 입력합니다. 파일 경로에 와일드카드(\*)를 사용하면 특정 문자열 패턴을 포함한 모든 파일을 한 번에 조회할 수 있습니다. 파일을 읽어오려면 로그프레소 실행 계정에 접근권한이 부여되어 있어야 합니다.

**FILE\_IN\_ZIP**

ZIP 파일에 포함된 텍스트 파일 중에서 데이터를 읽어들일 파일 이름을 입력합니다. 파일 이름에 와일드카드(\*)를 사용하면 특정 문자열 패턴을 포함한 모든 파일을 한 번에 조회할 수 있습니다.

선택 매개변수

**limit=INT**

가져올 최대 레코드 개수(기본값: 제한 없음)

**offset=INT**

건너뛸 레코드 개수(기본값: 0)

#### 사용 예

/opt/logpresso/testdata.zip 파일에 압축된 텍스트 파일 중 iis.txt 파일 조회

zipfile /opt/logpresso/testdata.zip iis.txt

/opt/logpresso/testdata.zip 파일에 압축된 모든 텍스트 파일 조회

zipfile /opt/logpresso/testdata.zip \*.txt

/opt/logpresso 경로의 모든 ZIP 파일에 압축된 모든 텍스트 파일 조회

zipfile /opt/logpresso/\*.zip \*.txt

## 데이터 가공

### alertmsg

경보 알림 메시지의 언어 로케일을 변경합니다.

#### 문법

alertmsg [locale=LOCALE\_CODE]

선택 매개변수

**locale=LOCALE\_CODE**

경보 알림 메시지의 언어 로케일(기본값: en). 현재 지원하는 언어는 영어(en), 한국어(ko)입니다.

#### 설명

alretmsg 명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| category | 문자열 | 경보 유형 |
| description | 문자열 | 경보 설명 |
| message | 문자열 | 경보 메시지 |
| name | 문자열 | 경보 이름 |

#### 사용 예

경보 알림 메시지를 영어로 출력

table sys\_alerts | alertmsg

경보 알림 메시지를 한국어로 출력

table sys\_alerts | alertmsg locale=ko

### auditmsg

감사 로그의 로케일을 사용자가 지정한 언어로 변경합니다.

#### 문법

auditmsg locale=LOCALE\_CODE

필수 매개변수

**locale=LOCALE\_CODE**

사용자 세션에서 감사 로그 메시지에 적용할 언어 로케일(기본값: en). 현재 지원하는 언어는 영어(en), 한국어(ko)입니다.

#### 사용 예

sys\_audit\_log에 저장된 감사 로그를 한국어로 변환

table sys\_audit\_logs | auditmsg locale=ko

### boxplot

상자 그림(box plot)을 그리는데 필요한 최소, 최대, 사분위수를 계산합니다.

#### 문법

boxplot EXPR [by GRP\_FIELD, ...]

필수 매개변수

**EXPR**

통계 대상이 되는 계산 수식을 입력합니다.

선택 매개변수

**by GRP\_FIELD, ...**

상자 그림의 대상이 되는 필드 이름을 입력합니다. 그룹 단위로 나누어 최소, 최대, 사분위수를 구하려면 그룹을 구분하는 기준이 될 필드 이름을 쉼표(,)로 구분하여 입력합니다. 이 by 절은 EXPR 바로 뒤에 사용해야 합니다.

#### 설명

출력 필드는 아래와 같습니다:

* **count**: GRP\_FIELD 그룹별 레코드의 총개수. GRP\_FIELD를 지정하지 않은 경우, 전체 레코드의 개수.
* **GRP\_FIELD**: by 절에 입력된 그룹 키의 값
* **iqr1**: 그룹별 제1사분위수. 중앙값을 기준으로 하위 50%의 중앙값, 전체 데이터 중 하위 25%
* **iqr2**: 그룹별 제2사분위수(중앙값). 데이터를 순서대로 정렬했을 때 가장 중앙에 위치하는 값
* **iqr3**: 그룹별 제3사분위수. 중앙값을 기준으로 상위 50%의 중앙값, 전체 데이터 중 상위 25%
* **max**: 그룹별 최대값
* **min**: 그룹별 최소값

#### 사용 예

전체 CPU 부하에 대한 통계 요약

table sys\_cpu\_logs | eval usage = kernel + user | boxplot usage

일자별 CPU 부하에 대한 통계 요약

table sys\_cpu\_logs | eval day = string(\_time, "yyyy-MM-dd") | eval usage = kernel + user | boxplot usage by day

### bypass

모든 입력 값을 그대로 출력합니다. 모든 입력 필드에 대해 필드 인덱스를 생성하거나 스트림 쿼리에서 모든 결과를 그대로 통과시키는 목적으로 사용합니다.

#### 문법

bypass

### cube

집계 함수 사용 시 모든 항목별 합계가 필요한 경우 cube 함수를 사용합니다. by 절에 여러 필드가 있을 경우 모든 필드 조합에 대한 합계값을 표시합니다.

#### 문법

cube [OPTIONS] AGGR\_FUNC [as ALIAS], ... [by GRP\_FIELD, ...]

#### 필수 매개변수

**AGGR\_FUNC [as ALIAS], ...**

합계를 계산할 대상 [집계 함수](https://docs.logpresso.com/ko/query/section-aggregate-functions)(AGGR\_FUNC)와 필드 이름으로 사용할 별칭(ALIAS). 별칭을 지정하지 않으면 avg()와 같이 집계 함수 이름을 필드 이름으로 사용하므로 ALIAS의 사용을 권장합니다.

#### 선택 매개변수

**label=FIELD**

집계값에 부여할 레이블 필드(기본값: null)

**parallel=BOOL**

쿼리 결과를 병렬로 출력 여부(기본값: f)

* t: 쿼리 결과를 병렬로 출력. 쿼리 결과를 병렬로 출력하면 처리 속도가 증가하지만, 데이터의 순서를 보장하지 않습니다.
* f: 쿼리 결과를 병렬로 출력하지 않음

**by GRP\_FIELD, ...**

집계 대상 필드를 지정하는 by 절. 필드 구분자로 쉼표(,)를 사용합니다. 이 옵션은 AGGR\_FUNC [as ALIAS] 뒤에 사용해야 합니다.

#### 사용 예

웹 서버 로그 테이블 web\_access에서 레코드를 검색하여 **date** 필드 및 **status** 필드의 모든 순열에 대한 카운트의 부분합 및 총합계를 계산

table web\_access | eval date=string(date, "yyyy-MM-dd") | cube label="TOTAL\_COUNT" count by date, status

**action** 및 **status** 필드의 값에 의해 생성된 모든 조합의 개수 및 크기 집계를 계산(라벨은 TOTAL로 표시)

cube label=TOTAL count, sum(size) as size by action, status

#### 호환성

cube 명령어는 ENT #1804 2017-11-28\_13-31 버전부터 지원합니다.

### curvefit

입력 레코드 값에 대하여 최소제곱법을 이용한 선형회귀분석을 수행합니다.

#### 문법

curvefit [degree=INT] INDEPENDENT\_FIELD, DEPENDENT\_FIELD

필수 매개변수

**INDEPENDENT\_FIELD**

독립변수로 사용할 필드. 독립변수 값은 숫자형이어야 합니다.

**DEPENDENT\_FIELD**

종속변수로 사용할 필드. 종속변수 값은 숫자형이어야 합니다.

선택 매개변수

**degree=INT**

입력값을 근사시킬 다항 함수의 차수(기본값: 3)

#### 설명

최대 1만개의 입력 레코드 값에 대하여 최소제곱법을 이용한 선형회귀분석을 수행합니다. 독립변수 필드 값을 **\_x** 필드로, 그리고 계산된 함수 값을 **\_p** 필드로 출력합니다. 1만 개 이후의 레코드는 무시하고 쿼리를 종료합니다.

#### 사용 예

최근 1시간의 CPU 사용율 추이를 10차 다항함수로 근사

table duration=1h sys\_cpu\_logs| eval x = datediff(dateadd(now(), "hour", -1), \_time, "sec")| eval total = kernel + user| curvefit degree=10 x, total

### decodedhcp

DHCP 패킷을 디코딩합니다.

#### 문법

decodedhcp

#### 설명

출력하는 필드는 다음과 같습니다.

* **client\_ip**: DHCP 클라이언트의 IP 주소. IP 주소를 할당받지 못한 클라이언트의 주소는 0.0.0.0입니다.
* **client\_mac**: DHCP 클라이언트의 MAC 주소
* **fingerprint**: DHCP 핑거프린트가 있는 경우 표시. 각 번호의 의미는 IANA가 관리하는 DHCP 및 BOOTP 매개변수 목록을 참조: <https://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml>
* **gateway\_ip**: DHCP 릴레이 에이전트가 클라이언트를 대신해 DHCP 서버와 통신할 때 지정하는 게이트웨이 IP 주소. 일반적으로 DHCP 서버가 DHCP 에이전트와 통신할 수 있는 IP 주소
* **next\_server\_ip**: 보조 DHCP 서버의 IP 주소
* **options**: DHCP 옵션 번호를 배열로 보여주는 필드.
* **tx\_id**: DHCP 트랜잭션 식별자
* **your\_ip**: DHCP 서버가 클라이언트에게 할당하는 IP 주소

#### 사용 예

임의의 PCAP 파일에서 DHCP 통신만 디코딩

pcapfile /opt/logpresso/pcap/dhcp.pcap | decodedhcp

### decodedns

PCAP 파일에서 DNS 쿼리 내용을 디코딩해 보여줍니다. 패킷 단위로 디코딩하려면 pcapdecode 명령을 이용하십시오.

#### 문법

decodedns

#### 설명

출력하는 필드는 다음과 같습니다.

* **additionals**: 기타 관련 레코드(배열)
* **answers**: DNS 서버의 응답 메시지 원본(배열)
* **authorities**: 권한있는 DNS 서버에 관한 정보(배열)
* **bytes**: DNS 응답 페이로드의 크기(정수)
* **direction**: 트랜잭션 방향(문자열)
* c->s: 클라이언트의 요청
* s->c: 서버의 응답
* **domain**: 질의 대상 도메인 주소(문자열)
* **dst\_ip**: DNS 트랜잭션의 목적지 IP 주소. 일반적으로 DNS 서버의 주소(IP 주소)
* **dst\_port**: DNS 트랜잭션의 목적지 포트(정수)
* **flags**: DNS 헤더 플래그. 참조: [https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-12](https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml)
* **ip**: 도메인 주소와 연결된 IP 주소(IP 주소)
* **src\_ip**: DNS 트랜잭션의 출발지 IP 주소(IP 주소)
* **src\_port**: DNS 트랜잭션의 출발지 포트(정수)
* **status**: 쿼리 실행 결과(오류가 있으면 오류 메시지 표시)
* FORMAT\_ERROR: 질의 내용에 오류가 있어 서버가 처리할 수 없음
* NAME\_ERROR: 질의한 도메인 주소가 존재하지 않음
* NO\_ERROR: 오류 없음
* NOT\_IMPLEMENTED: DNS 서버가 요청한 질의를 지원하지 않음
* REFUSED: DNS 서버의 응답 거부
* SERVER\_FAILURE: 서버 오류로 질의를 처리할 수 없음
* **txid**: DNS 트랜잭션 ID(16진수 문자열)
* **type**: DNS 레코드 타입(A, AAAA, CNAME, MX, NS, PTR, SOA, SRV, TXT). 참조: [https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4](https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml).

#### 사용 예

임의의 PCAP 파일에서 DNS 통신만 디코딩

pcapfile /opt/logpresso/pcap/abnormal\_traffic.pcap | decodedns

#### 호환성

decodedns 명령어는 ENT #2309 2019-11-27\_10-43 버전부터 지원합니다.

### decodehttp

패킷에서 HTTP 헤더를 디코딩합니다.

#### 문법

decodehttp

#### 설명

출력하는 필드는 다음과 같습니다.

* **dst\_ip**: 목적지 IP 주소(IP 주소)
* **dst\_port**: 목적지 포트(정수)
* **host**: FQDN(Fully Qualified Domain Name) 형식의 웹 서버 이름(문자열)
* **method**: HTTP 메서드(문자열)
* **path**: 리소스 경로(문자열). 일반적으로 URI(Uniform Resource Identifier). 참조: <https://tools.ietf.org/html/rfc3986>
* **rcvd**: 받은 데이터(정수, 단위: 바이트)
* **req\_time1**: 첫번째 HTTP 요청 시간(epoch 형식 시간)
* **req\_time2**: 두 번째 HTTP 요청 시간(epoch 형식 시간)
* **res\_time1**: 첫번째 응답 시간(epoch 형식 시간)
* **res\_time2**: 두 번째 응답 시간(epoch 형식 시간)
* **sent**: 보낸 데이터(정수, 단위: 바이트)
* **src\_ip**: 출발지 IP 주소(IP 주소)
* **src\_port**: 출발지 포트(정수)
* **status**: 서버의 HTTP 응답 코드(정수). 참조: <https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml>

#### 사용 예

임의의 PCAP 파일에서 HTTP 통신만 디코딩

pcapfile /opt/logpresso/pcap/abnormal\_traffic.pcap | decodehttp

### decodesflow

sFLow 패킷을 디코딩합니다.

#### 문법

decodesflow

#### 설명

출력하는 필드는 다음과 같습니다.

* **agent\_addr**: 에이전트의 IP 주소
* **agent\_id**: 에이전트 식별자
* **counters**: **sampling\_type**이 counters일 때 다음과 같은 맵 형태 정보를 출력
* **admin\_status**: 관리자 포트 활성화 여부(true, false)
* **if\_direction** (0: 알 수 없음, 1: Full-duplex, 2: Half-duplex, 3: 수신, 4: 송신)
* **if\_index**: 인터페이스 식별자
* **if\_speed**: 연결 링크 bps
* **if\_type**: 이더넷은 6, 그 외 번호는 IANA가 할당한 인터페이스 타입 표준 번호 참조: <https://ietf.org/assignments/ianaiftype-mib/ianaiftype-mib>("IANAifType ::= TEXTUAL-CONVENTION", "SYNTAX INTEGER" 섹션)
* **in\_bcast\_pkts**: 받은 브로드캐스트 패킷 개수
* **in\_discards**: 받은 패킷 중 버려진 개수
* **in\_errors**: 받은 패킷 중 오류가 있는 패킷 개수
* **in\_mcast\_pkts**: 받은 멀티캐스트 패킷 개수
* **in\_octets**: 받은 용량(bytes)
* **in\_ucast\_pkts**: 받은 유니캐스트 패킷 개수
* **in\_unknown\_protos**: 받은 패킷 중에서 프로토콜을 알 수 없는 패킷 개수
* **oper\_status**: 실제 링크 활성화 여부(true, false)
* **out\_bcast\_pkts**: 보낸 브로드캐스트 패킷 개수
* **out\_discards**: 보낼 패킷 중에서 버려진 개수
* **out\_errors**: 보낼 패킷 중에서 오류가 있는 패킷 개수
* **out\_mcast\_pkts**: 보낸 멀티캐스트 패킷 개수
* **out\_octets**: 보낸 용량(bytes)
* **out\_ucast\_pkts**: 보낸 유니캐스트 패킷 개수
* **promisc\_mode**: 모든 패킷 수신 활성화 여부(true, false)
* **drops**: 성능 부족으로 손실된 패킷 개수. **sampling\_type**이 flow일 때 정보를 출력
* **dst\_ip**: 목적지 IP 주소. 일반적으로 sFlow 수집 서버의 주소를 표시.
* **dst\_port**: 목적지 포트 번호
* **flow**: **sampling\_rate**에 따라 무작위로 샘플링한 패킷
* **flow\_seq**: **src\_id**별로 샘플을 생성할 때마다 1씩 증가
* **flows**: **sampling\_type**이 flow일 때 샘플링한 플로우 데이터 정보를 출력
* **frame\_length**: 샘플링 하기 전 패킷의 길이(bytes)
* **header**: 이더넷 헤더 옥텟 바이트 스트림
* **protocol**: 연결 계층 프로토콜(예: ethernet)
* **stripped**: 연결 계층 헤더 옥텟을 추출하기 전에 패킷에서 제거된 옥텟 개수
* **input\_if\_index**: **sampling\_type**이 flow일 때 받은 인터페이스 식별자 정보를 출력
* **output\_if\_index**: **sampling\_type**이 flow일 때 보낸 인터페이스 식별자 정보를 출력
* **protocol**: 전송 계층 프로토콜(udp)
* **sample\_pool**: 샘플링 대상 원본 개수. **sampling\_type**이 flow일 때 정보를 출력
* **sample\_type**: 샘플링 종류
* **sampling\_rate**: 샘플링 비율. 지정된 패킷 개수 중에서 1개 추출. **sampling\_type**이 flow일 때 정보를 출력
* **src\_id**: 인터페이스 식별 번호
* **src\_id\_type**: RFC 2613에 정의된 인터페이스 타입(0: ifIndex, 1: smonVlanDataSource, 2: entPhysicalEntry). 관련 내용은 RFC 2613, "Remote Network Monitoring MIB Extensions for Switched Networks Version 1.0"에서 3.1.1 DataSource Objects 참조: <https://tools.ietf.org/html/rfc2613>
* **src\_ip**: 출발지 IP 주소. 일반적으로 sFlow 에이전트의 주소를 표시.
* **src\_port**: 출발지 포트 번호
* **uptime**: 에이전트의 부팅 후 가동 시간
* **ver**: sflow 버전(버전 5만 지원)

#### 사용 예

pcapfile /opt/logpresso/sonar/sflow.cap | decodesflow

### eval

우변의 표현식을 평가하여 새로운 필드를 할당하거나 기존의 필드 값을 대체합니다. 우변에는 값으로 평가될 수 있는 모든 조합의 표현식을 입력할 수 있습니다.

#### 문법

eval FIELD=EXPR, ...

필수 매개변수

**FIELD=EXPR, ...**

표현식을 새 필드 또는 기존 필드에 할당하는 구문. 구문을 여러 개 지정하려면 구분자로 쉼표(,)를 사용합니다. 여러 개의 FIELD=EXPR 쌍이 있으면 왼쪽부터 순서대로 평가합니다.

#### 사용 예

[int("100")](https://docs.logpresso.com/ko/query/int-function) 함수를 실행해 문자열을 정수로 변환한 다음 **num** 필드에 할당

[typeof()](https://docs.logpresso.com/ko/query/typeof-function) 함수를 호출해 특정 필드의 타입을 **type1**, **type2** 필드에 할당

json "{}" | eval type1 = typeof("string"), type2 = typeof(100)

필드 값 합산

json "{}" | eval sent = 100, rcvd = 200, total = sent + rcvd

[concat("hello", ", world")](https://docs.logpresso.com/ko/query/concat-function) 함수를 실행한 결과를 **msg** 필드에 할당

json "{}" | eval msg = concat("hello", ", world")

### evalc

우변의 표현식을 평가하여 새로운 쿼리 매개변수를 할당하거나 기존의 쿼리 매개변수 값을 대체합니다.

#### 문법

evalc VAR=EXPR

필수 매개변수

**VAR=EXPR**

쿼리 매개변수. 왼쪽에 매개변수 이름을, 오른쪽에 값으로 평가할 수 있는 표현식의 조합을 지정합니다. 표현식을 평가하여 얻은 값은 쿼리 매개변수에 할당됩니다. [set](https://docs.logpresso.com/ko/query/set-command)와 달리 쿼리 실행 시점에 모든 데이터에 대해 평가합니다.

#### 사용 예

count가 임계치가 넘는 경우 쿼래 매개변수 alert에 true를 할당

evalc alert = if(count > 100000, true, $("alert"))

모든 입력 데이터에 대해 우변 표현식 평가 후 대입되므로, 임계치를 넘지 않는 경우 [$()](https://docs.logpresso.com/ko/query/$-function) 함수를 이용하여 기존 변수 값을 그대로 다시 대입하도록 해야합니다.

### explode

지정된 배열의 각 원소마다 대응되는 행을 생성합니다. 일반적으로 배열(가로)을 열(세로) 방향으로 축 변환하려는 경우에 사용합니다. 지정된 필드가 존재하지 않거나, 배열이 아니거나, null인 경우 입력 행을 보존합니다.

#### 문법

explode FIELD

필수 매개변수

**FIELD**

배열을 포함하는 필드의 이름

#### 사용 예

IP 주소 통계 추출하기

json "[{line: '10.0.0.1 10.0.0.2'},{line:'10.0.0.2 10.0.0.3'}]" | eval ip = split(line, " ") | explode ip | stats count by ip

[json](https://docs.logpresso.com/ko/query/json-command): 원본 데이터([{line: '10.0.0.1 10.0.0.2'},{line:'10.0.0.2 10.0.0.3'}])를 생성합니다.

[eval](https://docs.logpresso.com/ko/query/eval-command): [split(line, " ")](https://docs.logpresso.com/ko/query/split-function) 함수를 평가한 값(배열 형식)을 새 필드인 ip에 할당합니다.

[explode](https://docs.logpresso.com/ko/query/explode-command): **ip** 필드에 저장된 배열의 요소 개수만큼 4개의 행을 생성합니다.

[stats](https://docs.logpresso.com/ko/query/stats-command): **ip** 필드의 값(IP 주소)을 기준으로 집계 함수 [count](https://docs.logpresso.com/ko/query/count-aggregate-function)를 실행해 값을 집계합니다. 그룹핑 함수를 지정하지 않았으므로 전체 레코드 개수를 반환합니다.

### fields

특정한 필드만 출력하거나, 특정한 필드만 선택적으로 제외합니다.

#### 문법

fields [-] FIELD, ...

필수 매개변수

**FIELD, ...**

쉼표(,)를 구분자로 하는 필드 목록

선택 매개변수

**-**

FIELD, ...에 열거한 필드들을 출력에서 생략(기본값: 사용 안 함)

#### 사용 예

**src\_ip**와 **action** 필드만 출력

fields src\_ip, action

**line** 필드만 제거

fields – line

### flowsearch

서브쿼리로 정의된 IP 네트워크 대역, 포트, 프로토콜 조건으로 구성된 플로우 규칙을 읽어들여 입력 레코드와 대조하고, 검색된 모든 플로우 식별자를 **\_flow** 필드에 배열로 출력합니다.

#### 문법

flowsearch [ SUBQUERY ]

필수 매개변수

**[ SUBQUERY ]**

플로우 규칙을 정의하는 서브쿼리를 대괄호쌍([ ])으로 감싸서 입력합니다.

#### 설명

플로우 규칙은 파일, 테이블, 원격 RDBMS 등 임의의 위치에서 읽어올 수 있으며, 필드 구성과 타입이 일치해야 유효한 규칙으로 인식합니다. 서브쿼리로 적용할 수 있는 플로우 규칙은 10,000개를 초과할 수 없습니다. 10,001번째 규칙부터는 무시합니다.

서브쿼리가 실패하면 **\_flowsearch\_error** 필드에 오류 원인을 출력합니다. flowsearch 명령문 뒤에 **\_flowsearch\_error** 필드 값이 존재하는지 검사하는 예외 처리문을 두면 의도하지 않은 오류 혹은 오동작을 방지할 수 있습니다.

입력 필드

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 이름 | 타입 | 필수 여부 | 설명 |
| src\_ip | IP 주소 | 필수 | 출발지 IP 주소 |
| src\_port | 정수 | 선택 (null 허용) | 출발지 포트 |
| dst\_ip | IP 주소 | 필수 | 목적지 IP 주소 |
| dst\_port | 정수 | 선택 (null 허용) | 목적지 포트 |
| protocol | 문자열 | 선택 (null 허용) | 프로토콜 문자열 |

입력 레코드의 필드가 타입이 일치하지 않거나 필수 필드가 누락된 경우에는, 플로우 규칙을 검사하지 않고 레코드를 원본 그대로 출력합니다.

플로우 규칙 필드

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 이름 | 타입 | 필수 여부 | 설명 |
| src\_ip | IP 주소 | 필수 | 출발지 IP 주소 |
| src\_cidr | 정수 | 필수 | 출발지 넷마스크 (0-32 범위의 정수) |
| src\_port | 정수 | 선택 (null 허용) | 출발지 포트 번호 (0-65535 범위의 정수) |
| dst\_ip | IP 주소 | 필수 | 목적지 IP 주소 |
| dst\_cidr | 정수 | 필수 | 목적지 넷마스크 (0-32 범위의 정수) |
| dst\_port | 정수 | 선택 (null 허용) | 목적지 포트 번호 (0-65535 범위의 정수) |
| protocol | 문자열 | 선택 (null 허용) | 프로토콜 문자열 (TCP, UDP, ICMP 등) |
| flow | 임의 타입 | 필수 | 플로우 식별자 |

즉, flowsearch 커맨드에 레코드가 1개 입력될 때마다 입력된 5-튜플 값을 플로우 규칙과 대조하여, 일치하는 플로우 식별자를 \_flow 필드에 목록으로 출력합니다.

플로우 규칙의 src\_ip, dst\_ip와 src\_cidr, dst\_cidr 필드는 모두 필수 입력이지만, 규칙의 src\_ip가 0.0.0.0 이고 src\_cidr이 0인 경우 모든 출발지 IP 주소에 대해 참이 되므로, 출발지 혹은 목적지에 대해 모든 값을 허용하려면 규칙에 0.0.0.0/0을 설정하면 됩니다.

예를 들어 아래의 플로우 규칙에 대해, 입력 레코드가 src\_ip=106.75.11.63, src\_port=57776, dst\_ip=106.246.20.67, dst\_port=80, protocol=TCP 인 경우, flow2가 일치하므로 출력 레코드에는 \_flow=["flow2"] 필드가 추가됩니다.

**플로우 규칙 예시**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| src\_ip | src\_cidr | src\_port | dst\_ip | dst\_cidr | dst\_port | protocol | flow |
| 211.36.133.0 | 24 | null | 106.246.20.67 | 32 | 80 | TCP | flow1 |
| 106.75.11.0 | 24 | null | 106.246.20.67 | 32 | null | TCP | flow2 |

#### 사용 예

json "{}"| eval src\_ip=ip("106.75.11.63"), src\_port=57776| eval dst\_ip=ip("106.246.20.67"), dst\_port=80, protocol="TCP"| # Initiating the flowsearch command that defines the flow search rule| flowsearch [ union [ json "{}" | eval src\_ip=ip("211.36.133.0"), dst\_ip=ip("106.246.20.67"), flow="flow1" ] | union [ json "{}" | eval src\_ip=ip("106.75.11.0"), dst\_ip=ip("106.246.20.67"), flow="flow2" ] | eval src\_cidr=24, dst\_cidr=32]| fields src\_ip, src\_port, dst\_ip, dst\_port, protocol, \_flow

### groovy

Groovy로 작성된 스크립트를 실행합니다.

#### 문법

groovy CLASS\_NAME

필수 매개변수

**CLASS\_NAME**

실행할 클래스의 이름.

#### 설명

Groovy는 Python, Ruby와 같은 언어의 영향을 받아 개발된 동적 객체 지향 언어로 JVM에서 동작합니다. 실행할 수 있는 스크립트 파일은 다음과 같은 제약 조건을 만족해야 합니다.

* 스크립트 파일 이름은 다음과 같은 형식으로 지정해야 합니다: CLASS\_NAME.groovy
* 로그프레소 설치 디렉터리 아래 **data/araqne-logdb-groovy/query\_scripts** 디렉터리에 있는 Groovy 스크립트만 실행할 수 있습니다.
* 로그프레소가 제공하는 패키지를 임포트해서 사용해야 합니다. 다음과 같은 패키지를 필요에 따라 사용하십시오.
* groovy.transform.CompileStatic (성능상 이점이 있으므로 권장)
* org.araqne.logdb.groovy.GroovyQueryScript (필수)
* org.araqne.logdb.QueryStopReason
* org.araqne.logdb.Row (필수)
* org.araqne.logdb.RowBatch
* org.araqne.logdb.RowPipe

Groovy 스크립트의 성능을 향상시키려면 다음과 같은 사항을 참고하십시오.

* 문자열 처리 메서드는 가능하면 사용하지 않도록 합니다. 문자열 객체가 많아지면 JVM에서 가비지 컬렉션이 빈번하게 일어납니다.
* split(), tokenize() 메서드 대신에 indexOf()이나 substring()을 사용하십시오. 코드는 길어지지만 더 좋은 처리 성능을 제공합니다.
* Pattern.compile()을 반복적으로 사용하지 마십시오. Matcher.reset()을 호출해 Matcher 인스턴스를 재사용하는 방식이 더 좋은 성능을 제공합니다.
* 예외 발생을 최소화하십시오.
* 예외가 빈번하게 발생하면 처리 성능이 현저하게 떨어집니다.
* 가능하다면, 발생할 수 있는 오류 케이스는 조건 검사를 통해 처리하십시오.

#### 사용 예

다음과 같은 스크립트를 **ToAscii.groovy**라는 이름으로 로그프레소 설치 디렉터리 아래 **data/araqne-logdb-groovy/query\_scripts** 디렉터리에 저장합니다.

import groovy.transform.CompileStatic;import org.araqne.logdb.Row;import org.araqne.logdb.groovy.GroovyQueryScript;@CompileStatic(groovy.transform.TypeCheckingMode.SKIP)class ToAscii extends GroovyQueryScript { def void onRow(Row row) { byte[] payload = row.get('payload') char[] chars = new char[payload.length]; for (int i = 0; i < payload.length; i++) { char c = (char) payload[i] if (c < 32 || c > 126) c = '.' chars[i] = c } row.put('text', new String(chars)) pipe.onRow(row) }}

이 스크립트는 PCAP 파일에서 디코딩되어 **payload** 필드에 출력된 바이너리 값 중에서 32번째 문자부터 127번째 문자를 ASCII 형식으로 인코딩해서 보여줍니다.

pcapfile /opt/logpresso/sonar/http-2.pcap | pcapdecode | groovy ToAscii

### limit

지정한 개수만큼 특정 위치에서 쿼리 결과를 읽어오고, 쿼리를 취소합니다.

#### 문법

limit [INT\_OFFSET] INT\_MAX

필수 매개변수

**INT\_MAX**

쿼리 결과에서 가져올 최대 개수. 지정한 쿼리 입력 개수에 도달하면 쿼리를 취소합니다. 일부 명령어는 쿼리가 취소되면 의도한대로 동작하지 않을 수 있으므로 주의해야 합니다.

선택 매개변수

**INT\_OFFSET**

쿼리 결과에서 건너뛸 행 개수(기본값: 0)

#### 사용 예

가장 먼저 입력으로 들어오는 5건만 조회하고 쿼리 취소

table sys\_cpu\_logs | limit 5

위에 예로 든 쿼리문은 다음과 동일한 결과를 갖습니다:

table limit=5 sys\_cpu\_logs

첫번째 레코드는 제외하고 2건만 조회한 후 쿼리 취소

table sys\_cpu\_logs | limit 1 2

위에 예로 든 쿼리문은 다음과 동일한 결과를 갖습니다:

table offset=1 limit=2 sys\_cpu\_logs

### mpsearch

수천 개 이상의 키워드 패턴을 고속으로 한 번에 검색합니다. 서브쿼리에서 지정된 패턴이 검색 대상 필드에서 검출되면, **\_mp\_result** 필드에 검출된 모든 패턴 목록을 포함하여 출력합니다.

#### 문법

mpsearch [FIELD,...] [ SUBQUERY ]

필수 매개변수

**[FIELD,...]**

멀티 패턴을 검색할 대상 필드 목록. 쉼표(,)를 구분자로 사용합니다. 필드를 지정하지 않으면 모든 필드에 대해 검색을 수행합니다.

**[ SUBQUERY ]**

검색할 키워드 패턴의 목록을 조회하는 서브쿼리를 대괄호쌍([ ])으로 감싸서 입력

#### 설명

서브쿼리의 출력은 **expr**, **expr2**, **rule** 문자열 필드를 포함해야 합니다.

* **expr** (필수): 문자열을 불리언 표현식으로 조합하여 작성합니다. 스캔대상 필드의 문자열 값에서 해당되는 문자열을 고속으로 검출한 후, 표현식과 일치하는지 확인합니다.
* **expr2** (선택): **expr** 필드의 문자열 불리언 표현식이 참일 때, 다른 필드의 값을 이용해 추가 검색할 수 있는 기회를 선택적으로 제공합니다.
* **rule** (필수): 패턴 식별자 혹은 이름을 기입합니다.

패턴의 예시는 아래와 같습니다:

**패턴 예시**

|  |  |  |
| --- | --- | --- |
| expr (필수) | expr2 (선택) | rule (필수) |
| "addextendedproc" and "xp\_cmdshell" |  | xp\_cmdshell |
| "REMOTE\_ADDR" and ("fputs" or "fwrite") | path == "lib.php" | zb now\_connect |

만약 xp\_cmdshell 패턴만 탐지되었다면, **\_mp\_result** 필드의 값은 아래와 같습니다:

[ { "expr": "\"addextendedproc\" and \"xp\_cmdshell\"", "rule": "xp\_cmdshell" } ]

#### 사용 예

외부 DB에서 패턴 목록을 로딩하여 signature 필드를 대상으로 멀티 패턴매칭

mpsearch signature [ dbquery RULE\_DB select rule, expr, expr2 from web\_rules ]

### order

출력할 특정한 필드를 지정한 순서로 정렬하고 나머지 필드는 사전순으로 정렬해서 표시합니다.

#### 문법

order FIELD, ...

**FIELD, ...**

순서를 지정할 필드 이름을 순서대로 열거합니다. 구분자로 쉼표(,)를 사용합니다. 여기에 열거하지 않은 필드는 사전순으로 정렬합니다.

#### 사용 예

sys\_cpu\_logs 테이블의 필드 출력 순서를 **kernel**, **idle**, **user**, **\_time**, **\_table**, **\_id** 순서로 정렬

table sys\_cpu\_logs | order kernel, idle, user, \_time, \_table, \_id

sys\_cpu\_logs 테이블의 필드 출력 순서를 **idle**, **kernel**을 가장 먼저 출력하고 나머지는 사전순으로 정렬

table sys\_cpu\_logs | order idle, kernel

#### 호환성

order 명령어는 ENT #1660 2017-07-19\_00-18 버전부터 지원합니다.

### parallel

서브쿼리를 사용하여 입력 데이터를 병렬로 처리하고, 각 서브쿼리 결과를 합쳐서 전달합니다.

#### 문법

parallel core=INT [ SUBQUERY ]

필수 매개변수

**[ SUBQUERY ]**

병렬로 처리할 서브쿼리문을 대괄호쌍([ ]) 안에 입력

**core=INT**

서브쿼리의 병렬 처리에 사용할 CPU의 논리 코어 개수

#### 사용 예

먼저 테스트용 데이터 테이블 생성을 생성합니다.

json "{}" | repeat count=5000000| set a=0 | evalc a=$("a") + 1| eval b=$("a")| fields b| import create=t big\_table

병렬로 서브 쿼리를 실행합니다.

table big\_table | parallel core=4 [eval i=int(b)] | stats count by b | sort -count

### parse

입력 데이터에 특정 파서를 지정하거나, 파싱 규칙을 지정해 출력합니다.

#### 문법

parse [overlay=t] PARSERparse [overlay=t] [field=TARGET\_FIELD] PARSING\_RULE, ...

필수 매개변수

**PARSER**

파서 이름. 사용할 수 있는 파서의 이름은 웹 콘솔에서 확인할 수 있습니다.

* (STD, ENT) **시스템 설정 > 파서/트랜스포머 > 파서에서 이름 필드 확인**
* (MAE, SNR) **수집 > 원본 로그 파서, 정규화 파서**에서 파서 식별자 필드 확인

파서를 지정하면 파서가 출력할 필드가 이미 정의되어 있으므로 field=TARGET\_FIELD 옵션을 함께 사용할 수 없습니다.

**PARSING\_RULE, ...**

사용자 정의 파싱 규칙 목록. 구분자로 쉼표(,)를 사용합니다. 파싱 규칙문의 형식은 "START\_ANCHOR\*STOP\_ANCHOR" as FIELD\_NAME입니다.

* START\_ANCHOR\*STOP\_ANCHOR: 파싱 앵커
* as FIELD\_NAME: 필드 이름으로 사용할 레이블

시작 문자열(START\_ANCHOR)과 끝 문자열(STOP\_ANCHOR)을 인식해 문자열을 파싱하고, as 절로 부여된 레이블을 필드 이름으로 사용합니다.

선택 매개변수

**overlay=BOOL**

원본 데이터의 출력 옵션(기본값: f).

* t: 파싱된 데이터를 필드에 출력하고, 원본 데이터를 **line** 필드에 출력
* f: 파싱된 데이터만 필드에 출력

**field=TARGET\_FIELD**

입력 데이터 스트림에서 파싱할 값이 있는 필드 이름(기본값: line). 이 옵션을 파서(PARSER)와 함께 사용할 수 없습니다.

#### 사용 예

openssh 파서를 이용해 ssh\_log 테이블에 저장된 로그를 파싱

table from=20200601 to=20200701 ssh\_log | parse openssh

로그에서 시작과 끝 텍스트를 지정하여 필드 추출(다음 내용을 'sample.txt'로 저장해서 사용)

Nov 11 00:00:00 session: Proto:17, Policy:pass, Rule:9000, Type:open, Start\_Time:Nov 11 00:00:00, End\_Time:-

아래 명령으로 위와 같은 원본에서 **session**, **proto**, **policy**, **rule**, **end\_time** 필드를 추출할 수 있습니다.

textfile /opt/logpresso/sample.txt | parse "session:\* " as session, "Proto:\*," as proto, "Policy:\*," as policy, "Rule:\*," as rule, "Start\_Time:\*," as start\_time, "End\_Time:\*" as end\_time

### parsecsv

CSV(comma-separated value) 또는 TSV(tab-separated value) 문자열을 파싱합니다.

#### 문법

parsecsv [field=TARGET\_FIELD] [overlay=BOOL] [strict=BOOL] [tab=BOOL] [FIELD, ...]

선택 매개변수

**field=TARGET\_FIELD**

파싱할 값이 저장된 필드 이름(기본값: line)

**overlay=BOOL**

입력 레코드의 원본 필드 출력 여부(기본값: f)

* t: 입력 레코드에 파싱된 결과를 덮어쓴 데이터를 출력
* f: 파싱된 데이터만 필드에 출력

**strict=BOOL**

RFC4180[(https://tools.ietf.org/html/rfc4180)](https://tools.ietf.org/html/rfc4180)의 준수 옵션 (기본값: f)

* t: 마이크로소프트 엑셀과 동일하게 RFC4180 준수: <https://tools.ietf.org/html/rfc4180>. 이 옵션을 tab=t와 함께 사용할 수 없습니다.
* f: CSV 파일을 유연하게 파싱

**tab=BOOL**

탭(tab) 문자를 구분자로 사용 여부 (기본값: f)

* t: 탭(tab) 문자를 구분자로 사용
* f: 쉼표(,)를 구분자로 사용

대상 개체

**FIELD, ...**

파싱된 필드에 사용할 이름 목록. 구분자로 쉼표(,)를 사용합니다. 필드 이름을 지정하지 않으면 이름을 순서대로 column0, column1, ..., colnumnN으로 부여합니다.

#### 사용 예

쉼표로 구분된 텍스트 파싱

json "{line: '\"foo\",\"bar\"'}" | parsecsv

쉼표로 구분된 텍스트를 파싱하여 앞에서부터 순서대로 name1, name2 필드 이름을 부여

json "{line: '\"foo\",\"bar\"'}" | parsecsv name1, name2

### parsejson

JSON 문자열을 파싱합니다.

#### 문법

parsejson [field=TARGET\_FIELD] [overlay=BOOL]

선택 매개변수

**field=TARGET\_FIELD**

파싱할 값이 저장된 필드 이름(기본값: line).

**overlay=BOOL**

원본 데이터의 출력 옵션(기본값: f).

* t: 파싱된 데이터를 필드에 출력하고, 원본 데이터를 **line** 필드에 출력
* f: 파싱된 데이터만 필드에 출력

#### 사용 예

**line** 필드의 JSON 텍스트를 파싱

json "{line: ' {\"foo\": \"bar\"}'}" | parsejson

### parsekv

키와 값의 쌍으로 이루어진 문자열을 파싱합니다.

#### 문법

parsekv [field=TARGET\_FIELD] [kvdelim="CHAR"] [overlay=BOOL] [pairdelim="CHAR"]

선택 매개변수

**field=TARGET\_FIELD**

파싱할 값이 저장된 필드 이름(기본값: line)

**kvdelim="CHAR"**

키와 값을 구분하는 문자(기본값: =).

**overlay=t**

원본 데이터의 출력 옵션(기본값: f).

* t: 파싱된 데이터를 필드에 출력하고, 원본 데이터를 **line** 필드에 출력
* f: 파싱된 데이터만 필드에 출력.

**pairdelim="CHAR"**

키-값 쌍의 구분자로 사용할 문자(기본값: 공백 문자)

#### 사용 예

**line** 필드의 JSON 문자열을 키-값 쌍으로 변환

json "{line: 'src=1.2.3.4 src\_port=55324 dst=5.6.7.8 dst\_port=80'}"| parsekv kvdelim="=" pairdelim=" "

### parsemap

맵 형식 데이터에서 모든 키-값 쌍을 필드로 추출합니다.

#### 문법

parsemap [overlay=BOOL] field=TARGET\_FIELD

필수 매개변수

**field=TARGET\_FIELD**

지정된 입력 필드의 값을 파싱합니다. 대상 필드는 맵 타입이어야 합니다. 대상 필드의 값이 null이거나, 맵 타입이 아니라면, 원본 데이터를 그대로 전달합니다.

선택 매개변수

**overlay=t**

원본 데이터의 출력 옵션(기본값: f).

* t: 파싱된 데이터를 필드에 출력하고, 원본 데이터를 **line** 필드에 출력
* f: 파싱된 데이터만 필드에 출력.

#### 사용 예

**complex** 필드의 맵 데이터에서 모든 키-값 쌍을 필드로 추출

json "{'complex': {'id':100, 'name':'Logpresso'} }" | parsemap field=complex

### parsexml

XML 문서를 복합 객체의 집합으로 파싱합니다.

#### 문법

parsexml [field=TARGET\_FIELD] [overlay=BOOL]

선택 매개변수

**field=TARGET\_FIELD**

입력으로 받는 데이터 스트림에서 파싱할 값이 저장된 필드 이름(기본값: line)

**overlay=BOOL**

원본 데이터의 출력 옵션(기본값: f).

* t: 파싱된 데이터를 필드에 출력하고, 원본 데이터를 **line** 필드에 출력
* f: 파싱된 데이터만 필드에 출력

#### 사용 예

루트 XML 요소에 속한 하위 XML 요소를 필드로 추출합니다.

* XML 요소가 문자열만 포함한다면 요소 태그를 필드의 이름으로 사용하고, 필드의 값으로 문자열을 할당합니다.
* XML 요소에 속성이 있으면 각 XML 속성 이름-값 쌍을 맵의 키-값 쌍으로, XML 요소의 문자열을 **\_text** 필드의 값으로 변환합니다.

예를 들어, <doc><id>sample</id></doc> 형태의 XML을 파싱하면 **id** 필드에 sample 문자열 값이 할당됩니다.

형태의 XML이라면 **name** 필드에는 {"locale":"ko","\_text":"로그프레소"}와 같이 locale=ko, \_text=로그프레소 이렇게 2개의 키-값 맵이 할당됩니다. [parsemap](https://docs.logpresso.com/ko/query/parse-command) 명령어를 조합하면 복합 객체 안에 있는 맵에서 쉽게 필드를 추출할 수 있습니다.

json| parsexml| parsemap field=name overlay=t

### pcapdecode

패캣을 디코딩해 L4 메타데이터를 출력합니다.

#### 문법

pcapdecode

#### 설명

이 명령어의 출력 필드는 다음과 같습니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 유형 | 설명 |
| src\_mac | 문자열 | 출발지 MAC 주소 |
| dst\_mac | 문자열 | 목적지 MAC 주소 |
| vlan\_id | 정수 | VLAN ID |
| protocol | 문자열 | esp, icmp, tcp, udp 중 하나 |
| src\_ip | IP 주소 | 출발지 IP 주소 |
| src\_port | 정수 | 출발지 포트 |
| dst\_ip | IP 주소 | 목적지 IP 주소 |
| dst\_port | 정수 | 목적지 포트 |
| payload | 바이너리 | 패킷 페이로드 |

#### 사용 예

# 다운로드: https://raw.githubusercontent.com/logpresso/dataset/main/pcap/nslookup.pcap| pcapfile /opt/logpresso/nslookup.pcap | pcapdecode

### pcapreplay

지정한 PCAP 송수신 장치를 통해 PCAP 파일로 저장된 통신을 재현합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다. 인입된 트래픽을 IPS나 네트워크 통신 분석 장비의 모니터 포트로 전송하는 방식으로 응용해 사용할 수 있습니다.

#### 문법

pcapreplay device="DEVICE\_NAME" [pps=INT]

필수 매개변수

**device="DEVICE\_NAME"**

[system pcapdevices](https://docs.logpresso.com/ko/query/system-pcapdevices-command) 명령으로 식별된 디바이스 중에서 통신 패킷을 재현할 네트워크 인터페이스. 인터페이스를 지정하려면 **name**으로 식별되는 디바이스의 이름을 지정합니다.

선택 매개변수

**pps=INT**

패킷 전송 속도 PPS(packets per second) 단위로 지정

#### 설명

이 명령어가 동작하려면 로그프레소를 운영하는 시스템에 libpcap이나 winpcap과 같은 드라이버가 설치되어 있고, 로그프레소 프로세스가 네트워크 인터페이스에 대해 RAW I/O를 관리자 권한으로 사용할 수 있어야 합니다.

이 명령어는 들어오는 트래픽을 IPS나 네트워크 트래픽 분석 장비의 모니터 포트로 전송하는 방식으로 적용할 수 있습니다.

테이블에 저장된 패킷 데이터를 재현하려면 [table](https://docs.logpresso.com/ko/query/table-command) 명령에 order=asc 옵션을 적용해 패킷을 원래의 시간 순서대로 정렬해야 합니다.

#### 사용 예

최근 5분간 tapped\_traffic 테이블에 저장된 레코드에서 **payload** 필드를 읽어온 후 PCAP 송수신 장치 enp0s3에서 1,302,083 pps (약 1Gbps에 해당) 속도로 트래픽을 전송

table order=asc duration=5m tapped\_traffic | fields payload | pcapreplay device="enp0s3" pps=1302083

### pivot

집계 함수를 실행해 피봇을 실행한 결과를 출력합니다.

#### 문법

pivot [parallel=BOOL] AGGR\_FUNC [as ALIAS], ... [by|rows GRP\_FIELD, ...] [for|cols GRP\_FIELD, ...]

필수 매개변수

**AGGR\_FUNC [as ALIAS], ...**

[집계 함수](https://docs.logpresso.com/ko/query/section-aggregate-functions)(AGGR\_FUNC) 및 필드 이름으로 사용할 별칭(ALIAS)으로 구성된 쌍의 목록. 쉼표(,)를 구분자로 사용합니다. ALIAS는 필수가 아니지만 지정하는 것이 좋습니다. 별칭을 지정하지 않으면 count(), sum(sent\_pkts)과 같은 함수 이름을 필드 이름으로 사용합니다.

선택 매개변수

**parallel=BOOL**

쿼리 결과를 병렬로 출력 사용 여부(기본값: f)

* t: 쿼리 결과를 병렬로 출력. 처리 속도는 증가하지만 데이터의 순서를 보장하지 않습니다. 데이터의 순서가 중요한 쿼리문에서는 이 옵션을 사용하지 마십시오.
* f: 쿼리 결과를 병렬로 출력하지 않음

**by|rows FIELD, ...**

by 또는 rows 지시자와 함께 필드를 지정하면 필드의 값을 단위로 하여 집계 함수를 적용합니다.

**for|cols FIELD, ...**

by 또는 rows 지시자로 지정한 필드에 대해 for 또는 cols 지시자로 지정된 필드 값을 단위로 하여 집계 함수를 적용합니다.

'by|rows' 절이나 'for|cols' 절이 지정되지 않으면 이전 쿼리 명령어에서 넘어오는 전체 로그를 하나의 그룹으로 계산합니다. 그룹 필드를 기준으로 정렬되는 부수적인 효과가 있습니다.

#### 사용 예

[count()](https://docs.logpresso.com/ko/query/count-aggregate-function) 함수를 호출해 전체 행 개수를 계산

pivot count

[count()](https://docs.logpresso.com/ko/query/count-aggregate-function) 함수를 호출해 **src\_ip** 필드 값의 고유 항목 당 개수를 계산

pivot count by src\_ip

[count()](https://docs.logpresso.com/ko/query/count-aggregate-function) 함수를 호출하여 **src\_ip**와 **dst\_ip** 필드에 대해 **protocol** 필드 값(예: TCP, UDP, ICMP)별로 행 개수를 계산

pivot count by src\_ip, dst\_ip for protocol

**src\_ip**와 **dst\_ip** 필드에 대해 **protocol** 필드 값(예: TCP, UDP, ICMP)별로 행 개수([count()](https://docs.logpresso.com/ko/query/count-aggregate-function))와 트래픽 용량([sum(bytes)](https://docs.logpresso.com/ko/query/sum-aggregate-function))을 계산

pivot sum(bytes) as bytes, count rows src\_ip, dst\_ip cols protocol

### prev

입력 데이터 스트림에서 지정된 필드(예: **count**)를 조회하고, 다음 입력 레코드에 직전 레코드의 필드 값을 접두사 \*\*prev\_\*\*가 붙은 필드(예: **prev\_count**)에 추가합니다. 이 명령어는 주로 데이터의 변화량을 추출하는 용도로 사용됩니다.

#### 문법

prev INPUT\_FIELD, ...

필수 매개변수

**INPUT\_FIELD, ...**

이전 값을 추적할 필드. 하나 이상 필드를 지정하려면 구분자로 쉼표(,)를 사용합니다. 지정된 필드에 대해 이전 값을 저장하는 필드(접두사로 prev\_가 추가된 필드)에 이전 레코드의 값을 저장합니다.

#### 사용 예

1분 단위로 GC 횟수의 변화량 계산

table sys\_gc\_logs | timechart span=1m count | prev count | eval delta = count - prev\_count

GC 발생 간격이 10초 이내인 GC 로그 조회

table order=asc sys\_gc\_logs | prev \_time | eval interval = datediff(prev\_\_time, \_time, "sec") | search interval < 10

### rename

원본 필드 이름을 as 절에서 지정한 필드 이름으로 변경합니다.

#### 문법

rename FIELD as NEW\_NAME[, FIELD as NEW\_NAME, ...]

필수 매개변수

**FIELD**

원본 필드의 이름

**as NEW\_NAME**

새로 사용할 필드 이름을 지정하는 as 절

#### 사용 예

src\_ip 필드 이름을 Source로 변경

rename src\_ip as Source

### repeat

지정한 숫자만큼 결과를 반복합니다. 이 명령은 반복 순서를 보장하지 않습니다. 각 행별로 반복되는 경우도 있고 일정 단위의 집합별로 반복되는 경우도 있습니다.

#### 문법

repeat count=INT

필수 매개변수

**count=INT**

결과를 반복할 횟수

#### 사용 예

최근 CPU 사용률 10건을 3번씩 표시

table limit=10 sys\_cpu\_logs | repeat count=3

임의의 데이터 100건 생성

json "{}" | repeat count=100 | eval seq=seq() | eval rand\_value=rand(100)

### rex

지정된 필드에서 정규표현식을 이용하여 필드를 추출합니다.

#### 문법

rex field={FIELD|line} "REGEX"

필수 매개변수

**field=FIELD**

정규 표현식을 이용하여 문자열을 추출할 대상 필드.

**"REGEX"**

필드 이름을 부여할 수 있도록 확장된 정규표현식. 정규표현식 그룹을 만들 때 (?<field>.\*) 형식으로 지정하면 그룹에 매칭된 문자열을 **field** 필드에 출력합니다.

#### 사용 예

**line** 필드에서 GET /game/flash/ 또는 POST /game/flash/으로 시작하는 파일 경로를 검색해 매칭된 문자열을 **filename** 필드에 출력

rex field=line "(GET|POST) /game/flash/(?<filename>([^ ]\*))"

**line** 필드에서 타임스탬프 패턴의 문자열을 추출해 **timestamp** 필드에 출력

rex field=line "(?<timestamp>\d+-\d+-\d+ \d+:\d+:\d+)"

**line** 필드에서 문자열을 추출해 **url**과 **querystring** 필드에 출력

rex field=line "(GET|POST) (?<url>[^ ]\*) (?<querystring>[^ ]\*) "

### rollup

그룹별 부분 집계와 전체 집계를 계산합니다. by 절에 여러 필드가 있을 경우, 필드 순서대로 부분 집계와 전체 집계를 표시합니다.

#### 문법

rollup [label=VALUE] AGGR\_FUNC [as ALIAS], ... [by GRP\_FIELD, ...]

필수 매개변수

**AGGR\_FUNC [as ALIAS], ...**

[집계 함수](https://docs.logpresso.com/ko/query/section-aggregate-functions)(AGGR\_FUNC) 및 필드 이름으로 사용할 별칭(ALIAS)으로 구성된 쌍의 목록. 쉼표(,)를 구분자로 사용합니다. 별칭(ALIAS)은 필수가 아니지만 지정하는 것이 좋습니다. 별칭을 지정하지 않으면 count(), sum(sent\_pkts)과 같은 함수 이름을 필드 이름으로 사용합니다.

선택 매개변수

**label=VALUE**

그룹핑 필드에 할당할 값(기본값: null).

**by GRP\_FIELD, ...**

by 지시자와 함께 그룹핑 필드를 지정하면 필드의 값을 단위로 하여 집계 함수를 적용합니다.

#### 사용 예

**action** 필드별 개수 및 전체 개수

rollup count by action

**action**, **status** 그룹핑 필드에 대해 항목별 개수와 **size** 필드 항목별 합계 계산 (라벨은 "TOTAL"로 표시)

rollup label=TOTAL count, sum(size) as size by action, status

#### 호환성

rollup 명령어는 ENT #1804 2017-11-28\_13-31 버전부터 지원합니다.

### search

지정된 표현식과 일치하는 입력 데이터를 검색합니다.

#### 문법

search [limit=INT] EXPR

필수 매개변수

**EXPR**

검색 조건 표현식. 예를 들어, "KEY == VALUE" 또는 "KEY != VALUE" 형태의 비교 조건식, 또는 불리언 조건식을 입력할 수 있습니다. and, or 같은 논리 연산자를 이용해 조건식을 연결할 수 있습니다.

EXPR이 참일 때에만 데이터를 다음 쿼리 명령어로 전달합니다.

선택 매개변수

**limit=INT**

반환할 최대 레코드 개수(기본값: 제한 없음)

#### 사용 예

**line** 필드에 game 문자열을 포함하는 로그 (와일드카드 지원)

search line == "\*game\*"

상태 코드가 200이 아닌 로그

search status != 200

src\_ip가 1.2.3.4이고 dst\_port가 22인 경우

search src\_ip == ip("1.2.3.4") and dst\_port == 22

### serial

순서가 중요한 명령문을 실행할 수 있도록 입력을 튜플 단위로 직렬화하여 실행하고, 서브쿼리 결과를 합쳐서 전달합니다.

#### 문법

serial [ SUBQUERY ]

**[ SUBQUERY ]**

스트림을 처리할 수 있는 명령어로 구성된 서브쿼리문을 대괄호 쌍([ ])으로 감싸서 입력합니다.

#### 사용 예

테이블 쿼리 후 행 단위로 CEP 함수([evtctxgetvar()](https://docs.logpresso.com/ko/query/evtctxgetvar-function), [evtctxsetvar()](https://docs.logpresso.com/ko/query/evtctxsetvar-function)) 적용

table iis | # CEP 함수 쿼리문을 직렬화 실행 | serial [ search cs\_uri\_stem == "\*game\*" | evtctxadd topic=TEST key=cs\_uri\_stem maxrows=0 true | eval prev\_ip = evtctxgetvar("TEST", cs\_uri\_stem, "prev\_ip") | eval \_dummy = evtctxsetvar("TEST", cs\_uri\_stem, "prev\_ip", c\_ip)]| fields \_time, cs\_method, prev\_ip, c\_ip, cs\_uri\_stem, cs\_uri\_query

### signature

**line** 필드로부터 특수문자의 집합으로 구성된 시그니처를 추출합니다. 보통 파서를 개발하기 전에 패턴 유형별 로그 샘플을 추출하기 위한 용도로 사용합니다.

#### 문법

signature

#### 사용 예

각 시그니처별 첫번째 샘플 로그 추출

signature | stats first(line) by signature

### sort

지정한 필드를 기준으로 입력 데이터를 정렬합니다.

#### 문법

sort [limit=INT] [-]FIELD, ... [by PARTITION\_FIELD, ...]

필수 매개변수

**[-]FIELD, ...**

출력할 순서로 정렬한 필드 목록. 필드 구분자로 쉼표(,)를 사용합니다. 필드의 기본 차순은 오름차순입니다. 내림차순으로 정렬하려면 필드 이름 앞에 '–'를 붙입니다.

선택 매개변수

**limit=INT**

정렬된 검색 결과에서 추출할 레코드 개수(기본값)

**by PRTITION\_FIELD, ...**

파티션 필드 값을 기준으로 파티셔닝 후 파티션별 정렬을 수행합니다. limit 옵션과 by 문을 같이 사용할 경우, 각 파티션마다 N개를 추출합니다.

#### 사용 예

**count** 필드를 기준으로 상위 10개 내림차순 추출

sort limit=10 -count

**bytes**와 **pkts** 필드 기준으로 상위 10개 내림차순 추출

sort limt=10 -bytes, -pkts

각 **src**와 **dst** 필드에 대해 **bytes**와 **pkts** 기준으로 상위 10개 내림차순 추출

sort limt=10 -bytes, -pkts by src, dst

### stats

그룹을 대상으로 동작하는 집계 함수의 평가 결과를 출력합니다.

#### 문법

stats [parallel=BOOL] AGGR\_FUNC [as ALIAS], ... [by GRP\_FIELD, ...]

필수 매개변수

**AGGR\_FUNC [as ALIAS], ...**

실행할 집계 함수(AGGR\_FUNC)를 이용해 표현식을 입력합니다. as 절을 이용해 집계 함수값을 담을 필드에 이름(ALIAS)을 지정할 수 있습니다. 이름을 지정하지 않으면 **count()**, **sum(sent\_pkts)** 등 함수 이름을 필드 이름으로 사용하므로 필드 이름(ALIAS)을 지정하는 것이 좋습니다.

선택 매개변수

**parallel=BOOL**

쿼리 결과의 병렬 출력 여부(기본값: f). 처리 속도는 증가하지만 데이터의 순서를 보장하지 않습니다. 데이터의 순서가 중요한 쿼리문에서는 이 옵션을 사용하지 마십시오.

* t: 쿼리 결과를 병렬로 출력
* f: 쿼리 결과를 병렬로 출력하지 않음

**by GRP\_FIELD, ...**

by 절을 이용해 집계 함수 결과를 그룹화할 필드를 지정합니다. 그룹화 필드를 지정하지 않으면 이전 쿼리 명령어에서 넘어오는 전체 로그를 하나의 그룹으로 계산합니다. 그룹 필드를 기준으로 정렬되는 부수적인 효과가 있습니다.

#### 사용 예

전체 레코드 개수

stats count

**src\_ip** 필드 값별로 레코드 개수 계산하기

stats count by src\_ip

**src\_ip**, **dst\_ip** 필드 쌍으로 그룹화하여 레코드 개수 계산하기

stats count by src\_ip, dst\_ip

**src\_ip**, **dst\_ip** 필드 쌍으로 그룹화하여 [sum(bytes)](https://docs.logpresso.com/ko/query/sum-aggregate-function)와 레코드 개수([count](https://docs.logpresso.com/ko/query/count-aggregate-function)) 계산하기

stats sum(bytes) as bytes, count by src\_ip, dst\_ip

### timechart

지정된 시간 단위마다 집계 함수의 결과를 계산합니다. by 절을 이용하여 그룹 필드를 지정하는 경우, 그룹 필드 값으로 필드가 생성되면서 필드별 통계 값을 계산합니다.

#### 문법

timechart span=INT{y|mon|w|d|h|m|s} AGGR\_FUNC [as ALIAS], ... [by GRP\_FIELD, ...]

필수 매개변수

**span=INT{y|mon|w|d|h|m|s}**

**\_time** 필드의 시간 단위. y(연), mon(월), w(주), d(일), h(시), m(분), s(초) 단위로 지정할 수 있습니다. 가령, 10m은 10분 단위입니다. 월 단위 mon을 사용하는 경우, 집계가 가능하도록 12의 약수 중에서 1mon, 2mon, 3mon, 4mon, 6mon만 지정할 수 있습니다. 즉, 3mon은 허용되지만 5mon은 허용되지 않습니다. 12mon 대신에 1y를 사용합니다. 단위가 y일 때, 1y만 허용됩니다.

**AGGR\_FUNC [as ALIAS], ...**

실행할 집계 함수(AGGR\_FUNC)를 이용해 표현식을 입력합니다. as 절을 이용해 집계 함수값을 담을 필드에 이름(ALIAS)을 지정할 수 있습니다. 이름을 지정하지 않으면 **count()**, **sum(sent\_pkts)** 등 함수 이름을 필드 이름으로 사용하므로 필드 이름(ALIAS)을 지정하는 것이 좋습니다.

선택 매개변수

**by GRP\_FIELD, ...**

집계에 사용할 그룹 필드 목록. 구분자로 쉼표(,)를 사용합니다.

#### 사용 예

10분 단위 로그 발생량

timechart span=10m count

1분 단위 bytes 변화 추이

timechart span=1m sum(bytes)

각 목적지 포트의 1시간 단위 로그 발생 갯수

timechart span=1h count by dst\_port

### tojson

주어진 필드 값들을 JSON 형식 문자열로 변환합니다.

#### 문법

tojson [output=TARGET\_FIELD] [FIELD, ...]

선택 매개변수

**output=TARGET\_FIELD**

JSON 형식으로 변환한 문자열을 저장할 필드(기본값: \_json)

**FIELD, ...**

JSON 형식으로 변환할 원본 필드를 지정합니다(기본값: 모든 필드)

#### 사용 예

전체 필드를 json 문자열로 변환해 **result** 필드에 할당

tojson output=result

**\_time**, **line** 필드를 json 문자열로 변환해 **jsonlog** 필드에 할당

tojson output=jsonlog \_time, line

## 데이터 매핑

### geocode\_kr

대한민국 행정구역 코드표를 조회합니다. 행정구역 명칭 필드과 함께 제공하는 코드 필드는 대시보드의 대한민국 지도 위젯의 지역과 매핑이 가능합니다.

#### 문법

geocode\_kr

### lookup

룩업 테이블을 조회하여 특정한 필드 값을 다른 값으로 변환합니다.

#### 문법

lookup LOOKUP\_TABLE KEY\_FIELD output MAP\_FIELD [as ALIAS], ...

필수 매개변수

**LOOKUP\_TABLE**

필드 값 변환에 사용할 룩업 테이블. 로그프레소는 country (ISO 2자리 국가 코드), region (지역), city (도시), latitude (위도), longitude (경도) 필드로 구성된 geoip 룩업 테이블을 내장하고 있습니다. 이 테이블을 이용해 IP 주소 타입 혹은 문자열인 입력 필드 값을 매핑 필드의 값으로 변환할 수 있습니다.

**KEY\_FIELD**

룩업 테이블에서 키로 동작하는 필드 이름

**output MAP\_FIELD [as ALIAS], ...**

룩업 테이블에서 키 값과 일치하는 레코드를 검색하고, 해당 레코드에서 지정된 매핑 필드(MAP\_FIELD)의 값을 출력 필드(ALIAS)에 할당합니다. as 절을 이용해 출력 필드(ALIAS)를 지정할 수 있습니다. 생략하면 매핑 필드(MAP\_FIELD) 이름이 그대로 사용됩니다.

로그프레소 셸에서 logdb.loadCsvLookup 명령으로 미리 매핑 테이블을 적재하거나, geoip처럼 lookup 쿼리 명령어를 지원하는 확장 모듈을 설치할 수 있습니다.

#### 사용 예

geoip를 이용해 IP 주소를 위치 정보로 변환하기

lookup geoip src\_ip output country

lookup geoip src\_ip output region

lookup geoip src\_ip output city

lookup geoip src\_ip output latitude, longitude

### lookuptable

룩업 테이블의 내용을 조회합니다. 웹 콘솔에서 데이터베이스 기반으로 생성한 룩업용 매핑 테이블, geoip 매핑 테이블, [memlookup](https://docs.logpresso.com/ko/query/memlookup-command) 명령어를 이용한 룩업에 대한 내용은 조회가 되지 않습니다.

웹 콘솔 메뉴 **쿼리 > 룩업**에서 룩업 테이블을 추가할 수 있습니다.

#### 문법

lookuptable LOOKUP\_TABLE [limit=INT] [offset=INT] [FIELD\_LIST]

필수 매개변수

**LOOKUP\_TABLE**

조회할 룩업 테이블

선택 매개변수

**limit=INT**

가져올 최대 레코드 개수(기본값: 제한 없음)

**offset=INT**

건너뛸 레코드 개수(기본값: 0)

**FIELD\_LIST**

조회할 필드 목록. 구분자로 쉼표(,)를 사용합니다.

#### 사용 예

룩업 테이블 country\_code의 모든 필드 보기

lookuptable country\_code

룩업테이블 country\_code에서 **code** 필드 30건만 보기

lookuptable country\_code limit=30 code

룩업테이블 **country\_code**에서 **country**, **population** 필드 내용 보기

lookuptable country\_code country, population

### memlookup

인메모리(in-memory) 룩업 테이블을 생성, 삭제하거나, 룩업 테이블의 모든 데이터를 조회합니다.

#### 문법

(파이프로 전달받은 데이터를 이용해서) 인메모리 룩업 테이블을 생성하려면

memlookup op=build name=TABLE key=KEY\_FIELD FIELD, ...

인메모리 룩업 테이블을 삭제하려면

memlookup op=drop name=TABLE

인메모리 룩업 테이블의 메타데이터를 조회하거나, 특정한 인메모리 룩업 테이블의 전체 레코드를 조회하려면

memlookup [op=list] [name=TABLE]

필수 매개변수

**op={build|drop|list}**

수행할 작업(operation)(기본값: list).

* build: 쿼리문이 완료될 때까지 입력으로 전달받은 데이터를 이용해 룩업 테이블을 생성(build)합니다.
* drop:, name 옵션으로 지정한 룩업 테이블을 삭제합니다.
* list: name 옵션으로 지정한 룩업 테이블을 조회합니다. memlookup으로 생성되지 않은 룩업 테이블이라면, 쿼리가 실패합니다. memlookup 명령어에 아무런 옵션을 주지 않고 실행하면 op=list 옵션만 지정해 실행하는 것과 같습니다

**name=TABLE**

op 옵션으로 지정된 작업을 수행할 대상 테이블을 지정합니다. op=list일 때, 인메모리 룩업 테이블을 지정하지 않으면 모든 인메모리 룩업 테이블의 목록을 보여줍니다. 이 때 보여주는 정보는 다음과 같습니다: **name**(룩업 이름), **key**(키 필드 이름), **size**(룩업 테이블의 레코드 개수)

**key=KEY\_FIELD**

op=build일 때 사용하는 옵션으로, 키 필드를 지정합니다.

FIELD, ...op=build일 때, 테이블을 구성할 필드 목록을 지정합니다. 구분자로 쉼표(,)를 사용합니다.

#### 사용 예

쿼리를 통한 매핑 테이블 만들기

status, desc1, desc2 컬럼을 가지고 있는 CSV 파일에서 status 컬럼을 키로 하고 desc1, desc2 컬럼을 output으로 하는 http\_status 인메모리 룩업 테이블을 생성합니다.

csvfile http\_status.csv | memlookup op=build name=http\_status key=status desc1, desc2

룩업 테이블 목록 조회

memlookup으로 생성된 룩업 테이블 정보를 확인할 수 있습니다. 반환되는 정보는 룩업 테이블 이름과 키 컬럼, 그리고 전체 레코드 개수입니다.

memlookup

위 명령문은 다음 명령문과 결과가 동일합니다.

memlookup op=list

특정한 룩업 테이블 전체 레코드 조회

인메모리 룩업 테이블의 이름을 지정하면 해당 테이블의 모든 레코드를 조회할 수 있습니다.

memlookup name=http\_status

위 명령문은 다음 명령문과 결과가 동일합니다.

memlookup op=list name=http\_status

인메모리 룩업 테이블 삭제

연산자 (op) 옵션 값으로 drop을 부여하여 지정된 룩업 테이블을 삭제합니다. 룩업의 이름을 명시하지 않은 경우 에러가 발생합니다.

memlookup op=drop name=http\_status

### nslookup

도메인 필드로 지정된 값을 읽어와 도메인 주소 질의를 수행하고 결과를 보여줍니다.

#### 문법

nslookup ns=IP\_ADDR [OPTIONS] DOMAIN\_FIELD output FIELD, ...

필수 매개변수

**ns=IP\_ADDR**

DNS 서버의 IPv4 또는 IPv6 주소

**DOMAIN\_FIELD**

도메인 네임 필드

**output FIELD\_LIST**

DNS 응답에서 조회할 필드를 쉼표(,)로 구분하여 지정. 다음 중에서 선택해서 입력

* additionals: 그 밖에 다른 정보들
* answers: DNS 서버의 응답 결과
* authorities: 도메인 레코드를 관리하는 DNS 서버(authoritative server) 정보
* flags: 요청/응답 메시지의 필드 컨트롤 플래그 값(출럭 필드에 다음과 같은 내용이 표시됨)
* AA: 도메인 주소에 대한 권한이 있는 서버
* TC: 메시지가 길어서 잘림
* RD: 재귀적 질의가 요청됨
* RA: 재귀적 질의 가능
* ip: IPv4 또는 IPv6 주소
* status: 쿼리 실행 결과(오류가 있으면 오류 메시지 표시)
* FORMAT\_ERROR: 질의 내용에 오류가 있어 서버가 처리할 수 없음
* NAME\_ERROR: 질의한 도메인 주소가 존재하지 않음
* NO\_ERROR: 오류 없음(0)
* NOT\_IMPLEMENTED: DNS 서버가 요청
* REFUSED: DNS 서버의 응답 거부
* SERVER\_FAILURE: 서버 오류로 질의를 처리할 수 없음

선택 매개변수

**cache=INT**

DNS 응답 캐시 크기(기본값: 1048576, 약 1 MB)

**timeout=INT**

DNS 응답 대기 시간(기본값: 5, 단위: 초)

**type=TYPE**

DNS 서버에 질의할 DNS 레코드 유형(기본값: A). 다음 중 하나를 지정:

* A: 질의할 도메인 주소에 연결된 IPv4 주소
* AAAA: 질의할 도메인 주소에 연결된 IPv6 주소
* CNAME: 캐노니컬 네임. 다른 도메인 이름의 별칭 주소.
* MX: 지정한 도메인 네임으로 들어오는 메일을 받는 서버
* NS: 도메인 주소를 관리하는 DNS 서버 정보
* PTR: 역 DNS 검색에 쓰이는 캐노니컬 네임 포인터 주소
* TXT: 사람이나 기계가 읽을 수 있는 텍스트 정보

#### 사용 예

spamhouse 테이블 **domain** 필드에 기록된 도메인 주소들에 대해 정보를 조회

table spamhouse| nslookup timeout=5 ns="1.1.1.1" domain output ip, status, flags, answers, authorities, additionals

#### 호환성

nslookup 명령어는 ENT #2309 2019-11-27\_10-43 버전부터 지원합니다.

## 데이터 적재

### drop

들어오는 모든 입력을 버립니다.

#### 문법

drop

#### 설명

셸에서 스크립트 파일 형태로 부수적인 효과가 있는 쿼리 명령어를 실행하고 출력 결과는 버릴 때, 배치 실행만 하고 쿼리 결과가 필요 없을 때, 이전 명령어의 쿼리 수행 시간만을 측정하려고 할 때 사용합니다.

### import

입력되는 모든 레코드를 지정된 테이블에 기록합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

import TABLE

매개변수

**TABLE**

데이터를 저장할 테이블의 이름

선택 매개변수

**create=t**

t로 설정하면, 데이터를 저장할 테이블의 유무를 확인합니다(기본값: f). 테이블이 없으면 자동으로 테이블을 생성합니다. 이 옵션을 사용하지 않거나, f로 설정하면 테이블이 존재하지 않으면 쿼리가 실패합니다.

#### 사용 예

기존 전체 sys\_cpu\_logs 데이터를 old\_sys\_cpu\_logs에 입력

table order=asc sys\_cpu\_logs | import create=t old\_sys\_cpu\_logs

### insert

입력된 필드 값을 기준으로 테이블을 선택하여 데이터를 입력합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

insert table=FIELD

매개변수

**table=FIELD**

지정된 필드 이름을 로그에서 찾아 해당 값을 테이블 이름으로 하여 데이터를 기록합니다. 데이터가 기록될 때 table 옵션에 지정된 필드는 제외하여 기록되며 해당 필드가 없는 로그는 기록되지 않습니다.

### outputcsv

특정 필드의 값을 CSV 또는 TSV 파일로 내보냅니다.

#### 문법

outputcsv [OPTIONS] FILE\_PATH FIELD, ...

필수 매개변수

**FILE\_PATH**

CSV/TSV 파일로 내보낼 경로

**FIELD, ...**

CSV 또는 TSV 파일에 출력할 필드 목록. 필드 구분자로 쉼표(,)를 사용합니다.

CSV와 TSV 형식은 모든 행의 필드 구성이 동일해야 하므로 로그프레소의 outputcsv 명령을 사용할 때는 반드시 출력필드를 정의해야 합니다. 출력필드의 순서와 무관하게 데이터를 내보내려면 outputjson 명령을 사용하십시오.

선택 매개변수

**append=BOOL**

FILE\_PATH에 이미 파일이 있을 때 이어서 쓰기 제어 옵션(기본값: f)

* t: FILE\_PATH로 지정한 파일의 끝에 데이터를 이어서 작성하고, 파일이 없으면 새 파일을 생생합니다. overwrite=t일 때, 함께 사용할 수 없습니다.
* f: 이어서 쓰기 비활성화. 이미 파일이 있으면 쿼리가 실패합니다.

**bom=BOOL**

파일 헤더에 BOM(Byte Order Mark)를 사용 여부

* t: BOM을 파일 헤더에 추가
* f: BOM을 파일 헤더에 추가하지 않음

**encoding=CHARSET**

문자열 인코딩 형식(기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>.

**flush=INT{y|mon|w|d|h|m|s}**

출력 버퍼를 지우는 주기. 주기 단위는 y(연), mon(월), w(주), d(일), h(시), m(분), s(초) 중 하나를 사용할 수 있습니다. 예를 들어, 5초마다 버퍼를 비우려면 5s로 지정합니다.

**overwrite=BOOL**

FILE\_PATH에 이미 파일이 있을 때 덮어쓰기 제어 옵션(기본값: f)

* t: 파일 덮어쓰기 활성화. append=t일 때, 함께 사용할 수 없습니다.
* f: 파일 덮어쓰기 비활성화. 이미 파일이 있으면 쿼리가 실패합니다.

**partition=BOOL**

FILE\_PATH의 매크로 기능 제어 옵션(기본값: f). partition=t일 때, FILE\_PATH에 매크로를 이용해 시간에 따라 디렉터리 및 파일 경로를 변경할 수 있습니다.

* t: 매크로 활성화
* f: 매크로 비활성화

사용할 수 있는 매크로는 {logtime:FMT}과 {now:FMT}가 있습니다. 입력 예시는 사용 예 2번을 참조하십시오.

* {logtime:FMT}: 로그 발생 시각을 기준으로 디렉터리나 파일에 이름 부여
* {now:FMT}: 현재 시각을 기준으로 디렉터리나 파일에 이름 부여

파티션 옵션을 지정하고 경로에 매크로를 사용하지 않으면 쿼리가 실패합니다.

**tab=BOOL**

필드 구분자로 탭(tab) 문자 사용 여부(기본값: f)

* t: 탭(tab) 문자를 구분자로 사용. TSV(Tab-separated Values) 파일을 처리할 때 유용합니다.
* f: 구분자로 쉼표(,)를 사용

**tmp=TMP\_FILE\_PATH**

임시 파일 경로. 이 옵션을 설정하면 임시 파일을 만들어 해당 경로에 출력한 다음, 쿼리문 정상적으로 종료되는 시점에 FILE\_PATH로 지정한 파일 경로로 이동시킵니다.

#### 사용 예

ippair.csv 파일로 **src\_ip**와 **dst\_ip** 필드의 값을 기록

outputcsv /opt/logpresso/files/ippair.csv src\_ip, dst\_ip

매크로를 이용해 로그 발생 연월일로 디렉터리를 지정하고, 현재 시각 기준 파일 이름을 생성하는 방식으로 **src\_ip**와 **dst\_ip** 필드 값을 기록

outputcsv partition=t /opt/logpresso/files/{logtime:/yyyy/MM/dd/}/{now:HHmm}.csv src\_ip, dst\_ip

### outputjson

특정 필드의 값을 JSON 형식으로 내보냅니다. 각 JSON 레코드는 개행으로 구분됩니다.

#### 문법

outputjson [OPTIONS] FILE\_PATH [FIELD, ...]

필수 매개변수

**FILE\_PATH**

CSV/TSV 파일로 내보낼 경로

선택 매개변수

**append=BOOL**

FILE\_PATH에 이미 파일이 있을 때 이어서 쓰기 제어 옵션(기본값: f)

* t: FILE\_PATH로 지정한 파일의 끝에 데이터를 이어서 작성하고, 파일이 없으면 새 파일을 생생합니다. overwrite=t일 때, 함께 사용할 수 없습니다.
* f: 이어서 쓰기 비활성화. 이미 파일이 있으면 쿼리가 실패합니다.

**bom=BOOL**

파일 헤더에 BOM(Byte Order Mark)를 사용 여부

* t: BOM을 파일 헤더에 추가
* f: BOM을 파일 헤더에 추가하지 않음

**encoding=CHARSET**

문자열 인코딩 형식(기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>.

**flush=INT{y|mon|w|d|h|m|s}**

출력 버퍼를 지우는 주기. 주기 단위는 y(연), mon(월), w(주), d(일), h(시), m(분), s(초) 중 하나를 사용할 수 있습니다. 예를 들어, 5초마다 버퍼를 비우려면 5s로 지정합니다.

**overwrite=BOOL**

FILE\_PATH에 이미 파일이 있을 때 덮어쓰기 제어 옵션(기본값: f)

* t: 파일 덮어쓰기 활성화. append=t일 때, 함께 사용할 수 없습니다.
* f: 파일 덮어쓰기 비활성화. 이미 파일이 있으면 쿼리가 실패합니다.

**partition=BOOL**

FILE\_PATH의 매크로 기능 제어 옵션(기본값: f). partition=t일 때, FILE\_PATH에 매크로를 이용해 시간에 따라 디렉터리 및 파일 경로를 변경할 수 있습니다.

* t: 매크로 활성화
* f: 매크로 비활성화

사용할 수 있는 매크로는 {logtime:FMT}과 {now:FMT}가 있습니다. 입력 예시는 사용 예 3번을 참조하십시오.

* {logtime:FMT}: 로그 발생 시각을 기준으로 디렉터리나 파일에 이름 부여
* {now:FMT}: 현재 시각을 기준으로 디렉터리나 파일에 이름 부여

파티션 옵션을 지정하고 경로에 매크로를 사용하지 않으면 쿼리가 실패합니다.

**tmp=TMP\_FILE\_PATH**

임시 파일 경로. 이 옵션을 설정하면 임시 파일을 만들어 해당 경로에 출력한 다음, 쿼리문 정상적으로 종료되는 시점에 FILE\_PATH로 지정한 파일 경로로 이동시킵니다.

**FIELD, ...**

JSON 파일에 출력할 필드 목록. 필드 구분자로 쉼표(,)를 사용합니다. 필드를 선택하지 않으면 모든 필드 값을 JSON 파일에 기록합니다.

#### 사용 예

output.json 파일로 모든 필드를 기록

outputjson /opt/logpresso/files/output.json

ippair.json 파일에 src\_ip와 dst\_ip를 기록

outputjson /opt/logpresso/files/ippair.json src\_ip, dst\_ip

기록 매크로를 이용해 로그 발생 연월일로 디렉터리를 지정하고, 현재 시각 기준 파일 이름을 생성하는 방식으로 src\_ip와 dst\_ip 필드 값을 기록

outputjson partition=t /opt/logpresso/files/{logtime:/yyyy/MM/dd/}/{now:HHmm}.json src\_ip, dst\_ip

### outputpcap

입력으로 받은 **payload** 필드를 지정한 파일 시스템 경로에 PCAP 파일로 기록합니다.

#### 문법

outputpcap FILE\_PATH

필수 매개변수

**FILE\_PATH**

파일을 저장할 경로

#### 사용 예

pcap\_stream 로그 수집기에서 발생하는 스트림을 5분간 모니터링하면서 목적지나 출발지 포트가 80인 패킷만 저장

logger window=5m localhost\pcap\_stream| pcapdecode| search src\_port==80 or dst\_port==80| outputpcap /opt/logpresso/files/http.pcap

### outputtxt

지정된 파일시스템 경로에 주어진 필드 값들을 텍스트 파일로 기록합니다.

#### 문법

outputtxt [append=BOOL] [delimiter=CHAR] [encoding=CHARSET] [flush=INT{y|mon|w|d|h|m|s}] [gz=BOOL] [partition=BOOL] [tmp=TMP\_FILE\_PATH] FILE\_PATH FIELD, ...

필수 매개변수

**FILE\_PATH**

파일로 내보낼 경로

**FIELD, ...**

파일에 출력할 필드 목록. 필드 구분자로 쉼표(,)를 사용합니다.

선택 매개변수

**append=BOOL**

FILE\_PATH에 이미 파일이 있을 때 이어서 쓰기 제어 옵션(기본값: f)

* t: FILE\_PATH로 지정한 파일의 끝에 데이터를 이어서 작성하고, 파일이 없으면 새 파일을 생생합니다. overwrite=t일 때, 함께 사용할 수 없습니다.
* f: 이어서 쓰기 비활성화. 이미 파일이 있으면 쿼리가 실패합니다.

**bom=BOOL**

파일 헤더에 BOM(Byte Order Mark)를 사용 여부

* t: BOM을 파일 헤더에 추가
* f: BOM을 파일 헤더에 추가하지 않음

**delimiter="CHAR"**

필드 구분자로 사용할 문자(기본값: 공백문자).

**encoding=CHARSET**

문자열 인코딩 형식(기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>.

**flush=INT{y|mon|w|d|h|m|s}**

출력 버퍼를 지우는 주기. 주기 단위는 y(연), mon(월), w(주), d(일), h(시), m(분), s(초) 중 하나를 사용할 수 있습니다. 예를 들어, 5초마다 버퍼를 비우려면 5s로 지정합니다.

**gz=t**

GZIP 압축 사용 여부(기본값: f)

* t: 텍스트 파일을 gz 아카이브 파일로 압축
* f: GZ 압축 기능 사용 안 함

**overwrite=BOOL**

FILE\_PATH에 이미 파일이 있을 때 덮어쓰기 제어 옵션(기본값: f)

* t: 파일 덮어쓰기 활성화. append=t일 때, 함께 사용할 수 없습니다.
* f: 파일 덮어쓰기 비활성화. 이미 파일이 있으면 쿼리가 실패합니다.

**partition=BOOL**

FILE\_PATH의 매크로 기능 제어 옵션(기본값: f). partition=t일 때, FILE\_PATH에 매크로를 이용해 시간에 따라 디렉터리 및 파일 경로를 변경할 수 있습니다.

* t: 매크로 활성화
* f: 매크로 비활성화

사용할 수 있는 매크로는 {logtime:FMT}과 {now:FMT}가 있습니다. 입력 예시는 사용 예 2번을 참조하십시오.

* {logtime:FMT}: 로그 발생 시각을 기준으로 디렉터리나 파일에 이름 부여
* {now:FMT}: 현재 시각을 기준으로 디렉터리나 파일에 이름 부여

파티션 옵션을 지정하고 경로에 매크로를 사용하지 않으면 쿼리가 실패합니다.

**tmp=TMP\_FILE\_PATH**

임시 파일 경로. 이 옵션을 설정하면 임시 파일을 만들어 해당 경로에 출력한 다음, 쿼리문 정상적으로 종료되는 시점에 FILE\_PATH로 지정한 파일 경로로 이동시킵니다.

#### 사용 예

ippair.txt 파일에 src\_ip와 dst\_ip 필드의 값을 기록

outputtxt /opt/logpresso/files/ippair.txt src\_ip, dst\_ip

기록 매크로를 이용해 로그 발생 연월일로 디렉터리를 지정하고, 현재 시각 기준 파일 이름을 생성하는 방식으로 src\_ip와 dst\_ip 필드 값을 기록

outputtxt partition=t /home/Logpresso/{logtime:/yyyy/MM/dd/}-{now:HHmm}.txt src\_ip, dst\_ip

### sendmail

입력되는 레코드를 메일로 전달합니다. sendmail 명령어는 입력이 들어오는 즉시 메일 큐에 적재하고, 메일을 비동기적으로 발송합니다. 이 명령어를 실행하려면 관리자 권한이 필요하고, 시스템 설정 메뉴에서 메일 서버가 설정되어 있어야 합니다.

#### 문법

sendmail [html=BOOL]

선택 매개변수

**html=BOOL**

**message** 필드에 있는 데이터를 html 형식으로 파싱 여부(기본값: f)

* t: **message** 필드 데이터를 html로 파싱
* f: **message** 필드 데이터를 평문 텍스트로 파싱

#### 설명

sendmail 명령어는 다음과 같은 필드를 입력으로 받습니다.

**to**

쉼표(,)로 구분된 메일 주소 목록

**subject**

메일 제목

**message**

메일 본문 텍스트

입력 필드가 누락되었거나, 주소 목록이 잘못된 경우 **\_sendmail\_fail** 필드에 오류 메시지를 기록합니다. 단, SMTP 설정 누락으로 인한 오류는 표시되지 않습니다.

로그프레소 셸에서 logpresso.mailQueue 명령으로 현재 대기 중인 발송 메일 목록을 확인할 수 있으며, logpresso.clearMailQueue 명령으로 일괄 삭제할 수 있습니다.

#### 사용 예

**to**, su**b**ject, **message** 필드를 생성해 sendmail 명령어로 전달해 메일 전송

json "{}"| eval to="example\_1@example.com, example\_2@example.com", subject="Hello World", message="<h1>Hello, World</h1>"| sendmail html=t

#### 호환성

html 옵션은 ENT-3.10.2004.0 버전부터 사용 가능합니다.

### sendsyslog

#### 문법

sendsyslog [OPTIONS] dst=IP\_ADDR

필수 매개변수

**dst=IP\_ADDR**

Syslog 서버의 IP 주소

선택 매개변수

**format=json|txt**

전송할 로그의 형식을 json, txt 중에서 선택해 지정(기본값: txt).

* json: 입력으로 받는 데이터를 모두 JSON 형식으로 변환해 전송
* txt: **line** 필드의 문자열 값을 그대로 전송

**port=INT**

Syslog 서버의 리스닝 포트 번호(기본값: 514). 1 ~ 65535 사이의 정수를 입력할 수 있습니다.

**pri=NUM**

RFC 5424 (https://tools.ietf.org/html/rfc5424)에 정의된 PRI 상수값(기본값: 134). 이는 local0와 INFO 수준에 해당됩니다.

PRI 상수값은 Facility 값의 8배수에 Severity 값을 더한 값입니다. 다음 표를 참조해서 계산된 값을 사용하십시오.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Facility(↓)<br />Severity(→) | 0/Emer | 1/Alert | 2/Crit | 3/Error | 4/Warn | 5/Notice | 6/Info | 7/Debug |
| 0 / kern | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 / user | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 2 / mail | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 3 / deamon | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 4 / auth | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 5 / syslog | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 6 / lpr | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 7 / news | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 8 / uucp | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 |
| 9 / clock | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| 10 / authpriv | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 |
| 11 / ftp | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| 12 / ntp | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| 13 / audit | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| 14 / alert | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 |
| 15 / solaris-cron | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| 16 / local0 | 128 | 129 | 130 | 131 | 132 | 133 | **134** (기본값) | 135 |
| 17 / local1 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| 18 / local2 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 |
| 19 / local3 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| 20 / local4 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 |
| 21 / local5 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| 22 / local6 | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 |
| 23 / local7 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |

**src=IP\_ADDR**

출발지 IP 주소를 로그프레소 서버의 IP 주소가 아닌 임의의 IP 주소로 바꿔서 전송하려는 경우 설정

출발지 IP 주소를 변경해 전송하려면 운영체제에 libpcap이나 winpcap 라이브러리가 설치되어 있어야 합니다. 또한 araqne-pcap 라이브러리를 운영체제에 맞게 다시 컴파일해야 할 수 있습니다. 출발지 IP 주소를 지정하는 경우에만 PCAP을 사용하여 주소를 변경한 패킷을 생성하여 전송합니다. MTU를 넘는 패킷 크기는 전송에 실패하므로 주의해야 합니다.

## 데이터 병합

### join

입력으로 받는 데이터의 필드와 서브 쿼리 결과 필드를 비교해 결합(join)합니다.

join은 스트림 쿼리를 지원하지 않습니다. 스트림 쿼리에 조인을 적용하려면 streamjoin 명령어를 사용하세요.

#### 문법

join [type={cross|full|inner|left|leftonly|right|rightonly}] KEY\_FIELD, ... [ SUBQUERY ]

필수 매개변수

**KEY\_FIELD, ...**

조인의 기준이 키 필드. 필드 구분자로 쉼표(,)를 사용합니다. type=cross일 때, 예외적으로 키 필드 목록을 사용하지 않습니다.

**[ SUBQUERY ]**

입력 데이터와 조인할 데이터를 출력하는 서브쿼리문을 대괄호 쌍([ ])으로 감싸서 입력

선택 매개변수

**type={cross|full|inner|left|leftonly|right|rightonly}**

조인 유형을 지정합니다(기본값: inner).

* cross: 'Catersian product'라고도 합니다. 입력 데이터의 집합(M개 레코드)과 서브쿼리 결과 집합(N개)을 결합해 M x N 개의 레코드로 구성된 집합을 출력합니다. cross 조인을 사용할 때, KEY\_FIELD를 지정하지 않아야 합니다.
* full: 키가 일치하는 레코드는 병합해서, 키가 일치하지 않는 레코드는 있는 그대로 출력합니다. 데이터의 합집합과 유사합니다.
* 키가 일치하면, 입력 데이터 레코드에 서브쿼리 필드를 결합해 출력합니다.
* 키가 일치하지 않으면, 입력 데이터와 서브쿼리 데이터를 각각 그대로 출력합니다.
* inner: 키가 일치하는 레코드만 결합해서 출력합니다. 키를 포함하지 않는 레코드는 출력하지 않습니다. 데이터의 교집합과 유사합니다. 일반적으로 조인이라 하면 "inner 조인"을 의미합니다.
* left: 키가 일치하는 레코드는 결합해서 출력하고, 키가 일치하지 않는 레코드는 입력 데이터의 레코드만 출력합니다.
* leftonly: 입력 레코드 집합 중에서 서브쿼리 결과 집합과 키가 일치하지 않는 레코드만 출력합니다. 키가 일치하는 레코드는 출력하지 않습니다.
* right: 키가 일치하는 레코드는 결합해서 출력하고, 키가 일치하지 않는 레코드는 서브쿼리 결과만 출력합니다.
* rightonly: 서브쿼리 결과 집합 중에서 입력 레코드 집합과 키가 일치하지 않는 레코드만 출력합니다. 키가 일치하는 레코드는 출력하지 않습니다.

#### 사용 예

**code** 필드를 키로 하여 inner 조인

# code 필드를 갖는 json을 입력 데이터로 전달 | json "[{'code':1}, {'code':2}, {'code':3}]" | # code, name 필드를 갖는 json을 반환하는 서브쿼리문 | # 입력 데이터와 서브쿼리문의 결과 데이터를 code 필드를 키로 하여 inner join | join code [ json "[ {'code':1, 'name':'foo'}, {'code':2, 'name':'bar'} ]" ]

**쿼리 실행 결과 (inner join)**

|  |  |
| --- | --- |
| code | name |
| 1 | foo |
| 2 | bar |

서브 쿼리에서 조회되는 결과를 제외하고 출력(leftonly 조인)

json "[ {'field1': 'A'}, {'field1': 'B'}, {'field1': 'C'}, {'field1': 'D'}]" | join type=leftonly field1 [ json "[ {'field1': 'A', 'field2': 'Foo'}, {'field1': 'D', 'field2': 'Bar'} ]" ]

**쿼리 실행 결과 (leftonly join)**

|  |
| --- |
| field1 |
| B |
| C |

위에서 실행한 쿼리문은 다음 쿼리문과 실행 결과가 동일합니다(left 조인의 응용).

json "[ {'field1': 'A'}, {'field1': 'B'}, {'field1': 'C'}, {'field1': 'D'}]" | join type=left field1 [ json "[ {'field1': 'A', 'field2': 'Foo'}, {'field1': 'D', 'field2': 'Bar'} ]" ] | search isnull(field2)

부서별 통계 출력, 통계 값이 존재하지 않더라도 모든 부서를 출력(right 조인)

json "[{'id': 1, '건수': 1000}, {'id':2, '건수': 2000}]" | join type=right id [ json "[ {'id':1, '부서':'영업'}, {'id':2, '부서':'운영'}, {'id':3, '부서':'기술'} ]" ]

**쿼리 실행 결과 (right join)**

|  |  |  |
| --- | --- | --- |
| id | 부서 | 건수 |
| 1 | 영업 | 1000 |
| 2 | 운영 | 2000 |
| 3 | 기술 |  |

문서보안과 매체제어 위반 로그를 계정 기준으로 조합하고, 일치하지 않는 경우 각 로그 출력 (full 조인)

json "[ {'계정':'bob', '문서보안위반': 1}, {'계정':'alice', '문서보안위반': 5} ]" | join type=full 계정 [ json "[ {'계정':'alice', '매체제어위반': 8}, {'계정':'clark', '매체제어위반': 3} ]" ]

**쿼리 실행 결과 (full join)**

|  |  |  |
| --- | --- | --- |
| 계정 | 문서보안위반 | 매체제어위반 |
| bob | 1 |  |
| alice | 5 | 8 |
| clark |  | 3 |

#### 호환성

type 중에서 cross, rightonly는 ENT-3.0.2003.0 버전부터 사용 가능합니다.

### streamjoin

입력으로 받는 스트림 데이터의 필드와 서브 쿼리 결과 필드를 비교해 조인(join)합니다.

#### 문법

streamjoin [OPTIONS] KEY\_FIELD, ... [ SUBQUERY ]

#### 필수 매개변수

**KEY\_FIELD, ...**

조인의 기준이 키 필드. 필드 구분자로 쉼표(,)를 사용합니다.

**[ SUBQUERY ]**

입력 데이터와 조인할 데이터를 출력하는 서브쿼리문을 대괄호 쌍([ ])으로 감싸서 입력

선택 매개변수

**timeout=INT{s}**

서브쿼리가 완료될 때까지 대기할 시간(기본값: 무한정 대기)

**type=inner|left|leftonly**

조인 유형(기본값: inner).

* inner: 키가 일치하는 레코드만 결합해서 출력합니다. 키를 포함하지 않는 레코드는 출력하지 않습니다. 데이터의 교집합과 유사합니다. 일반적으로 조인이라 하면 "inner 조인"을 의미합니다.
* left: 키가 일치하는 레코드는 결합해서 출력하고, 키가 일치하지 않는 레코드는 입력 데이터의 레코드만 출력합니다.
* leftonly: 입력 레코드 집합 중에서 서브쿼리 결과 집합과 키가 일치하지 않는 레코드만 출력합니다. 키가 일치하는 레코드는 출력하지 않습니다.

#### 설명

streamjoin 명령어는 서브쿼리의 결과를 오프힙 메모리에 적재하고 해시 조인을 수행하므로 [join](https://docs.logpresso.com/ko/query/join-command) 명령어보다 속도가 빠르며, 스트림 쿼리에서도 사용할 수 있습니다. 하지만, inner와 left, leftonly 조인만 할 수 있고, 처리할 수 있는 데이터의 크기는 메모리 풀 용량에 따라 제한됩니다. 서브쿼리가 실패하면 **\_streamjoin\_fail** 필드에 예외 메시지를 추가합니다.

로그프레소를 실행할 때 아래 옵션을 지정하여 메모리 풀 크기를 조절할 수 있습니다. 기본값은 '500M'입니다. 다음 예시와 같이 메모리 풀 크기를 지정할 수 있습니다:-Dlogpresso.streamjoin.max\_buffer\_size=1G

메모리 사용 현황을 아래의 쿼리로 조회할 수 있습니다:

* 메모리 풀 사용 현황: system memory pools
* 쿼리별 메모리 사용 현황: system memory objects

#### 사용 예

데이터베이스에서 가져온 데이터와 **code** 필드를 키로 하여 조인

json "[ {'code':1}, {'code':2}, {'code':3} ]" | streamjoin code [ dbquery ora select code, description from tbl\_codes ]

데이터베이스에서 가져온 데이터와 **code** 필드를 키로 하여 조인. 단, SQL 쿼리를 10초로 제한

json "[ {'code':1}, {'code':2}, {'code':3} ]" | streamjoin timeout=10s code [ dbquery ora select code, description from tbl\_codes ]

#### 호환성

type 중에서 leftonly는 ENT-3.0.2003.0 버전부터 사용 가능합니다.

### union

서브 쿼리의 결과를 병합합니다. union은 다른 쿼리와 병행하여 실행되므로 출력 순서를 보장하지 않습니다. 통계 처리를 수행하는 경우처럼, 순서가 중요하지 않으면서 높은 수행 성능이 필요할 때에 주로 사용합니다.

#### 문법

union [ SUBQUERY ]

필수 매개변수

**[ SUBQUERY ]**

입력 데이터와 조인할 데이터를 출력하는 서브쿼리를 대괄호 쌍([ ])으로 감싸서 입력

#### 사용 예

2개 DB의 SQL 쿼리 결과를 병합

dbquery db1 select \* from nodelist| union [ dbquery db2 select \* from nodelist ]

## 이벤트 연관 분석

### evtctxadd

입력 데이터가 조건식과 일치하면 지정된 키로 이벤트 컨텍스트를 생성합니다.

#### 문법

evtctxadd dynamic=t key=KEY\_FIELD CONDITIONAL\_EXPR

또는

evtctxadd [expire=INT{mon|d|h|m|s}] [maxrows=INT] [timeout=INT{mon|d|h|m|s}] topic=STR key=KEY\_FIELD CONDITIONAL\_EXPR

필수 매개변수

**dynamic=BOOL**

입력 레코드로부터 topic, expire, timeout, maxrows을 전달받아 사용하는 기능(기본값: f)

* t: 입력 레코드로부터 동적으로 topic, expire, timeout, maxrows을 받아서 사용. dynamic=t일 때, topic, expire, timeout, maxrows 옵션을 함께 사용할 수 없습니다.
* f: 사용 안 함

**topic=STR**

이벤트 컨텍스트의 이름(토픽). 토픽은 인-메모리 데이터베이스의 테이블의 이름과 같은 역할을 합니다. dynamic=t일 때, 이 옵션을 사용할 수 없습니다.

**key=KEY\_FIELD**

키 필드의 이름. 이벤트 컨텍스트를 구분하는 고유 키는 키 필드에 저장합니다.

**CONDITIONAL\_EXPR**

이벤트 컨텍스트를 생성할 조건이 되는 표현식

선택 매개변수

**expire=INT{mon|d|h|m|s}**

이벤트 컨텍스의 삭제 시점을 mon (월), d (일), h (시), m (분), s (초) 단위로 지정. 이벤트 컨텍스트가 생성된 후 지정한 시간이 지나면 컨텍스트를 삭제합니다. expire가 설정되면 이벤트 컨텍스트 조건식 CONDITIONAL\_EXPR에 일치하는 입력 데이터가 있어도 만료 시간은 연장되지 않습니다. dynamic=t일 때 이 옵션을 사용할 수 없습니다.

**maxrows=INT**

이벤트 컨텍스트에 저장할 레코드의 최대 개수(기본값: 10). dynamic=t일 때 이 옵션을 사용할 수 없습니다.

**timeout=INT{mon|d|h|m|s}**

이벤트 컨텍스트 조건식 CONDITIONAL\_EXPR에 일치하는 이벤트가 수신된 시점부터 지정된 시간이 지나면 이벤트 컨텍스트를 삭제합니다. mon (월), d (일), h (시), m (분), s (초) 단위로 지정할 수 있습니다. dynamic=t일 때 이 옵션을 사용할 수 없습니다.

#### 사용 예

**전문 전송 후 응답 수신이 10초 이상 경과하면 타임아웃 발생**

evtctxadd topic=txmatch key=txkey timeout=10s type == "send"| evtctxdel topic=txmatch key=txkey type == "recv"

예시로 든 쿼리문은 다음과 같은 이벤트 컨텍스트 생성/삭제 명령으로 구성되어 있습니다.

* evtctxadd: **type** 필드 값이 send인 경우 이벤트 컨텍스트 생성
* [evtctxdel](https://docs.logpresso.com/ko/query/evtctxdel-command): **type** 필드 값이 recv인 경우 이벤트 컨텍스트 삭제

두 명령문 모두 이벤트 컨텍스트 조건이 발생하면 토픽 주제인 txmatch와 이벤트 컨텍스트 키 필드인 **txkey**를 묶어서 이벤트 컨텍스트를 구분합니다.

이제 다음과 같은 이벤트 데이터가 입력으로 전달된다고 하면,

json "{'txkey':'001122', 'type':'send'}"json "{'txkey':'001122', 'type':'recv'}"

첫번째 데이터가 입력되면 이벤트 컨텍스트가 생성됩니다. 두 번째 데이터의 입력 시간에 따라 서로 다른 이벤트가 발생합니다.

* 10초 안에 입력되면 이벤트 컨텍스트 삭제 (EventCause.REMOVAL) 이벤트
* 10초가 지난 후 입력하거나, 입력하지 않으면 타임아웃 (EventCause.TIMEOUT) 이벤트

발생한 이벤트 컨텍스트의 삭제 원인에 따라 후속 명령으로 서로 다른 처리를 수행할 수 있습니다.

### evtctxdel

입력 데이터가 조건식과 일치하는 경우 주어진 키로 이벤트 컨텍스트를 제거합니다.

#### 문법

evtctxdel {dynamic=t|topic=STR} key=KEY\_FILED CONDITIONAL\_EXPR

필수 매개변수

**dynamic=BOOL**

입력 레코드로부터 topic을 전달받아 사용하는 기능(기본값: f)

* t: 입력 레코드로부터 동적으로 topic을 받아서 사용. 이 값을 설정하면 topic 옵션을 사용할 수 없습니다.
* f: 사용 안 함

**topic=STR**

이벤트 컨텍스트의 이름(토픽). 토픽은 인-메모리 데이터베이스의 테이블의 이름과 같은 역할을 합니다. dynamic=t일 때, 이 옵션을 사용할 수 없습니다.

**key=KEY\_FIELD**

이벤트 컨텍스트를 구분하는 키 값을 추출할 필드

**CONDITIONAL\_EXPR**

제거할 이벤트 컨텍스트의 표현식

### evtctxdrop

지정된 주제에 해당하는 모든 이벤트 컨텍스트를 일괄 삭제합니다.

#### 문법

evtctxdrop all=BOOL

또는

evtctxdrop topic="STR"

**all=t**

t로 설정하면 모든 이벤트 컨텍스트를 일괄적으로 삭제합니다. 이 옵션을 topic=STR과 함께 사용할 수 없습니다.

**topic=STR**

지정된 주제를 가진 모든 이벤트 컨텍스트를 삭제합니다. 이 옵션을 all=t와 함께 사용할 수 없습니다.

### evtctxlist

이벤트 컨텍스트 목록을 조회합니다.

#### 문법

evtctxlist [topic=STR]

선택 매개변수

**topic=STR**

주제와 일치하는 이벤트 컨텍스트. 옵션을 지정하지 않으면 전체 이벤트 컨텍스트 목록을 조회합니다.

## 머신 러닝

### anomalies

이상탐지 포레스트 모델(Isolation Forest, 일부 데이터를 샘플링하여 의사결정 나무 모델을 생성하는 방식)을 사용하여 이상치를 예측하는 데이터를 출력합니다.

#### 문법

사전에 준비된 학습 모델에 기반한 이상치 예측

anomalies [sample=INT] [size=INT] model=MODEL

서브쿼리를 통해 이상탐지 포레스트 분석을 실행

anomalies [sample=INT] [size=INT] FIELD, ... [ SUBQUERY ]

필수 매개변수

**FIELD, ...**

이상탐지 포레스트 분석에서 사용할 필드 목록. 필드 구분자로 쉼표(,)를 사용합니다.

**model=MODEL**

이상탐지 포레스트 분석 모델의 이름을 입력합니다. 로그프레소 엔진에 CLI로 접속하여 학습 모델을 생성하고 학습을 진행할 수 있습니다.

**[ SUBQUERY ]**

이상탐지 포레스트 분석에서 사용할 트레이닝셋 데이터 조회 쿼리를 입력합니다.

선택 매개변수

**sample=INT**

이상탐지 포레스트에서 학습에 사용할 샘플 갯수(기본값: 샘플 개수의 제곱근).

**size=INT**

이상탐지 포레스트를 구성하는 트리 갯수(기본값: 100)

#### 설명

이상치 값은 **\_score** 필드에 출력되고, 0 ~ 1 사이의 값으로 표시됩니다.

* 값이 1에 가까울수록 이상치일 확률이 높습니다.
* 0.5보다 훨씬 작은 값은 정상적인 관측값입니다.
* 모든 점수가 0.5에 가까우면 이상치가 없을 확률이 높습니다.

#### 사용 예

이름이 anomal\_stock인 학습 모델을 사용한 이상치 예측

# 다운로드: https://raw.githubusercontent.com/logpresso/dataset/main/stocks.csv | table stocks | anomalies model=anomal\_stock | eval anom = if(\_score>0.7, stocks, null)

서브쿼리로 트레이닝 셋을 사용하는 경우

table stocks | anomalies sample=256 stocks [ csvfile /test/sam\_train.csv | eval \_time=date(date, "yyyyMMdd"), stocks = int (stocks) | fields \_time, stocks ] | eval anom = if(\_score>0.65, stocks, null) | fields \_time, anom, stocks

### forecast

주어진 시계열 데이터에 대해 예측 데이터를 출력합니다.

#### 문법

forecast [OPTIONS] TIME\_SERIES\_FIELD [by GRP\_FIELD, ...]

필수 매개변수

**TIME\_SERIES\_FIELD**

시계열 데이터를 갖는 필드

#### 선택 매개변수

**count=INT**

출력할 데이터 행 개수(기본값: 5)

**period=INT**

데이터의 시계열 주기. 지정하지 않으면 고속 푸리에 변환(Fast Fourier Transform, FFT)을 이용해 자동으로 주기를 계산합니다.

**seed=INT**

고정 시드 값. 같은 입력에 같은 결과를 유지하고 싶을 때 시드 값을 지정합니다.

**time=FIELD**

시간 레코드로 사용할 필드(기본값: **\_time** 필드).

**by GRP\_FIELD, ...**

by 지시자와 함께 집계에 사용할 그룹핑 필드 목록. 구분자로 쉼표(,)를 사용합니다. 이 옵션은 TIME\_SERIES\_FIELD 뒤에 지정해야 합니다.

#### 설명

[timechart](https://docs.logpresso.com/ko/query/timechart-command) 명령어 등으로 시간 필드의 데이터의 간격이 일정하도록 설정한 후 forecast 명령어를 사용하는 것을 권장합니다. 입력 데이터는 파티션 별로 4건 이상 있어야 하며, period의 값은 입력 데이터 건 수의 1/2 이하이어야 합니다.

#### 사용 예

**count** 필드의 예측 데이터를 출력

forecast count

**traffic** 필드의 시계열 주기를 5로 잡고, **set\_time** 필드에 기재된 시간에 따라 시계열 예측 데이터를 출력. 같은 결과를 유지하기 위해 고정 시드 값으로 1234를 할당

forecast period=5 time=set\_time seed=1234 traffic

**region** 필드를 기준으로 **sent\_bytes** 필드의 시계열 예측 데이터를 10건씩 집계하여 추출

forecast count=10 sent\_bytes by region

### kmeans

유클리디안 거리를 기반으로 입력 레코드를 K개의 클러스터로 분류합니다.

#### 문법

kmeans [k=INT] [iter=INT] FIELD, ...

필수 매개변수

**FIELD, ...**

계산 대상 필드 목록. 쉼표(,)를 구분자로 사용합니다. 필드 값은 숫자형이어야 하며, 지정된 필드 값이 숫자가 아닌 입력 레코드는 무시됩니다. 최대 10만개의 입력 레코드를 허용하며, **\_cluster** 필드에 1부터 증가하는 번호로 클러스터를 분류하여 출력합니다. 유효 입력 레코드가 10만개를 넘으면 쿼리를 종료합니다.

선택 매개변수

**k=INT**

클러스터 수(기본값: 3)

**iter=INT**

계산 반복 횟수(기본값: 10000)

#### 사용 예

머신 러닝에서 흔히 인용되는 iris 데이터를 이용하여 시험할 수 있습니다. 길이와 너비를 이용하여 분류를 수행한 후, 실제 종(species) 이름과 비교해봅니다(다운로드: <https://github.com/illinois-cse/data-fa14/blob/gh-pages/data/iris.csv>).

csvfile /opt/logpresso/iris.csv| eval sepal\_length = double(sepal\_length), sepal\_width = double(sepal\_width)| kmeans k=4 iter=100000 sepal\_length, sepal\_width

### lof

가장 인접한 k개의 이웃을 기준으로 각 점의 밀도를 계산하고, 인접한 이웃과 상대적인 밀도의 비율을 계산하여 LOF (Local Outlier Factor) 지수를 계산합니다.

#### 문법

lof [k=INT] FIELD, ... [by GRP\_FIELD, ...]

필수 매개변수

**FIELD, ...**

정수, 실수, 날짜와 같은 숫자로 구성된 값을 포함한 필드 목록. 구분자로 쉼표(,)를 사용합니다.

선택 매개변수

**eps=DOUBLE**

데이터 간 최소 거리 조정계수(기본값: 0.00001). 데이터 간 거리의 합을 나눈 값이 무한으로 발산하지 않도록 조정하기 위해 사용합니다.

**k=INT**

계산에 사용할 이웃 노드 개수(기본값: 10).

**[by GRP\_FIELD\_1, ...]**

LOF 지수 계산에 사용할 그룹핑 필드 목록. 구분자로 쉼표(,)를 사용합니다. 이 옵션은 FIELD, ... 뒤에 지정해야 합니다.

by 절을 사용해 그룹별 스코어링을 계산하려면 각 그룹의 레코드 수는 이웃 노드 수(k=INT으로 지정한 값)보다 많아야 합니다. 이웃 노드 수보다 그룹의 레코드 수가 적으면 모든 점이 하나의 군집으로 잡히기 때문에 LOF 지수(**\_lof** 필드)가 의도한대로 계산되지 않습니다.

#### 설명

각 레코드마다 \_lof 필드에 LOF 지수를 계산하며, 이 값은 다음과 같이 분류할 수 있습니다:

* 값이 1보다 클 때(LOF(k) > 1): 군집의 바깥쪽에 위치합니다. 1보다 클수록 이상치(anomaly)일 가능성이 높습니다.
* 값이 1의 근사값일 때(LOF(k) ≈ 1): 군집의 경계에 위치합니다.
* 값이 1보다 작을 때(LOF(k) < 1): 군집의 내부에 위치합니다.

#### 사용 예

**sepal\_length**와 **sepal\_width** 필드 값을 기준으로 이상치를 계산합니다(다운로드: <https://raw.githubusercontent.com/illinois-cse/data-fa14/gh-pages/data/iris.csv>).

wget url="https://raw.githubusercontent.com/illinois-cse/data-fa14/gh-pages/data/iris.csv" | eval line = split(line, "\n") | explode line | split sep="," sepal\_length,sepal\_width,petal\_length,petal\_width,species| eval sepal\_length = double(sepal\_length), sepal\_width = double(sepal\_width)| lof sepal\_length, sepal\_width| search \_lof > 2

### rforest

랜덤 포레스트 모델(Random Forest, 여러 개의 결정 트리를 임의로 학습하는 방식)을 사용하여 예측 데이터를 출력합니다.

#### 문법

사전에 준비된 학습 모델에 기반한 예측

rforest [size=INT] model=MODEL

서브쿼리를 통해 랜덤 포레스트 분석을 실행

rforest [size=INT] target=TARGET\_FILED FIELD, ... [ SUBQUERY ]

필수 매개변수

**FIELD, ...**

랜덤 포레스트 분석에서 사용할 필드들을 입력합니다.

**model=MODEL**

랜덤 포레스트 모델의 이름을 지정합니다. 로그프레소 엔진에 CLI로 접속하여 학습 모델을 생성하고 학습을 진행할 수 있습니다.

**target=TARGET\_FIELD**

랜덤 포레스트 분석에서 target 변수에 분류값으로 사용할 필드를 지정합니다.

**[ SUBQUERY ]**

랜덤 포레스트 분석에서 사용할 트레이닝셋 데이터 조회 쿼리를 입력합니다.

선택 매개변수

**size=INT**

포레스트를 구성하는 트리 갯수(기본값: 100)

#### 설명

이 명령어를 실행하면 **\_guess** 필드에 target 변수의 값을 추정하여 출력합니다.

#### 사용 예

이름이 rforest\_titanic인 랜덤 포레스트 모델을 사용한 예측

# 다운로드: https://raw.githubusercontent.com/logpresso/dataset/main/titanic/train.csv table titanic\_test | rforest model=rforest\_titanic | eval \_guess = if(\_guess=="0", "사망 ", "생존")

서브쿼리로 트레이닝 셋을 사용하는 경우

table titanic\_test | rforest target=Survived Pclass, Sex, Age, Fare, Embarked [ csvfile /test/train.csv | eval Age=double(Age), Fare=double(Fare), CanbinLetter=nvl(substr(Cabin, 0, 1), "--"), TicketType=if(isnull(long(Ticket)), substr (Ticket, 0, indexof(Ticket, " ")), "--") | rex field=Name ", (?<Title>[^.]+)" | eval Survived = if(Survived=="0", " 사망 ", "생존") ]

### stl

시계열 데이터를 추세(trend), 계절적 변동(seasonality), 잔차(error)로 분해합니다. stl 쿼리 결과는 파티션 필드(by 절로 지정한 필드) 별로 최대 1000건 출력되며, 파티션 필드를 지정하지 않은 경우 전체 결과는 1000건으로 제한됩니다.

stl 출력 제한 건수를 늘리고 싶으면 -Dlogpresso.stl.limit=N 부팅 옵션을 추가하여 원하는 값을 입력하면 됩니다.

#### 문법

stl [period=INT{y|mon|w|d|h|m|s}] NUMERIC\_FIELD [by FIELD]

**period=INT{y|mon|w|d|h|m|s}**

시계열 주기를 지정합니다. s(초), m(분), h(시), d(일), w(주), mon(월), y(연) 단위로 지정할 수 있습니다. 지정한 주기에 따라 시계열 데이터가 반복성을 갖는다고 가정하고 분석합니다. 시계열 주기를 지정하지 않으면 스펙트럼 분석을 통해 자동으로 계절변동 주기를 계산합니다.

**NUMERIC\_FIELD**

계산 대상 시계열 데이터를 지정합니다. 필드 값은 정수, 실수, 날짜처럼 숫자이어야 합니다.

**by FIELD**

by 절이 지정되면 필드 값 별로 그룹이 만들어집니다. 가령, by dst\_port가 지정되면 80, 443 등 dst\_port 필드의 값이 필드의 이름이 되고 시간 단위로 통계가 산출됩니다.

#### 설명

STL은 seasonal-trend decomposition procedure based on loess의 약어로, period가 생략된 경우, 스펙트럼 분석을 통해 자동으로 계절변동 주기를 계산합니다.

stl 명령어는 시계열 데이터를 분석하여 \_trend(추세), \_seasonal(계절적 변동), \_error(잔차) 필드를 출력합니다. 시계열 데이터가 주기성이 없으면(예를 들어 'period=0m') \_seasonal 필드를 출력하지 않습니다.

## 프로시저

### proc

사용자 정의 프로시저를 실행합니다.

#### 문법

proc PROC\_NAME(PARAMETER, ...)

필수 매개변수

**PROC\_NAME(PARAMETER, ...)**

프로시저에 정의된 매개변수 형식에 맞추어 인자를 넘겨주면, 인자가 쿼리 매개변수로 설정된 후 미리 정의된 쿼리가 실행됩니다. 프로시저에서 정의한 매개변수 형식에 맞추어 상수로 평가될 수 있는 표현식 인자로 전달할 수 있습니다. 프로시저의 소유자 혹은 권한을 부여받은 사용자가 프로시저의 소유자 권한으로 쿼리를 실행합니다.

#### 사용 예

웹 콘솔에서 최근 24시간 동안 N% 이상의 과부하 기록을 추출하는 쿼리문을 프로시저로 저장합니다. 프로시저의 이름은 cpu\_overload으로 합니다. 프로시저 쿼리문은 [$()](https://docs.logpresso.com/ko/query/$-function) 함수로 쿼리 매개변수를 참조하도록 작성할 수 있습니다.

table duration=1d sys\_cpu\_logs | search kernel + user >= $("threshold")

이제 쿼리문을 작성할 때 다음과 같이 프로시저를 호출할 수 있습니다:

proc cpu\_overload(90)

## 외부 시스템 연동

### dbcall

SQL 저장 프로시저를 호출하고 실행한 결과(결과 집합 및/또는 출력 매개변수)를 반환합니다.

#### 문법

dbcall PROFILE {SQL\_STATEMENT}

필수 매개변수

**PROFILE**

JDBC 연결에 사용할 프로파일을 지정합니다.

프로파일은 웹 콘솔에서 구성할 수 있습니다. ENT-3.10.2009.0. SNR-3.1.2008.0 배포 버전부터 JDBC 프로파일이 접속 프로파일에 통합되었습니다.

**{SQL\_STATEMENT}**

SQL 저장 프로시저를 호출하는 쿼리문을 입력합니다.

SQL 쿼리문에 콜론(:)으로 시작하는 입력 매개변수와 출력 매개변수를 정의할 수 있습니다.

* 입력 매개변수는 :name 형식이며, set 명령어로 정의한 쿼리 매개변수가 삽입됩니다.
* 출력 매개변수는 :name(type) 형식으로 정의합니다. 사용할 수 있는 출력 매개변수 타입은 'varchar', 'int', 'datetime'이 있습니다.

출력하는 방식은 다음과 같습니다.

* 출력 매개변수만 반환하면, 출력 매개변수로 구성된 튜플 1건을 출력합니다.
* 결과 집합과 출력 매개변수를 반환하면 결과 집합의 모든 튜플에 출력 매개변수 필드를 추가하여 반환합니다.
* 다수의 결과 집합을 반환하면 모든 결과 집합을 순차적으로 조회하여 출력합니다.

#### 사용 예

Microsoft SQL Server에서 특정 테이블의 컬럼 구성 조회

dbcall mssql {call msdb.dbo.sp\_columns("log\_shipping\_primaries")}

Microsoft SQL Server에서 id가 1000인 line 값을 사용자 정의 프로시저로 조회

set id = 1000 | dbcall mssql {call GetLine(:id, :line(varchar))}

### dbload

입력으로 받는 쿼리 결과를 SQL 쿼리문으로 변환해 외부 SQL 서버에 입력합니다. dboutput 명령어와 동일한 기능을 수행하고, rowretry 옵션의 기본값만 다릅니다.

#### 문법

dbload PROFILE [OPTIONS] table=TABLE FIELD, ...

필수 매개변수

**PROFILE**

JDBC 접속 프로파일. 프로파일은 웹 콘솔에서 구성할 수 있습니다.

ENT-3.10.2009.0. SNR-3.1.2008.0 배포 버전부터 JDBC 프로파일이 접속 프로파일에 통합되었습니다.

**table=TABLE**

데이터를 입력할 테이블 이름

**FIELD, ...**

데이터베이스에 입력할 필드 목록. 필드 구분자로 쉼표(,)를 사용합니다. 필드 이름 앞에 + 기호를 붙이면 키 컬럼으로 인식합니다.

필드 이름은 대상 테이블의 컬럼 이름과 일치해야 합니다. 필드와 컬럼의 이름이 일치하지 않으면, dbload 명령문 앞에 [rename](https://docs.logpresso.com/ko/query/rename-command) 명령문을 이용해 SQL 데이터베이스의 컬럼과 이름을 일치시키십시오.

선택 매개변수

**batchsize=INT**

데이터베이스 배치 트랜잭션에 적용할 처리 단위. 단위가 크면 한 번에 많이 커밋하므로 효율적이지만, 트랜잭션이 실패했을 때 롤백할 레코드도 증가합니다. 권장하는 값은 2000입니다. 처리 단위를 지정하지 않으면 1건 단위로 커밋하므로 느릴 수 있습니다.

**database=SCHEMA**

접속 후 사용할 데이터베이스(또는 스키마)

**rowretry=BOOL**

행 단위로 재시도 여부(기본값: t). 설정하면 성능에서 손해를 볼 수 있지만 데이터 손실을 최소화할 수 있습니다.

* t: 배치 트랜잭션이 실패했을 때 행 단위로 트랜잭션 수행
* f: 배치 트랜잭션이 실패했을 때 행 단위 트랜잭션을 하지 않음

**stoponfail=BOOL**

실패한 쿼리 명령이 있을 때 트랜잭션의 중지 여부(기본값: f).

* t: 쿼리가 실패했을 때 트랜잭션을 중지
* f: 실패한 트랜잭션을 건너뛰고 다음 트랜잭션을 실행

**type=update**

실행할 SQL 쿼리의 유형을 insert, update 중에서 지정(기본값: insert).

update로 설정하면 FIELD에 적어도 1개 이상의 필드를 키 컬럼으로 지정해야 합니다. SQL 데이터베이스에서 키 컬럼이 있는지 확인하고(SQL SELECT 쿼리문 수행), 키 컬럼이 없으면 INSERT 명령을, 키 컬럼이 있으면 UPDATE 명령을 수행합니다.

#### 호환성

dbload 명령어는 ENT #2309 2019-11-27\_10-43 버전부터 사용할 수 있습니다.

### dblookup

입력 레코드를 SQL 쿼리문의 플레이스홀더에 대응시켜서 실행하고, 조회되는 첫번째 레코드를 필드로 할당합니다.

#### 문법

dblookup PROFILE [bypass=BOOLEAN\_EXPR] SQL\_SYNATAX

필수 매개변수

**PROFILE**

JDBC 연결에 사용할 프로파일을 지정합니다.

프로파일은 웹 콘솔에서 구성할 수 있습니다. ENT-3.10.2009.0. SNR-3.1.2008.0 배포 버전부터 JDBC 프로파일이 접속 프로파일에 통합되었습니다.\* (ENT, STD) 시스템 설정 > 접속 프로파일\* (SNR) 시스템 > 접속 프로파일

**SQL\_STATEMENT**

실행할 SQL 쿼리문을 입력합니다. SQL 쿼리문에 콜론(:)으로 시작하는 입력 매개변수를 정의할 수 있습니다.

* 입력 매개변수는 :name 형식이며, 입력 레코드의 필드 값이 대입됩니다.

선택 매개변수

**bypass=BOOLEAN\_EXPR**

SQL 쿼리문을 실행하지 않을 조건을 불리언 표현식으로 정의합니다. 조건이 참이면 SQL 쿼리문을 실행하지 않고 출력을 내보냅니다.

bypass=BOOLEAN\_EXPR은 일반적으로 조건절에 들어갈 필드 값이 존재하지 않으면 SQL을 실행하지 않도록 조건식을 구성합니다.

#### 사용 예

login 값으로 사용자 성명(name)과 성별(sex)을 조회하여 필드 확장

json "{'login':'logpresso'}"| dblookup USERDB bypass="isnull(login)" select name, sex from users where login = :login

### dboutput

입력으로 받는 쿼리 결과를 SQL 쿼리문으로 변환해 외부 SQL 서버에 입력합니다. [dbload](https://docs.logpresso.com/ko/query/dbload-command) 명령어와 동일한 기능을 수행하고, rowretry 옵션의 기본값만 다릅니다.

#### 문법

dboutput PROFILE [OPTIONS] table=TABLE FIELD, ...

필수 매개변수

**PROFILE**

JDBC 접속 프로파일. 프로파일은 웹 콘솔에서 구성할 수 있습니다.

ENT-3.10.2009.0. SNR-3.1.2008.0 배포 버전부터 JDBC 프로파일이 접속 프로파일에 통합되었습니다.

**table=TABLE**

데이터를 입력할 테이블 이름

**FIELD, ...**

데이터베이스에 입력할 필드 목록. 필드 구분자로 쉼표(,)를 사용합니다. 필드 이름 앞에 + 기호를 붙이면 키 컬럼으로 인식합니다.

필드 이름은 대상 테이블의 컬럼 이름과 일치해야 합니다. 필드와 컬럼의 이름이 일치하지 않으면, dbload 명령문 앞에 [rename](https://docs.logpresso.com/ko/query/rename-command) 명령문을 이용해 SQL 데이터베이스의 컬럼과 이름을 일치시키십시오.

선택 매개변수

**batchsize=INT**

데이터베이스 배치 트랜잭션에 적용할 처리 단위. 단위가 크면 한 번에 많이 커밋하므로 효율적이지만, 트랜잭션이 실패했을 때 롤백할 레코드도 증가합니다. 권장하는 값은 2000입니다. 처리 단위를 지정하지 않으면 1건 단위로 커밋하므로 느릴 수 있습니다.

**database=SCHEMA**

접속 후 사용할 데이터베이스(또는 스키마)

**rowretry=BOOL**

행 단위로 재시도 여부(기본값: f). 설정하면 성능에서 손해를 볼 수 있지만 데이터 손실을 최소화할 수 있습니다.

* t: 배치 트랜잭션이 실패했을 때 행 단위로 트랜잭션 수행
* f: 배치 트랜잭션이 실패했을 때 행 단위 트랜잭션을 하지 않음

**stoponfail=BOOL**

실패한 쿼리 명령이 있을 때 트랜잭션의 중지 여부(기본값: f).

* t: 쿼리가 실패했을 때 트랜잭션을 중지
* f: 실패한 트랜잭션을 건너뛰고 다음 트랜잭션을 실행

**type=update**

실행할 SQL 쿼리의 유형을 insert, update 중에서 지정(기본값: insert).

update로 설정하면 FIELD에 적어도 1개 이상의 필드를 키 컬럼으로 지정해야 합니다.

### dbquery

외부 데이터베이스 서버를 대상으로 SQL 질의를 수행합니다.

#### 문법

dbquery PROFILE SQL\_STATEMENT

**PROFILE**

JDBC 연결에 사용할 프로파일을 지정합니다.

프로파일은 웹 콘솔에서 구성할 수 있습니다. ENT-3.10.2009.0. SNR-3.1.2008.0 배포 버전부터 JDBC 프로파일이 접속 프로파일에 통합되었습니다.

**SQL\_STATEMENT**

질의할 SQL 쿼리문을 입력합니다. JDBC를 통해 조회되는 모든 결과 집합을 키-값 쌍으로 읽어들입니다.

SQL 쿼리문에 콜론(:)으로 시작하는 입력 매개변수와 출력 매개변수를 정의할 수 있습니다.

* 입력 매개변수는 :name 형식이며, [set](https://docs.logpresso.com/ko/query/set-command) 명령어로 정의한 쿼리 매개변수가 삽입됩니다.
* 출력 매개변수는 :name(type) 형식으로 정의합니다. 사용할 수 있는 출력 매개변수 타입은 varchar, int, datetime이 있습니다.

#### 사용 예

weblogs 테이블에서 100건의 로그 조회

dbquery oracle select \* from weblogs where rownum <= 100

입력 매개변수를 이용해 employee 테이블에서 최근 1주일 간 입사자 목록을 조회

# 입력 매개변수: created\_at | set created\_at = string(dateadd(now(), "day", -7), "yyyy-MM-dd") | dbquery emp select \* from employee where created\_at >= :created\_at

#### 성능

MS SQL Server 2008에서 쿼리 속도를 계측한 결과는 아래와 같습니다.

* 로그프레소 서버 하드웨어 사양: Intel Core i5-2467M 1.6GHz
* DB 서버 하드웨어 사양: Intel Core i5 750 2.67GHz, RAM 4GB
* 데이터: IIS 로그 1999194건
* 쿼리: dbquery sql select \* from logs
* 소요시간: 27.3초 (74,000건/초)

### dbscript

SQL 스크립트를 실행해 데이터를 읽어옵니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

dbscript PROFILE [cs=CHARSET] SQL\_FILE\_PATH [:parameter ...]

필수 개체

**PROFILE**

JDBC 연결에 사용할 접속 프로파일

프로파일은 웹 콘솔에서 구성할 수 있습니다. ENT-3.10.2009.0. SNR-3.1.2008.0 배포 버전부터 JDBC 프로파일이 접속 프로파일에 통합되었습니다.

**SQL\_FILE\_PATH**

실행할 SQL 스크립트 파일의 절대 경로와 스크립트에서 참조할 매개변수를 공백문자로 구분하여 입력합니다. SQL 스크립트 파일의 최대 길이는 1MB(1,048,576 bytes)를 초과할 수 없습니다.

SQL 스크립트 파일은 다음과 조건을 만족해야 합니다. - 'SELECT' 쿼리만 사용할 수 있습니다. - 물음표(?)를 사용하여 매개변수가 삽입될 위치를 지정할 수 있습니다.

선택 매개변수

**cs=CHARSET**

파일의 인코딩을 지정합니다(기본값: utf-8). 인코딩 형식 이름은 IANA Charset Registry에 등록된 Preferred MIME Name 또는 Aliases에 등록된 이름을 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**:parameter ...**

공백문자를 구분자로 하는 매개변수 목록. 매개변수의 이름은 콜론(:)으로 시작해야 하고, 매개변수의 순서대로 SQL 파일의 플레이스홀더를 대치합니다. 매개변수는 [set](https://docs.logpresso.com/ko/query/set-command) 명령어를 사용하여 설정하거나, [proc](https://docs.logpresso.com/ko/query/proc-command) 명령어의 호출 인자를 매개변수로 넘겨받을 수 있습니다. 프로시저의 호출은 [dbcall](https://docs.logpresso.com/ko/query/dbcall-command) 명령어를 참조하십시오.

### ftp

FTP 서버에서 파일 시스템을 탐색하거나, 쿼리 결과를 파일로 전송할 수 있습니다.

#### 문법

파일 시스템 목록 조회

ftp PROFILE ls [encoding=CHARSET] PATH

텍스트 기반 또는 JSON 형식 파일 읽기

ftp FTP\_PROFILE cat [encoding=CHARSET] [format=json] [limit=INT] [offset=INT] PATH

특정한 레코드의 필드 값을 텍스트, CSV, 또는 JSON 형식으로 파일 전송

ftp FTP\_PROFILE put [append=t] [overwrite=t] [encoding=CHARSET] [fields=FIELD,...] [format={json|csv}] PATH

하위 명령

**cat**

FTP 서버에서 PATH로 지정된 경로의 파일 내용을 읽어와 **line** 필드에 출력합니다. 출력할 수 있는 파일 형식은 텍스트, CSV, JSON입니다.

**ls**

FTP 서버에서 PATH로 지정된 경로의 파일 목록을 보여줍니다.

**put**

fields 옵션으로 지정된 필드의 값들을 PATH로 지정된 경로에 파일로 전송합니다.\*텍스트 형식일 때 필드는 탭으로 구분, 빈 필드는 '-'으로 표시

* CSV 형식일 때 첫번째 줄은 필드 이름 목록이며, 빈 필드는 빈 문자열로 표시

옵션

**append=BOOL**

쿼리 결과를 FTP 서버에 전송할 때 사용할 수 있는 옵션으로, PATH로 지정된 파일이 있으면 파일에 이어서 씁니다. overwrite=t와 함께 사용할 수 없습니다.

**overwrite=t**

쿼리 결과를 FTP 서버에 전송할 때 사용할 수 있는 옵션으로, PATH로 지정된 파일이 있으면 파일을 무시하고 덮어씁니다. append=t와 함께 사용할 수 없습니다.

**encoding=CHARSET**

파일 인코딩 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**fields=,FIELD,...**

FTP 서버에 전송할 필드 목록. 구분자로 쉼표(,)를 사용합니다.

* 텍스트 또는 CSV 파일로 전송할 때 이 옵션을 생략하면 **line** 필드를 출력합니다.
* JSON 파일로 전송할 때 이 옵션을 생략하면 모든 필드를 출력합니다.
* appent=t 옵션을 사용할 때 데이터의 일관성을 유지할 수 있도록 항상 fields 옵션의 목록 순서를 동일하게 유지해주세요.

**format={json|csv}**

조회/전송할 파일 형식. 텍스트 파일을 조회, 전송할 때에는 생략합니다.

* CSV 파일을 조회할 때에 이 옵션은 필요 없습니다.
* CSV 파일을 전송하려면 format=csv을 지정해야 합니다.
* JSON 파일을 조회하거나 전송할 때에는 format=json을 지정해야 합니다.

**limit=INT**

FTP 서버에서 파일을 읽어올 때 출력할 행의 개수(기본값: 무제한)

**offset=INT**

FTP 서버에서 파일을 읽을 때 건너뛸 행 개수(기본값: 0)

개체

**PROFILE**

FTP 접속 프로파일. 프로파일은 웹 콘솔에서 구성할 수 있습니다.

ENT-3.10.2009.0. SNR-3.1.2008.0 배포 버전부터 FTP 프로파일이 접속 프로파일에 통합되었습니다.- (ENT, STD) 시스템 설정 > 접속 프로파일- (SNR) 시스템 > 접속 프로파일

**PATH**

디렉터리 또는 파일의 절대 경로. 전송할 때에는 디렉터리 경로가 아니라 단일 파일 경로를 입력해야 합니다. 파일을 조회할 때에 파일 경로에 와일드카드(\*)를 사용하면 특정 문자열 패턴을 포함한 모든 파일을 한 번에 조회할 수 있습니다.

#### 사용 예

srv 프로파일 설정 후, 원격 디렉터리 조회

ftp srv ls /data/logs

출력 필드들은 각각 아래의 의미를 갖습니다:

* type(문자열): 디렉터리일 때 'dir', 파일일 때 'file'
* name(문자열): 파일 이름
* file\_size(정수): 파일 용량, 디렉터리일 때 '0'
* modified\_at(날짜): 마지막 수정 시각
* owner(문자열): 소유자
* group(문자열): 소유 그룹

logpresso.sh 파일의 첫 5행을 조회

ftp srv cat limit=5 /logpresso.sh

/sample.json 파일을 JSON 형식으로 파싱하여 조회

ftp srv cat format=json /sample.json

JMX 클래스 로딩 로그 중 LoadedClassCount, UnloadedClassCount만 /tmp/output.txt 파일에 출력

table classloading | ftp srv put fields=UnloadedClassCount,LoadedClassCount /tmp/output.txt

JMX 클래스 로딩 로그를 JSON 파일로 출력

table classloading | ftp srv put format=json /tmp/classloading.json

JMX 클래스 로딩 로그 중 LoadedClassCount, UnloadedClassCount만 /tmp/output.json 파일에 출력

table classloading | ftp srv put format=json fields=LoadedClassCount,UnloadedClassCount /tmp/classloading.json

### hdfs

HDFS를 탐색하거나, 쿼리 결과를 파일로 전송할 수 있습니다.

#### 문법

파일 목록 조회

hdfs PROFILE {ls|lsr} PATH

파일 내용 읽기

hdfs PROFILE cat [format=csv|json|sequence] [limit=INT] [offset=INT] PATH

특정한 레코드의 필드 값을 텍스트, CSV, JSON 형식으로 파일 전송

hdfs PROFILE put [format=csv|json] [limit=INT] [offset=INT] [partition=t] PATH

특정한 레코드의 필드 값을 HDFS 시퀀스 형식으로 파일 전송

hdfs PROFILE put format=sequence [compression\_type=block|record] [fields=FIELD,...] [key\_type=HDFS\_TYPE] [key\_field=KEY\_FIELD] [value\_type=HDFS\_TYPE] [value\_field=VALUE\_FIELD] PATH

필수 매개변수

**PROFILE**

HDFS 연결에 사용할 접속 프로파일

프로파일은 웹 콘솔에서 구성할 수 있습니다. ENT-3.10.2009.0. SNR-3.1.2008.0 배포 버전부터 HDFS 프로파일이 접속 프로파일에 통합되었습니다.

**{ls|lsr}**

ls 명령어는 PATH로 지정된 경로의 파일 목록을 보여줍니다. lsr은 PATH로 지정한 경로의 디렉터리 뿐만 아니라 하위의 모든 디렉터리를 탐색합니다.

**cat**

HDFS 파일 시스템에 있는 텍스트 파일, CSV 파일, JSON 파일, HDFS 시퀀스 파일 내용을 조회합니다. format 옵션으로 지정된 파일 형식에 맞춰 파싱합니다.

* 텍스트 기반 파일일 때, 파일 내용을 line 필드에 출력합니다.
* CSV 형식일 때, 첫 줄을 필드 행으로 인식합니다. 데이터에 개행 문자가 있더라도 CSV 규칙에 맞으면 여러 개의 줄을 하나의 필드 값으로 인식합니다.
* JSON 형식일 때, 파일을 행 단위로 읽어서 파싱합니다.
* HDFS 시퀀스 형식일 때, HDFS의 Writable 구현을 로그프레소 타입(Java의 데이터 타입)으로 변환하여 레코드 단위로 읽어옵니다. 변환 형식은 로그프레소의 HDFS 데이터 변환 타입을 참조하십시오.
* 키 필드 이름은 key로 지정됩니다. 키는 원본 타입에 관계없이 문자열로 변환합니다.
* 값 필드 이름은 값의 타입에 따라 다른 필드를 사용합니다.
* MapWritable 타입일 때: 내부의 키-값 매핑이 반환되는 행의 필드로 반환됩니다. 하둡에 내장된 Writable 구현을 로그프레소 타입으로 변환해 출력합니다.
* MapWritable 타입이 아닐 때: value 필드에 출력합니다.

**put**

fields 옵션으로 지정된 필드의 값들을 HDFS 파일 시스템에 파일로 전송합니다. 파일은 PATH로 지정된 경로에 생성됩니다.

* 텍스트 형식일 때 fields 옵션으로 지정한 필드 값을 전송합니다. 필드를 지정하지 않으면 line 필드의 값을 전송합니다.
* CSV 형식일 때 첫번째 줄은 fields 옵션을 필드 이름 목록을 기록합니다. 필드에 값이 없으면 빈 문자열로 표시합니다. fields 옵션을 지정하지 않으면 line 필드의 값을 전송합니다.
* JSON 형식일 때, fields 옵션으로 지정한 필드 값만 전송합니다. fields 옵션을 생략하면 모든 필드 값을 전송합니다.
* HDFS 시퀀스 형식일 때, 다음과 같은 경우가 아니라면 시퀀스 형식으로 전송합니다. 변환 형식은 로그프레소의 HDFS 데이터 변환 타입을 참조하십시오.
* 전송할 레코드의 키나 값 중에 하나라도 비어 있는 행은 전송하지 않습니다.
* 값과 타입이 일치하지 않는 경우,
* string 타입은 문자열로 변환합니다.
* int, long, float, double과 같은 숫자 타입은 0으로 변환합니다.
* bool 타입은 false로 변환합니다.
* 정밀도를 손상하지 않고 변환할 수 있으면 변환 후 출력합니다 (예: long 타입이 지정되었지만 int 값이 들어오면 long으로 변환 후 출력).

**PATH**

파일의 절대 경로

선택 매개변수

**compression\_type=block|record**

압축 형식. 이 옵션이 없으면 압축하지 않습니다. record는 레코드 단위 압축, block은 블록 단위 압축을 의미합니다.

**format=csv|json|sequence**

조회하거나 전송할 파일 형식. 텍스트 파일을 조회하거나 전송할 때에는 이 옵션을 생략합니다.

**key\_type=HDFS\_TYPE**

로그프레소의 HDFS 데이터 변환 타입에서 HDFS 타입에 정의된 타입을 지정합니다.

**key\_field=KEY\_FIELD**

키 필드 이름. 설정하지 않으면 1부터 시작하는 LongWritable 카운터를 사용합니다.

**limit=INT**

파일을 읽어올 때 출력할 행의 개수(기본값: 무제한).

**offset=INT**

파일을 읽어올 때 건너뛸 행 개수(기본값: 0).

**partition=t**

t로 설정하면 매크로를 이용해 디렉터리 경로를 지정할 수 있습니다. 매크로를 이용하면 디렉터리나 파일 이름을 시간에 따라 변경합니다.

**fields=FIELD,...**

put 명령으로 HDFS로 전송할 필드 목록. 구분자로 쉼표(,)를 사용합니다.

**value\_type=HDFS\_TYPE**

로그프레소의 HDFS 데이터 변환 타입에서 HDFS 타입에 정의된 타입을 지정합니다.

**value\_field=VALUE\_FIELD**

값 필드 이름. 설정하지 않으면 전체 필드를 하나의 MapWritable로 전송합니다.

#### 설명

cat으로 파일을 읽어들일 때 PATH에 와일드카드(\*)를 사용하면 특정 문자열 패턴을 포함한 모든 파일을 한 번에 조회할 수 있습니다.

put으로 파일을 전송할 때 partition 옵션을 t로 설정하면 매크로를 이용해 시간에 따라 디렉터리 및 파일 경로를 변경하도록 경로를 지정할 수 있습니다. 파티션 옵션을 지정하고 경로에 매크로를 사용하지 않으면 쿼리가 실패합니다.

사용할 수 있는 매크로는 {logtime:FMT}과 {now:FMT}가 있습니다.

매크로는 중괄호 쌍({ })으로 감싸 입력합니다. 입력 예시는 사용 예 7번을 참조하십시오.

로그프레소는 Java 표준 데이터 타입과 IP 주소와 같이 로그프레소에서 정의한 데이터 타입을 사용합니다. HDFS에서 데이터를 가져오거나 전송할 때 HDFS 데이터 타입에 맞춰 변환 작업을 수행합니다. 타입별 데이터 변환은 다음 표를 참조하십시오.

**로그프레소와 HDFS 데이터 변환 타입**

|  |  |  |
| --- | --- | --- |
| 로그프레소 타입 | HDFS 타입 | 설명 |
| string | Text | 문자열 |
| null | NullWritable | 널 (null) |
| bool | BooleanWritable | 불리언 (참/거짓) |
| int | IntWritable, VIntWritable | 4바이트(32비트) 정수 |
| long | LongWritable, VLongWritable | 8바이트(64비트) 정수 |
| float | FloatWritable | 단정도 실수(single precision) |
| double | DoubleWritable | 배정도 실수(double precision) |

#### 사용 예

vm 이름의 프로파일로 접속하여 루트 경로 파일 목록을 조회

hdfs vm ls /

출력하는 필드는 다음과 같습니다.

* type(문자열): 디렉터리인 경우 dir, 파일인 경우 file
* name (문자열): 파일 이름
* path (문자열): 파일의 절대 경로
* replication (정수): 복제본 수, 디렉터리인 경우 0
* file\_size (정수): 파일 크기, 디렉터리인 경우 0
* block\_size (정수): 블럭 크기, 디렉터리인 경우 0
* modified\_at (날짜): 마지막 수정 시각
* permission (문자열): 권한 설정
* owner (문자열): 소유자
* group (문자열): 소유그룹

vm 프로파일로 접속하여 /tmp/LICENSE.txt 파일의 첫 행을 건너뛰고 5개 행을 조회

hdfs vm cat offset=1 limit=5 /tmp/LICENSE.txt

vm 프로파일로 접속하여 /tmp/malware.csv 파일에서 3개 행을 조회

hdfs vm cat format=csv limit=3 /tmp/malware.csv

vm 프로파일로 접속하여 /tmp/iis.json 파일에서 1개 행을 조회

hdfs vm cat format=json limit=1 /tmp/iis.json

vm 프로파일로 접속하여 /tmp/classloading.seq 파일에서 2개 레코드를 조회

hdfs vm cat format=sequence limit=2 /tmp/classloading.seq

JMX 클래스 로딩 로그 중 UnloadedClassCount와 LoadedClassCount만 /tmp/class.txt 경로에 출력

table classloading | hdfs vm put fields=UnloadedClassCount,LoadedClassCount /tmp/class.txt

sys\_cpu\_logs 로그를 /tmp 밑의 날짜별 디렉터리에 출력

table sys\_cpu\_logs | eval line=concat("idle: ", idle, ", kernel: ", kernel, ", user: ", user) | hdfs vm put partition=t /tmp/{logtime:yyyyMMdd}/cpu.txt

JMX 클래스로딩 로그 중 LoadedClassCount, UnloadedClassCount, TotalLoadedClassCount 출력

table classloading | hdfs vm put format=csv fields=LoadedClassCount,UnloadedClassCount,TotalLoadedClassCount /tmp/classloading.csv

JMX 클래스로딩 로그를 JSON 파일로 출력

table classloading | hdfs vm put format=json /tmp/classloading.json

JMX 클래스로딩 로그 전체를 HDFS 시퀀스 파일로 출력

table classloading | hdfs vm put format=sequence /tmp/classloading.seq

JMX 클래스로딩 로그 중 LoadedClassCount 값을 출력

table classloading | hdfs vm put format=sequence value\_type=long value\_field=LoadedClassCount /tmp/classloading.seq

### mongo

몽고DB를 탐색하거나, 쿼리 결과를 파일로 전송합니다.

#### 문법

mongo PROFILE [database=DB\_NAME] SUBCOMMANDmongo PROFILE [database=DB\_NAME] MONGODB\_METHOD

필수 매개변수

**PROFILE**

몽고DB 접속 프로파일. 웹 콘솔에서 프로파일을 구성할 수 있습니다.

ENT-3.10.2009.0, SNR-3.1.2008.0부터 몽고DB 프로파일이 접속 프로파일에 통합되었습니다.

**SUBCOMMAND**

몽고DB 세션에서 실행할 명령: dbs, cols

**cols**

database=DB\_NAME 옵션으로 지정한 데이터베이스의 컬렉션 목록을 출력합니다. 데이터베이스를 지정하지 않으면 모든 데이터베이스의 컬렉션 목록을 보여줍니다.

**dbs**

몽고DB의 모든 데이터베이스 목록을 출력합니다. 다른 매개변수 없이 단독으로 사용합니다.

**MONGO\_METHOD**

몽고DB 세션에서 실행할 네이티브 메서드. COL\_NAME으로 지정한 컬렉션에 대해 입력 레코드나 표현식 등을 매개변수 인자로 받아 메소드를 실행합니다.

선택 매개변수

**database=DB\_NAME**

cols 명령 또는 MONGO\_METHOD로 지정한 메서드의 대상이 되는 데이터베이스

#### 설명

COMMAND별 출력 필드

dbs 명령의 출력 필드는 다음과 같습니다.

* **name**: 데이터베이스 이름
* **disk\_usage**: 디스크 사용량 (바이트)
* **empty**: 데이터베이스가 비어 있으면 true, 그렇지 않으면 false

cols 명령의 출력 필드는 다음과 같습니다.

* **name**: 컬렉션 이름
* **type**: 타입 (collection)
* **options**: 컬렉션 구성 옵션
* **info**: 읽기 전용 여부 등 속성 정보
* **idIndex**: \_id 인덱스 명세

MONGO\_METHOD

mongodb 명령어는 다음과 같은 몽고DB 네이티브 메소드를 지원합니다. [몽고DB 레퍼런스 매뉴얼](https://www.mongodb.com/docs/manual/reference/method/)에서 각 메소드의 사용법을 확인할 수 있습니다.

* [db.COL\_NAME.find()](https://www.mongodb.com/docs/manual/reference/method/db.collection.find/): 문서 검색
* [db.COL\_NAME.insert()](https://www.mongodb.com/docs/manual/reference/method/db.collection.insert/): 컬렉션에 문서 삽입
* [db.COL\_NAME.updateOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/): 단일 문서 수정
* [db.COL\_NAME.updateMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateMany/): 여러 문서 수정
* [db.COL\_NAME.deleteOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.deleteOne/): 단일 문서 삭제
* [db.COL\_NAME.deleteMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.deleteMany/): 여러 문서 삭제

find()

db.COL\_NAME.find("FILTER\_EXPR", "PROJECTION\_EXPR")

COL\_NAME로 지정한 컬렉션에서 문서를 조회합니다.

**"FILTER\_EXPR"**

검색 조건(필터) 표현식. JSON 형식으로 작성합니다. [몽고DB 질의 연산자](https://www.mongodb.com/docs/manual/reference/operator/query/)를 사용할 수 있습니다. 매개변수로 null을 지정하면 모든 문서를 조회합니다.

**"PROJECTION\_EXPR"**

검색된 문서에서 가져올 필드 표현식. JSON 형식으로 작성합니다. 이 매개변수를 생략하면 메소드는 검색 조건과 일치하는 문서의 모든 필드를 반환합니다. 자세한 내용은 [Projection](https://www.mongodb.com/docs/manual/reference/method/db.collection.find/)을 참고하십시오.

insert()

db.COL\_NAME.insert()

입력으로 받는 레코드를 COL\_NAME으로 지정한 컬렉션에 문서로 기록합니다.

입력 레코드에 **\_id** 필드 값이 있으면 고유 식별 키(ObjectId)로 사용합니다. 고유 식별 키로 사용할 필드의 값은 모두 고유해야 합니다. 기록할 레코드에 **\_id** 필드 값이 없으면 몽고DB 드라이버가 12바이트의 16진수 고유 식별 키를 부여합니다. 사용 예 4, 5를 참조하십시오.

updateOne()

db.COL\_NAME.updateOne("KEY", "UPDATE")

COL\_NAME으로 지정한 컬렉션에서 입력받은 KEY 필드의 값으로 검색해 가장 먼저 검색된 문서에서 UPDATE 필드의 값을 수정합니다. 사용 예 6을 참조하십시오.

**\_id** 필드를 KEY로 사용할 때 고유식별자가 자동 생성된 12자리 바이너리인 경우, **\_id** 필드의 값을 바이너리 타입으로 설정해야 합니다.

updateMany()

db.COL\_NAME.updateMany("KEY\_LIST", "UPDATE\_LIST")

COL\_NAME으로 지정한 컬렉션에서 입력받은 KEY\_LIST 필드의 값으로 검색해 UPDATE\_LIST 필드의 값을 수정합니다. KEY\_LIST, UPDATE\_LIST는 모두 구분자로 쉼표(,)를 사용합니다.

db.COL\_NAME.updateOne() 메서드와 달리 검색된 모든 문서를 대상으로 수정합니다.

deleteOne()

db.COL\_NAME.deleteOne("KEY")

COL\_NAME으로 지정한 컬렉션에서 입력받은 KEY 필드의 값으로 검색해 가장 먼저 검색된 문서를 삭제합니다. 사용 예 7을 참조하십시오.

**\_id** 필드를 KEY로 사용할 때 고유식별자가 자동 생성된 12자리 바이너리인 경우, **\_id** 필드의 값을 바이너리 타입으로 설정해야 합니다.

deleteMany()

db.COL\_NAME.deleteMany("KEY\_LIST")

COL\_NAME으로 지정한 컬렉션에서 입력받은 KEY\_LIST 필드의 값과 일치하는 모든 문서를 삭제합니다. KEY\_LIST는 구분자로 쉼표(,)를 사용합니다.

#### 사용 예

nventory 컬렉션의 모든 문서 조회

mongo PROFILE db.inventory.find()

restaurants 컬렉션에서 stars 3점 이상의 문서 조회

mongo PROFILE db.restaurants.find("{stars: {$gte: 3}}")

restaurants 컬렉션에서 name, stars 필드만 출력

mongo PROFILE db.restaurants.find(null, "{name: true, stars: true}")

임의의 JSON 문서를 몽고DB restaurants 컬렉션에 입력

json "{ name: Café Con Leche, contact: { phone: 228-555-0149, email: cafeconleche@example.com, location: [-73.92502, 40.8279556] }, stars:3, categories: [Bakery, Coffee, Pastries] }" | mongo PROFILE db.restaurants.insert()

로그프레소 inventory 테이블의 최근 10건을 몽고DB inventory 컬렉션에 입력

table limit=10 inventory | mongo PROFILE db.inventory.insert()

name 필드를 기준으로 restaurants 컬렉션을 검색하고 stars 값을 수정

json "{}" | eval name="Café Con Leche", stars=4 | mongo PROFILE db.restaurants.updateOne("name", "stars")

ObjectId가 5982df1b7098262f64d4ffaf인 문서 1건을 검색하여 삭제

json "{}" | eval \_id = fromhex("5982df1b7098262f64d4ffaf") | mongo stream2 db.restaurants.deleteOne("\_id")

### rss

HTTP 통신을 통해 RSS1, RSS2, ATOM 형식의 피드를 수신해 출력합니다.

#### 문법

rss url="FEED\_URL" [strip=BOOL]

**url="FEED\_URL"**

RSS 피드의 URL을 입력합니다.

**strip=BOOL**

RSS 피드에서 HTML 태그 제거 여부(기본값: f)

#### 설명

RSS 피드를 읽어오면 각 레코드마다 아래와 같은 필드를 출력합니다.

* guid:식별자
* author:작성자
* title:제목
* content:본문
* link:URL 링크
* source:출처
* created\_at:생성시각
* modified\_at:변경시각

#### 사용 예

RSS 피드 조회

rss url="http://rss.slashdot.org/Slashdot/slashdotMain" strip=t

### sftp

SFTP 서버에서 파일 시스템을 탐색하거나, 쿼리 결과를 파일로 전송할 수 있습니다.

#### 문법

파일 목록 조회

sftp PROFILE ls PATH

파일 내용 읽기

sftp PROFILE cat [encoding=CHARSET] [limit=INT] [offset=INT] PATH

특정한 레코드의 필드 값을 텍스트, CSV, TSV, JSON 형식으로 파일 전송

sftp PROFILE put [append=t]|[overwrite=t] [encoding=CHARSET] [fields=FIELD\_1[,FIELD\_2,...]] [format={csv|json|text|tsv}] [multisession=t maxsession=INT] [partition=t] PATH

**PROFILE**

SFTP 연결에 사용할 프로파일을 지정합니다.

프로파일은 웹 콘솔에서 구성할 수 있습니다. ENT-3.10.2009.0. SNR-3.1.2008.0 배포 버전부터 SSH 프로파일이 접속 프로파일에 통합되었습니다.\* (ENT, STD) 시스템 설정 > 접속 프로파일\* (SNR) 시스템 > 접속 프로파일

**{cat|ls|put}**

sftp 세션에서 실행할 명령어를 입력합니다.

* cat: 서버에서 PATH로 지정된 경로의 파일 내용을 읽어와 line 필드에 출력합니다.
* ls: 서버에서 PATH로 지정된 경로의 파일 목록을 보여줍니다.
* put: 입력으로 전달받은 레코드나 fields 옵션으로 지정된 필드의 값들을 SFTP 서버에 파일로 전송합니다. 파일은 PATH로 지정된 경로에 생성됩니다.
* 텍스트 형식일 때 필드가 여러 개이면 탭으로 구분하고, 빈 필드는 '-'으로 표시
* CSV 형식일 때 첫번째 줄은 필드 이름 목록이며, 빈 필드는 빈 문자열로 표시

**append=BOOL**

SFTP 서버에 데이터를 전송할 때 사용할 수 있는 옵션으로, PATH로 지정된 파일이 있으면 파일에 이어서 씁니다. overwrite 옵션과 함께 사용할 수 없습니다.

**encoding=CHARSET**

파일 인코딩 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: https://www.iana.org/assignments/character-sets/character-sets.xhtml

**fields=FIELD[,FIELD,...]**

SFTP 서버에 데이터를 전송할 때 사용할 수 있는 옵션으로, 전송 대상 필드를 지정합니다. 여러 필드를 선택하려면 구분자로 쉼표(,)를 사용합니다.

* 텍스트 또는 CSV 파일로 전송할 때 이 옵션을 생략하면 line 필드를 출력합니다.
* JSON 파일로 전송할 때 이 옵션을 생략하면 모든 필드를 출력합니다.

appent=t 옵션을 사용할 때 데이터의 일관성을 유지할 수 있도록 항상 fields 옵션의 목록 순서를 동일하게 유지해주세요.

**format={csv|json|text|tsv}**

전송할 파일 형식을 지정합니다.

**limit=INT**

SFTP 서버에서 파일을 읽어올 때 출력할 행의 개수를 입력합니다. 기본값은 무제한입니다.

**multisession=t**

멀티세션을 사용할지 여부를 불린 값으로 지정합니다. 지정하지 않을 경우 사용하지 않습니다. 추가 세션을 여는데 시간이 더 걸려 오히려 사용하지 않을 때보다 성능이 떨어질 수 있어, 테스트 후 사용 여부를 결정하는 것이 바람직합니다.

**maxsession=INT**

멀티세션을 사용할 때, 최대로 열 세션 수를 지정합니다. 멀티세션 사용 여부를 체크하지 않고 이 옵션을 지정할 경우 쿼리가 실패합니다. 아무리 큰 수를 지정해도 접속 대상의 sshd\_config에서 지정한 MaxSessions 수 만큼 세션이 열립니다.

**offset=INT**

SFTP 서버에서 파일을 읽을 때 건너뛸 행 개수를 입력합니다. 기본값은 0입니다.

**overwrite=BOOL**

SFTP 서버에 데이터를 전송할 때 사용할 수 있는 옵션으로, PATH로 지정된 파일이 있으면 파일을 무시하고 덮어씁니다. append 옵션과 함께 사용할 수 없습니다.

**partition=BOOL**

t로 설정하면 매크로를 이용해 디렉터리 경로를 지정할 수 있습니다. 매크로를 이용하면 디렉터리나 파일 이름을 시간에 따라 변경합니다.

**PATH**

디렉터리 또는 파일의 절대 경로를 입력합니다. 전송할 때에는 디렉터리 경로가 아니라 단일 파일 경로를 입력해야 합니다. 파일을 조회할 때에 파일 경로에 와일드카드(\*)를 사용하면 특정 문자열 패턴을 포함한 모든 파일을 한 번에 조회할 수 있습니다.

#### 설명

partition 옵션을 t로 설정하면 매크로를 이용해 시간에 따라 디렉터리 및 파일 경로를 변경하도록 경로를 지정할 수 있습니다. 파티션 옵션을 지정하고 경로에 매크로를 사용하지 않으면 쿼리가 실패합니다.

사용할 수 있는 매크로는 {logtime:FMT}과 {now:FMT}가 있습니다.- {logtime:FMT}: 로그발생 시각을 기준으로 디렉터리나 파일에 이름 부여- {now:FMT}: 현재 시각을 기준으로 디렉터리나 파일에 이름 부여

매크로는 중괄호 쌍({ })으로 감싸 입력합니다. 입력 예시는 사용 예 6번을 참조하십시오.

#### 사용 예

srv 프로파일로 SSH 접속하여 원격 디렉터리 파일 조회

sftp srv ls /

조회 결과 필드들은 각각 아래의 의미를 가집니다:

* type(문자열): 디렉터리일 때 dir, 파일일 때 file
* is\_link(불리언): 심볼릭 링크 여부
* name(문자열): 파일 이름
* file\_size(정수): 파일 용량, 디렉터리일 때 0
* modified\_at(날짜): 마지막 수정 시각
* uid(정수): 소유자 ID
* gid(정수): 소유 그룹 ID
* perms(문자열): 파일 권한 정보

srv 프로파일로 접속하여 /logpresso.sh 파일의 첫 5행을 조회

sftp srv cat limit=5 /logpresso.sh

JMX 클래스 로딩 로그 중 UnloadedClassCount와 LoadedClassCount만 /tmp/class.txt 파일에 출력

table classloading | sftp srv put fields=UnloadedClassCount,LoadedClassCount /tmp/class.txt

JMX 클래스 로딩 로그를 /tmp/class.json 파일로 출력

table classloading | sftp srv put format=json /tmp/class.json

JMX 클래스 로딩 로그 중 LoadedClassCount, UnloadedClassCount, TotalLoadedClassCount 항목을 /tmp/class.csv 파일로 출력

table classloading | sftp srv put format=csv fields=LoadedClassCount,UnloadedClassCount,TotalLoadedClassCount /tmp/class.csv

디렉토리를 로그 시간 기준 년월일로, 파일 이름을 현재시간 기준 시분으로 JMX 클래스로딩 로그 중 LoadedClassCount, UnloadedClassCount, TotalLoadedClassCount 항목을 JSON 파일로 출력

table classloading | sftp srv put format=json partition=t fields=LoadedClassCount,UnloadedClassCount,TotalLoadedClassCount {logtime:/yyyy/MM/dd/}{now:HHmm}.txt

### wget

HTTP 통신으로 웹 리소스를 받아오거나, 수신한 결과를 line 필드에, 서버의 HTTP 코드는 \_wget\_code 필드에 출력합니다.

#### 문법

wget [auth="ID:PASSWD"] [body=FIELD] [encoding=CHARSET] [format={form|json|xml}] [header=FIELD\_MAP\_TYPE] [method={delete|get|post|put}] [selector="CSS\_SELECTOR"] [timeout=NUM] [url="SITE\_URL"]

**auth="ID:PASSWD"**

HTTP 기본 인증에 필요한 정보를 지정합니다.

**body=FIELD**

HTML 본문으로 사용할 필드를 지정합니다. method=post, method=put과 함께 사용합니다.

**encoding=CHARSET**

문자열 인코딩 형식(기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**format={form|json|xml}**

전송 포맷을 form, json, xml 중에서 선택할 수 있습니다. REST API 통신을 할 때 사용합니다(기본값: form).

**header=FIELD\_MAP\_TYPE**

HTTP 헤더로 사용 키-값으로 구성된 맵 타입 필드를 지정합니다. key, value 모두 문자열 타입인 값만 전송합니다. [dict()](https://docs.logpresso.com/ko/query/dict-function) 함수를 이용해 키-값 맵을 지정할 수도 있습니다:

**method={delete|get|post|put}**

HTTP 메서드를 지정합니다(기본값: get).

post 메서드는 다음과 같은 특징이 있습니다.

* 입력 레코드의 키-값을 x-www-form-urlencoded 형식으로 URL 인코딩하여 전송합니다.
* url 옵션과 함께 사용할 수 없으므로 header 옵션을 이용해 HTTP 헤더를 직접 입력합니다.

**selector="CSS\_SELECTOR"**

CSS의 selector와 동일한 문법으로 HTML DOM 트리에서 선택할 요소를 지정합니다.

**timeout=NUM**

HTTP 연결 타임아웃 시간을 초 단위로 지정합니다(기본값: 30초).

**url="SITE\_URL"**

연결할 웹 서비스 URL을 입력합니다. 지정된 웹 주소로 요청을 전송하고 응답을 수신합니다.

#### 사용 예

RSS 피드 제목 조회

wget url="https://logpresso.com/feed/" selector="item title" | explode elements | eval title = valueof(elements, "own\_text") | fields title

AbuseIPDB에서 IP 주소 평판 조회

json "{}" | eval ip = "118.25.6.39" | eval headers = dict("Key", "YOUR\_API\_KEY", "Accept", "application/json") | eval url = concat("https://api.abuseipdb.com/api/v2/check?ipAddress=", ip, "&maxAgeInDays=90") | wget method=get header=headers | parsejson | parsemap field=data

AbuseIPDB에 악성 IP 주소 신고

json "{}" | eval ip = "47.236.18.74", categories=14, comment = "Port scanning (count: 2790)" | eval headers = dict("Key", "YOUR\_API\_KEY", "Accept", "application/json") | eval url = concat("https://api.abuseipdb.com/api/v2/report") | wget method=post header=headers

# 소나 명령어

## 이벤트

### alert

입력 레코드를 이용하여 소나 이벤트를 생성합니다.

#### 문법

alert

#### 설명

alert는 컨트롤 노드에서만 사용 가능하고, 클러스터 관리자만 사용할 수 있습니다. 기본적으로 각 데이터 노드에서 실시간 룰 탐지 후 컨트롤 노드로 전송되는 이벤트를 수신하는 스트림 쿼리에 alert 명령을 설정해서 사용합니다.

중복된 이벤트가 수신된 경우 실시간 시나리오의 이벤트 축약 설정에 의해 제거될 수 있습니다. 또한, 실시간 시나리오 설정에 따라 티켓이 생성되거나 기존 티켓에 병합될 수 있습니다. 생성된 이벤트는 이벤트 메뉴에서 조회할 수 있습니다.

입력 레코드는 아래 규격을 만족해야 합니다.

입력 레코드는 다음과 같은 필드로 구성되어야 합니다:

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 이름 | 유형 | 타입 | 설명 |
| \_logger | 필수 | 32비트 정수 | 수집기 ID 식별자 |
| \_rule | 필수 | 32비트 정수 | 실시간 시나리오 ID 식별자 |
| \_time | 선택 | 날짜/시각 | 원본 이벤트 발생 시각. 값이 없거나 타입이 일치하지 않으면 입력 시점 시각으로 처리됩니다. |
| emp\_key | 선택 | 문자열 | 임직원 사번 |
| emp\_name | 선택 | 문자열 | 임직원 성명 |
| host\_ip | 선택 | IP 주소 | 사내 호스트 IP |
| src\_ip | 선택 | IP 주소 | 출발지 IP |
| src\_country | 선택 | 문자열 | 출발지 ISO 국가코드 |
| src\_port | 선택 | 32비트 정수 | 출발지 포트 |
| dst\_ip | 선택 | IP 주소 | 목적지 IP |
| dst\_country | 선택 | 문자열 | 목적지 ISO 국가코드 |
| dst\_port | 선택 | 32비트 정수 | 목적지 포트 |
| protocol | 선택 | 문자열 | 프로토콜 |
| action | 선택 | 문자열 | 대응 방식 |

### event

로그프레소 소나에서 시나리오 기반하여 탐지한 이벤트 목록을 조회합니다.

#### 문법

event [duration=INT{mon|w|d|h|m|s}] [from=yyyyMMddHHmmss] [to=yyyyMMddHHmmss] [order=STR] [raw=BOOL]

매개변수

duration, from, to과 같은 옵션으로 검색 기간/범위를 지정하지 않으면 모든 데이터를 검색합니다.

**duration=INT{mon|w|d|h|m|s}**

현재 시각을 기준으로 지정한 시간만큼 이전 데이터만 검색. mon(월), w(주), d(일), h(시), m(분), s(초) 단위로 입력합니다. 예를 들어 10s은 쿼리 실행 시각을 기준으로 "최근 10초"를 의미합니다. 이 옵션은 from, to와 함께 사용할 수 없습니다.

**from=yyyyMMddHHmmss**

검색할 기간의 시작 날짜를 yyyyMMddHHmmss 형식으로 지정. 입력한 시각부터 검색을 시작합니다. 앞부분만 입력하면 나머지 자리는 0으로 인식합니다. 예를 들어 20130605를 입력하면 20130605000000(2013년 6월 5일 0시 0분 0초)으로 인식합니다.

**to=yyyyMMddHHmmss**

검색할 기간의 끝 날짜를 yyyyMMddHHmmss 형식으로 지정. 입력한 시각은 검색 범위에 포함되지 않습니다.

**order=STR**

레코드의 정렬 순서 (기본값: desc)

* asc: 오름차순 정렬. 오래된 레코드부터 출력.
* desc: 내림차순 정렬. 최근 레코드부터 출력.

**raw=BOOL**

원본 이벤트 조회 여부(기본값: f)

* t: 이벤트를 원본 그대로 조회
* f: 정규화된 이벤트 정보를 조회

#### 설명

이 명령어는raw=f일 때 정규화된 이벤트 정보를, raw=t일 때 이벤트를 원본 형태 그대로 보여줍니다. 하나의 이벤트는 여러 개의 원본 로그에 매핑될 수 있으므로 정규화된 이벤트와 원본은 조회 레코드 개수에 차이가 있을 수 있습니다. 로그프레소 소나의 티켓 화면에서 근거자료로 활용되는 이벤트는 raw=t가 적용된 형태로 보여줍니다.

event의 출력 필드는 이벤트마다 각각 고유한 필드, 또는 마리아 DB의 데이터베이스 컬럼을 포함할 수 있으므로 가변적입니다. 다만, **\_time** 필드는 **\_log\_time**으로 표시합니다.

#### 사용 예

2023년 5월 23일 00:00:00부터 5월 23일 23:59:59까지 발생한 이벤트 조회

event from=20230523 to=20230524

## 자산 IP

### sonar-set-ip-address

입력 레코드에서 지정한 필드의 값을 내부 IP 자산 데이터베이스에 동기화합니다.

#### 문법

sonar-set-ip-address fields=FIELDS [batchsize=INT] [company=GUID]

필수 매개변수

**fields=FIELDS**

동기화할 대상 필드 목록. 쉼표와 공백문자(', ')를 구분자로 사용합니다. 하단 [입력 필드](https://docs.logpresso.com/ko/query/sonar-set-ip-address-command) 참조.

선택 매개변수

**batchsize=INT**

지정한 레코드 수의 배치 단위로 커밋을 실행합니다. 유효한 값의 범위는 1~5000 입니다. 대량의 IP 주소 정보를 동기화해야 하는 경우에 사용합니다. 이 옵션을 사용하면 트랜잭션 수가 감소하여 수행 성능이 향상되지만 오류 발생 시 배치 단위로 동기화에 실패합니다.

**company=GUID**

회사 (테넌트) GUID 식별자(기본값: 명령을 실행하는 사용자 계정이 소속된 회사의 GUID).

시스템 계정은 어느 회사에도 소속되어 있지 않습니다. 시스템 계정 권한으로 실행할 쿼리문은 반드시 이 옵션을 사용하여 동기화 대상 회사를 명시적으로 지정해야 합니다.

#### 설명

이 명령어는 입력 레코드의 ip 필드와 fields 옵션으로 지정한 필드 값을 기준으로 기존 내부 IP 주소 목록에 IP 주소 객체가 조회되면 입력 레코드 정보를 업데이트하고, IP 주소로 객체가 조회되지 않으면 새로 생성합니다.

입력 레코드는 반드시 ip 필드에 유효한 IP 주소 값을 포함해야 합니다. ip 필드 값은 문자열 타입이나 IP 주소 타입의 값이 모두 허용됩니다. ip 필드 값이 유효한 IP 주소가 아니면, 동기화에 실패하고 \_error 필드에 invalid ip 오류가 출력됩니다. ip 필드 값이 null인 경우에는 \_error 필드에 ip is null 오류가 출력됩니다.

fields 옵션에 동기화 대상 필드를 지정했으나 입력 레코드에 해당 필드가 존재하지 않으면 해당 필드의 값은 null입니다. 반대로, 명시적으로 fields 옵션에 동기화 대상 필드가 지정되지 않으면, 입력 레코드에 동기화 대상 필드와 동일한 이름의 필드가 존재하더라도 해당 값이 동기화되지 않습니다.

입력 필드

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 필드 | 이름 | 타입 | 길이 | 설명 |
| priority | 중요도 | 32비트 정수 |  | 3 (상), 2 (중), 1 (하) |
| category\_name | 장치 분류 | 문자열 | 255자 | 이름 불일치 시 미분류 처리 |
| hostname | 호스트명 | 문자열 | 255자 | 컴퓨터 이름 혹은 호스트명 |
| workgroup | 작업그룹 | 문자열 | 255자 | NT 도메인 혹은 워크그룹 |
| emp\_key | 담당자(정) 사번 | 문자열 | 255자 | 사번 불일치 시 무시됨 |
| emp\_key2 | 담당자(부) 사번 | 문자열 | 255자 | 사번 불일치 시 무시됨 |
| description | 비고 | 문자열 | 2000자 |  |
| os\_name | 운영체제 | 문자열 | 50자 |  |
| os\_ver | 운영체제 버전 | 문자열 | 20자 |  |
| mac | MAC 주소 | 문자열 | 20자 |  |
| location | 설치 위치 | 문자열 | 255자 |  |
| installed | 설치 일시 | 날짜 |  |  |
| ext0 | 확장 필드 #0 | 문자열 | 255자 |  |
| ext1 | 확장 필드 #1 | 문자열 | 255자 |  |
| ext2 | 확장 필드 #2 | 문자열 | 255자 |  |
| ext3 | 확장 필드 #3 | 문자열 | 255자 |  |
| ext4 | 확장 필드 #4 | 문자열 | 255자 |  |
| ext5 | 확장 필드 #5 | 문자열 | 255자 |  |
| ext6 | 확장 필드 #6 | 문자열 | 255자 |  |
| ext7 | 확장 필드 #7 | 문자열 | 255자 |  |
| ext8 | 확장 필드 #8 | 문자열 | 255자 |  |
| ext9 | 확장 필드 #9 | 문자열 | 255자 |  |

* category\_name: 장치 분류명이 내부 IP 목록과 일치하지 않으면 '미분류'로 처리합니다.
* emp\_key, emp\_key2: 필드의 값이 소나의 임직원 테이블에서 조회되지 않으면 입력으로 전달받은 사번을 무시합니다.

로그프레소 셸에서 다음과 같이 소나 전역 설정으로 내부 IP 자산 필드를 사용자 정의할 수 있습니다.logpresso> sonar.setGlobalOption ip\_custom\_fields "0=제조사,1=모델명"

출력 필드

이 명령어는 입력 레코드의 모든 필드를 그대로 출력하지만 오류가 발생하는 경우 \_error 필드를 추가로 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_error | 문자열 | 오류 내용 |

#### 사용 예

지니안 NAC 자산 정보를 동기화

다음 예시는 지니안 NAC 앱의 확장 명령어 [genian-nac-nodes](https://logpresso.store/ko/apps/genian-nac/commands/genian-nac-nodes)를 사용합니다. 지니안 NAC의 전체 IP 자산 정보를 로그프레소 소나에 동기화할 수 있습니다.

genian-nac-nodes | rename nt\_domain as workgroup, first\_seen as installed, platform as os\_name, nic\_vendor as ext0, type as category\_name| eval priority = if(category\_name != "PC", 2, 1)| sonar-set-ip-address batchsize=10 fields="priority, category\_name, hostname, workgroup, emp\_key, emp\_key2, description, os\_name, os\_ver, mac, location, installed, ext0"

## 데이터셋

### dataset

지정된 GUID에 해당하는 데이터셋을 조회합니다. 클러스터 관리자, 회사 관리자 권한 계정, 해당 데이터셋을 소유한 계정을 제외하고는 쿼리를 실행할 수 없습니다.

#### 문법

dataset guid=DATASET\_GUID

필수 매개변수

**guid=DATASET\_GUID**

데이터셋을 생성할 때 할당된 식별자

#### 사용 예

특정 데이터셋 조회

dataset guid=ac7c717e-86c3-482b-a08f-7b4a572cec79

#### 호환성

dataset 명령은 SNR #1059 2019-07-03\_19-12 버전부터 사용 가능합니다.

## 위협 인텔리전스

### matchfeed

입력 레코드에서 지정된 필드의 값을 위협 인텔리전스 피드와 대조하여 필터링합니다.

#### 문법

matchfeed [invert=BOOL] [name=FEED\_ID] [type=TYPE] fields=FIELD\_LIST

'name=FEED\_ID'와 'type=TYPE'을 동시에 사용할 수 없습니다. 둘 중 하나만 사용하십시오.

필수 매개변수

**name=FEED\_ID**

fields로 지정된 필드 레코드와 대조할 위협 인텔리전스 피드 식별자(기본값: 없음). 식별자(FEED\_ID)는 다음 표를 참조하십시오.

|  |  |  |  |
| --- | --- | --- | --- |
| 이름 | 유형 | 이름 | 설명 |
| otx | IP 주소 | OTX 피드 | ISAC이 제공하는 OTX (Open Threat Exchange) IP 주소 피드 |
| tor | IP 주소 | TOR 프록시 | ISAC이 제공하는 Tor 네트워크의 EXIT 라우터 IP 주소 피드 |
| mdl\_domain | 도메인 | 악성 도메인 주소 목록 | ISAC이 제공하는 랜섬웨어 등 악성코드 도메인 주소 피드 |
| mdl\_ip | IP 주소 | 악성 IP 주소 목록 | ISAC이 제공하는 악성 IP 주소 피드 |
| abusech | 도메인 | abuse.ch | ISAC이 제공하는 abuse.ch 악성 도메인 주소 피드 |
| malc0de | MD5 | malc0de | ISAC이 제공하는 malc0de.com 악성코드 MD5 해시 피드 |
| logpresso\_cti\_ip | IP 주소 | 악성 IP 주소 목록 | 로그프레소 CTI 서비스에서 제공하는 악성 IP 주소 피드 |
| logpresso\_cti\_domain | 도메인 | 악성 도메인 주소 목록 | 로그프레소 CTI 서비스에서 제공하는 도메인 주소 피드 |
| logpresso\_cti\_url | URL | 악성 URL 목록 | 로그프레소 CTI 서비스에서 제공하는 URL 피드 |
| logpresso\_cti\_md5 | MD5 | 악성코드 MD5 해시 목록 | 로그프레소 CTI 서비스에서 제공하는 MD5 해시 피드 |
| logpresso\_cti\_sha1 | SHA1 | 악성코드 SHA1 해시 목록 | 로그프레소 CTI 서비스에서 제공하는 SHA1 해시 피드 |
| logpresso\_cti\_sha256 | SHA256 | 악성코드 SHA256 해시 목록 | 로그프레소 CTI 서비스에서 제공하는 SHA256 해시 피드 |

이 외에, 소나에 설치한 앱이 제공하는 피드를 사용할 수 있습니다. 앱이 제공하는 피드 식별자는 해당 앱의 문서를 참조하십시오.

**type=TYPE**

위협 인텔리전스 피드와 대조할 값의 유형(기본값: 없음). 지정할 수 있는 값은 domain, email, ip, md5, sha256, url 입니다. type 옵션을 사용하면 해당 유형 정보가 포함된 모든 위협 인텔리전스 피드와 fields로 지정된 필드 값을 대조합니다.

* domain: 유형이 도메인 주소인 모든 피드
* ip: 유형이 IP 주소인 모든 피드
* md5: 유형이 MD5 해시인 모든 피드
* sha1: 유형이 SHA1 해시인 모든 피드
* sha256: 유형이 SHA256 해시인 모든 피드
* url: 유형이 URL인 모든 피드

**fields=FIELD\_LIST**

위협 인텔리전스 피드와 값을 대조할 필드 목록. 공백문자 없이 구분자로 쉼표(,)를 사용합니다.

선택 매개변수

**invert=BOOL**

fields 옵션으로 지정한 값을 위협 인텔리전스 피드와 대조한 결과를 반환하는 방법(기본값: f)

* t: 대조 결과에 fields로 지정한 값이 포함되지 않은 레코드를 반환
* f: 대조 결과에 fields로 지정한 값이 포함된 레코드를 반환

#### 설명

피드 식별자와 명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| feed\_name | 문자열 | 위협 인텔리전스 피드 식별자 |
| feed\_field | 문자열 | 위협 정보가 발견된 필드 이름 |
| feed\_invert | 불리언 | invert 옵션 값 |

### node-feed

node-feed 명령어는 수집노드에서 분석 노드와 동기화된 위협 인텔리전스 데이터를 조회합니다. node-feed 명령어는 수집 노드에서만 사용 가능합니다.

node-feed 명령은 SNR #1118 2019-08-01\_18-34 버전부터 사용 가능합니다.

#### 문법

node-feed name=[otx|tor|mdl\_ip|mdl\_domain|malc0de|abusech]

필수 매개변수

**name**

위협 인텔리전스 피드 식별자. otx, tor, mdl\_ip, mdl\_domain, malc0de, abusech 중 선택

#### 설명

피드 식별자는 [matchfeed](https://docs.logpresso.com/ko/query/matchfeed-command) 명령어 설명에 있는 피드 식별자를 참조하십시오.

|  |  |  |  |
| --- | --- | --- | --- |
| 이름 | 유형 | 이름 | 설명 |
| otx | IP 주소 | OTX 피드 | OTX (Open Threat Exchange) 피드를 이용하여 IP 주소 평판 정보를 실시간 대조합니다. |
| tor | IP 주소 | TOR 프록시 | EXIT 라우터 IP 주소를 대조하여 TOR 프록시를 통해 접속하는 IP 주소를 실시간으로 탐지합니다. |
| mdl\_domain | 도메인 | 멀웨어 도메인리스트 | 랜섬웨어 등 악성코드 도메인 피드를 대조하여 내부에서 C&C 도메인으로 접속하면 실시간 탐지합니다. |
| mdl\_ip | IP 주소 | 악성코드 IP 주소 목록 | 랜섬웨어 등 악성코드 IP 피드를 대조하여 내부에서 C&C IP 주소로 접속하면 실시간 탐지합니다. |
| abusech | 도메인 | Abuse.ch | 랜섬웨어가 접속하는 도메인을 실시간으로 탐지합니다. |
| malc0de | MD5 | malc0de | malc0de.com 악성코드 MD5 데이터베이스를 대조하여 악성 바이너리를 탐지합니다. |

#### 사용 예

수집 서버에서 동기화된 OTX 피드 조회

node-feed name=otx

수집 서버에서 동기화된 멀웨어 IP 리스트 조회

node-feed name=mdl\_ip

## 행위 프로파일

### behavior

행위 프로파일 설정에 따라 생성된 최신 데이터를 조회합니다. 이 명령어는 행위 프로파일 데이터가 위치한 분석 노드에서만 사용할 수 있습니다.

#### 문법

behavior guid=PROFILE\_GUID [from=yyyyMMddHHmmss] [to=yyyyMMddHHmmss]

필수 매개변수

**guid=PROFILE\_GUID**

행위 프로파일 GUID 식별자

선택 매개변수

**from**

검색할 기간의 시작 날짜를 yyyyMMddHHmmss 형식으로 지정. 입력한 시각부터 검색을 시작합니다. 앞부분만 입력하면 나머지 자리는 0으로 인식합니다. 예를 들어 20130605를 입력하면 20130605000000(2013년 6월 5일 0시 0분 0초)으로 인식합니다.

**to**

검색할 기간의 끝 날짜를 yyyyMMddHHmmss 형식으로 지정. 입력한 시각은 검색 범위에 포함되지 않습니다. 입력 방식은 from과 같습니다.

#### 설명

behavior 명령어는 행위 프로파일 설정에 따라 생성된 최신 데이터를 조회합니다. 애드혹 분석이나 배치 시나리오 탐지 시 조인하여 연관 분석하는 용도로 사용할 수 있습니다. 명령어가 실행되는 동안 guid 옵션으로 지정한 행위 프로파일은 읽기 잠금 상태가 됩니다.

### matchbehavior

행위 프로파일에 설정된 키 필드를 기준으로 행위 프로파일을 검색하여 검색된 레코드의 값 필드를 출력 레코드에 추가합니다.

#### 문법

matchbehavior [invert=BOOL] [verify=BOOL] guid=PROFILE\_GUID

필수 매개변수

**guid=PROFILE\_GUID**

행위 프로파일 GUID 식별자

선택 매개변수

**invert=BOOL**

검색 결과의 출력 형식(기본값: f)

* t: 행위 프로파일에 기준 키가 포함되어 있지 않으면 출력
* f: 행위 프로파일에 기준 키가 포함되어 있어 있으면 출력

**verify=BOOL**

쿼리 파싱 단계에서 행위 프로파일 개체의 유효성 검사 여부(기본값: t).

* t: 행위 프로파일 개체의 유효성을 검증
* f: 행위 프로파일 개체의 유효성을 검증하지 않음. 이 옵션은 시스템이 정책 동기화 단계에서 문법 오류를 내지 않도록 하기 위해 설정합니다.

#### 설명

matchbehavior 명령어는 행위 프로파일에 설정된 키 필드를 기준으로 행위 프로파일을 검색하여 검색된 레코드의 값 필드를 출력 레코드에 추가합니다. 행위 프로파일의 키 필드는 문자열 혹은 IP 주소 타입만 허용됩니다. 그 외의 타입은 검색 실패로 간주됩니다. invert 옵션이 활성화된 경우, 키 필드 기준으로 행위 프로파일 검색에 실패한 경우에만 출력을 내보냅니다. 출력 필드는 다음 표를 참조하세요.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| behavior\_guid | 문자열 | 행위 프로파일 GUID 식별자 |
| behavior\_invert | 불리언 | invert 옵션 값 |

### node-behavior

수집 노드에서 분석 노드와 동기화된 행위 프로파일 데이터를 조회합니다. 명령어는 수집 노드에서만 사용 가능합니다.

#### 문법

node-behavior [guid=PROFILE\_GUID]

선택 매개변수

**guid=PROFILE\_GUID**

행위 프로파일 GUID 식별자. 식별자를 지정하지 않으면 동기화된 행위 프로파일의 목록을 조회합니다.

#### 설명

guid 옵션으로 행위 프로파일 GUID 식별자를 지정하면 분석 노드와 동기화된 행위 프로파일의 필드를 조회합니다. 식별자를 지정하지 않으면 동기화된 행위 프로파일의 목록을 조회합니다.

행위 프로파일 GUID를 지정하지 않으면 다음 표와 같은 값을 보여줍니다.

**GUID를 생략했을 때 출력되는 필드**

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| id | 정수 | 정수 식별자 |
| guid | 문자열 | 행위 프로파일 GUID 식별자 |
| name | 문자열 | 행위 프로파일 이름 |
| description | 문자열 | 행위 프로파일 설명 |
| row | 정수 | 행위 프로파일 데이터 건수 |
| curr\_ver | 정수 | 현재 버전 번호 |
| company\_guid | 문자열 | 회사 GUID 식별자 |
| company\_name | 문자열 | 회사 이름 |
| schedule | 문자열 | 행위 프로파일 데이터 생성 주기 (CRON 형식) |
| key\_fields | 배열 | 행위 프로파일 키 필드 목록 (이름, 타입) |
| query | 문자열 | 행위 프로파일 재생성 쿼리 |
| created | 문자열 | 행위 프로파일 생성 일시 |
| updated | 문자열 | 행위 프로파일 마지막 수정 일시 |

#### 사용 예

수집 서버에서 동기화된 행위 프로파일 목록 조회

node-behavior

수집 서버에서 동기화된 특정 행위 프로파일 데이터 조회

node-behavior guid=c0a8c07f-34e3-48ca-a91c-5bb35684ae79

#### 호환성

node-behavior 명령은 SNR #1118 2019-08-01\_18-34 버전부터 사용 가능합니다.

## 주소 그룹

### matchblackip

주어진 주소 그룹을 이용하여 입력 레코드를 필터링합니다.

#### 문법

matchblackip [invert=BOOL] [verify=BOOL] fields=TARGET\_FIELD guid=BLACKLIST\_GUID

필수 매개변수

**guid=BLACKLIST\_GUID**

주소 그룹 GUID 식별자

**fields=TARGET\_FIELD**

쉼표(,)로 구분된 대상 필드 목록

선택 매개변수

**invert=BOOL**

검색 결과의 출력 형식(기본값: f)

* t: 주소 그룹에 대상 필드 값이 포함되어 있지 않으면 출력
* f: 주소 그룹에 대상 필드 값이 포함되어 있으면 출력

**verify=BOOL**

쿼리 파싱 단계에서 주소 그룹 개체의 유효성 검사 여부(기본값: t).

* t: 주소 그룹 개체의 유효성을 검증
* f: 주소 그룹 개체의 유효성을 검증하지 않음. 이 옵션은 시스템이 정책 동기화 단계에서 문법 오류를 내지 않도록 하기 위해 설정합니다.

#### 설명

matchblackip 명령어는 주어진 주소 그룹를 이용하여 입력 레코드를 필터링합니다. 대상 필드 중 하나라도 지정된 주소 그룹에 포함되면 출력으로 내보냅니다. invert 옵션이 활성화된 경우, 대상 필드 중 하나도 주소 그룹에 포함된 값이 없어야 출력으로 내보냅니다.

명령 실행 후 출력되는 필드는 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| blackip\_guid | 문자열 | 주소 그룹 GUID 식별자 |
| blackip\_name | 문자열 | 주소 그룹 이름 |
| blackip\_field | 문자열 | 블랙 IP 발견 필드 이름 |
| blackip\_invert | 불린 | invert 옵션 값 |

### node-ip-blacklist

수집 노드에서 분석 노드와 동기화된 해당 식별자의 주소 그룹 항목을 조회합니다. 이 명령은 수집 노드에서만 사용 가능합니다.

#### 문법

node-ip-blacklist [guid=BLACKLIST\_GUID]

선택 매개변수

**guid=BLACKLIST\_GUID**

주소 그룹 GUID 식별자

#### 설명

guid 옵션으로 주소 그룹 GUID 식별자를 지정하면 수집 노드에서 분석 노드와 동기화된 해당 식별자의 주소 그룹 항목을 조회합니다. 식별자를 입력하지 않으면 동기화된 주소 그룹 목록을 조회합니다.

명령 실행 후 출력되는 필드는 다음 표를 참조하십시오.

**주소 그룹 GUID 지정 시 출력 필드**

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| ip | IP | 해당 블랙리스트에 있는 IP |
| description | 문자열 | IP 설명 |

**주소 그룹 GUID 미지정 시 출력 필드**

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| id | 정수 | 정수 식별자 |
| guid | 문자열 | 주소 그룹 GUID 식별자 |
| name | 문자열 | 주소 그룹 이름 |
| description | 문자열 | 주소 그룹 설명 |
| count | 정수 | IP 항목 갯수 |
| version | 정수 | 주소 그룹 버전 |
| company\_guid | 문자열 | 회사 GUID 식별자 |
| company\_name | 문자열 | 회사 이름 |
| created | 문자열 | 주소 그룹 생성 일시 |
| updated | 문자열 | 주소 그룹 마지막 수정 일시 |

#### 사용 예

수집 서버에서 동기화된 주소 그룹 목록 조회

node-ip-blacklist

수집 서버에서 동기화된 특정 주소 그룹 항목 조회

node-ip-blacklist guid=efd0c9cf-8582-4d5a-938d-9bb6a990579c

#### 호환성

node-ip-blacklist 명령은 SNR #1118 2019-08-01\_18-34 버전부터 사용 가능합니다.

## 네트워크 대역

### matchnet

필드의 IP 주소 값이 지정된 네트워크 대역에 포함되는지 확인하고 결과를 출력합니다.

#### 문법

matchnet [invert=BOOL] [verify=BOOL] field=TARGET\_FIELD guid=NET\_GUID [tag=BOOL]

tag 매개변수는 4.0.2312.0 버전부터 사용 가능합니다.

필수 매개변수

**field=TARGET\_FIELD**

대상 필드 이름. 대상 필드의 값은 IPv4 주소, 문자열, 32비트 정수 타입이어야 하며, 그 외의 키 값은 검색 실패로 간주합니다. 문자열 및 32비트 정수 타입 값은 유효한 IPv4 주소로 변환되는 경우에만 검색을 시도합니다.

**guid=NET\_GUID**

네트워크 대역 GUID 식별자

선택 매개변수

**invert=BOOL**

검색 결과의 출력 형식(기본값: f)

* t: field 매개변수로 지정한 IP 주소 값이 네트워크 대역에 포함되지 않으면 출력
* f: field 매개변수로 지정한 IP 주소 값이 네트워크 대역에 포함되면 출력

**verify=BOOL**

쿼리 파싱 단계에서 네트워크 대역 개체의 유효성 검사 여부(기본값: t).

* t: 네트워크 대역 개체의 유효성을 검증
* f: 네트워크 대역 개체의 유효성을 검증하지 않음. 이 옵션은 시스템이 정책 동기화 단계에서 문법 오류를 내지 않도록 하기 위해 설정합니다.

**tag=BOOL**

매칭된 네트워크 대역 개체 정보 표시 여부(기본값: f).

* t: id, start\_ip, end\_ip, cidr 속성을 포함한 맵을 \_matchnet\_result 필드에 출력
* f: 네트워크 대역 개체의 정보를 출력하지 않음

### node-subnet-group

수집 노드에서 분석 노드와 동기화된 네트워크 대역 항목을 조회합니다. 이 명령어는 수집 노드에서만 사용할 수 있습니다.

#### 문법

node-subnet-group [guid=NET\_GUID]

**guid=NET\_GUID**

네트워크 대역 GUID 식별자. 식별자를 지정하지 않으면 동기화된 네트워크 대역의 목록을 출력합니다.

#### 설명

guid 옵션으로 네트워크 대역 GUID 식별자를 지정하면 지정된 GUID에 해당하는 네트워크 대역 정보를 보여줍니다. GUID를 지정하지 않으면 수집 노드에 동기화된 모든 네트워크 대역 목록을 보여줍니다.

**네트워크 대역 GUID 지정 시 출력 필드**

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| start\_ip | IP 주소 | IPv4 대역 범위 시작 IP 주소. 유형이 단일, CIDR인 경우, 해당 항목의 IP 주소. |
| end\_ip | IP 주소 | IPv4 대역 범위 끝 IP 주소. (유형이 범위인 항목의 IP 주소에만 해당) |
| cidr | 정수 | CIDR 값 |
| description | 문자열 | 설명 |

**네트워크 대역 GUID 미지정 시 출력 필드**

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| id | 정수 | 정수 식별자 |
| guid | 문자열 | 네트워크 대역 GUID 식별자 |
| name | 문자열 | 네트워크 대역 이름 |
| description | 문자열 | 네트워크 대역 설명 |
| subnet\_count | 정수 | 대역 항목 갯수 |
| company\_guid | 문자열 | 회사 GUID 식별자 |
| company\_name | 문자열 | 회사 이름 |
| user\_name | 문자열 | 계정 이름 |
| user\_guid | 문자열 | 계정 GUID 식별자 |

#### 사용 예

수집 서버에서 동기화된 네트워크 대역 목록 조회

node-subnet-group

수집 서버에서 동기화된 특정 네트워크 대역 항목 조회

node-subnet-group guid=96f342ee-0aa2-4234-ac7c-37a50c38b7bc

#### 호환성

node-subnet-group 명령은 SNR #1118 2019-08-01\_18-34 버전부터 사용 가능합니다.

## 포트 그룹

### matchport

입력 레코드의 포트, 프로토콜 값을 포트 그룹과 대조하고 결과를 출력합니다. 입력 레코드의 포트나 프로토콜 값이 null이거나 타입이 일치하지 않는 경우 출력하지 않습니다.

#### 문법

matchport [invert=BOOL] [port=PORT\_FIELD] [protocol=PROTO\_FIELD] [verify=BOOL] guid=PORT\_GUID

필수 매개변수

**guid=PORT\_GUID**

포트 그룹 GUID 식별자

선택 매개변수

**invert=BOOL**

검색 결과의 출력 형식(기본값: f)

* t: port, protocol 옵션으로 지정한 필드의 값이 포트 그룹에 포함되지 않으면 출력
* f: port, protocol 옵션으로 지정한 필드의 값이 포트 그룹에 포함되어 있으면 출력

**port=PORT\_FIELD**

포트 필드의 이름(기본값: port)

**protocol=PROTO\_FIELD**

프로토콜 필드 이름(기본값: protocol)

**verify=BOOL**

쿼리 파싱 단계에서 포트 그룹 개체의 유효성 검사 여부(기본값: t).

* t: 포트 그룹 개체의 유효성을 검증
* f: 포트 그룹 개체의 유효성을 검증하지 않음. 이 옵션은 시스템이 정책 동기화 단계에서 문법 오류를 내지 않도록 하기 위해 설정합니다.

### node-port-group

수집 노드에서 분석 노드와 동기화된 포트 그룹 항목을 조회합니다. 명령어는 수집 노드에서만 사용할 수 있습니다.

#### 문법

node-port-group [guid=PORT\_GUID]

선택 매개변수

**guid=PORT\_GUID**

포트 그룹 GUID

#### 설명

guid 옵션으로 포트 그룹 GUID를 지정하면 GUID에 해당하는 포트 그룹 정보를 보여줍니다. GUID 식별자를 지정하지 않으면 수집 노드에 동기화된 모든 포트 그룹 목록을 보여줍니다.

출력 필드

guid에 포트 그룹 GUID를 지정하면 다음과 같은 필드를 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| protocol | 문자열 | 프로토콜(TCP 혹은 UDP) |
| start | 정수 | 포트 범위 시작 값 |
| end | 정수 | 포트 범위 끝 값 |
| description | 문자열 | 설명 |

guid에 포트 그룹 GUID를 지정하지 않으면 다음과 같은 필드를 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| id | 정수 | 정수 식별자 |
| guid | 문자열 | 포트 그룹 GUID 식별자 |
| name | 문자열 | 포트 그룹 이름 |
| description | 문자열 | 포트 그룹 설명 |
| port\_count | 정수 | 포트 범위 항목 갯수 |
| company\_guid | 문자열 | 회사 GUID 식별자 |
| company\_name | 문자열 | 회사 이름 |
| user\_name | 문자열 | 계정 이름 |
| user\_guid | 문자열 | 계정 GUID 식별자 |
| created\_at | 문자열 | 포트 그룹 생성 일시 |
| updated\_at | 문자열 | 포트 그룹 마지막 수정 일시 |

#### 사용 예

수집 서버에서 동기화된 포트 그룹 목록 조회

node-port-group

수집 서버에서 동기화된 특정 포트 그룹 항목 조회

node-port-group guid=2da1fa00-da63-4fb5-a443-46260c555697

#### 호환성

node-port-group 명령은 SNR #1118 2019-08-01\_18-34 버전부터 사용 가능합니다.

## 패턴 그룹

### matchsig

필드의 문자열 값이 지정된 패턴 그룹에 포함되는지 확인하고 결과를 출력합니다.

#### 문법

matchsig [invert=BOOL] [verify=BOOL] guid=SIG\_GUID field=TARGET\_FIELD

필수 매개변수

**guid=SIG\_GUID**

패턴 그룹 GUID 식별자

**field=TARGET\_FIELD**

대상 필드 이름. 대상 필드의 값은 문자열 타입이어야 하며, 그 외의 키 값은 검색 실패로 간주합니다.

선택 매개변수

**invert=BOOL**

검색 결과의 출력 형식(기본값: f)

* t: 대상 필드의 값이 패턴 그룹에 포함되어 있지 않으면 출력
* f: 대상 필드의 값이 패턴 그룹에 포함되어 있어 있으면 출력

**verify=BOOL**

쿼리 파싱 단계에서 패턴 그룹 개체의 유효성 검사 여부(기본값: t).

* t: 패턴 그룹 개체의 유효성을 검증
* f: 패턴 그룹 개체의 유효성을 검증하지 않음. 이 옵션은 시스템이 정책 동기화 단계에서 문법 오류를 내지 않도록 하기 위해 설정합니다.

#### 설명

패턴 그룹은 네트워크 침입탐지 장비처럼 수천 개 이상의 키워드를 동시에 테스트할 수 있도록 아호 코라식(Aho-Corasick) 알고리즘을 사용합니다. 먼저 입력 문자열에 대해 패턴 그룹에 속한 모든 키워드를 한 번에 매칭하고, 그 후에 키워드로 선별된 패턴들의 검증식을 순차적으로 실행하여 최종적으로 패턴과 일치하는 룰 이름을 출력합니다.

**패턴 예시**

|  |  |  |
| --- | --- | --- |
| expr (필수) | expr2 (선택) | rule (필수) |
| 키워드 패턴: 1차 고속 탐지 | 불린 표현식: 2차 필터링 | 패턴 이름 |
| "addextendedproc" and "xp\_cmdshell" |  | xp\_cmdshell |
| "REMOTE\_ADDR" and ("fputs" or "fwrite") | path == "lib.php" | zb now\_connect |

예를 들어, 패턴 "REMOTE\_ADDR" and ("fputs" or "fwrite") 검증식 path == "lib.php" 룰 zb now connect 로 설정한 경우, 대상 필드 값에서 REMOTE\_ADDR 문자열과 fputs 혹은 fwirte 문자열이 동시에 검색되는지 확인하고, 그 후에 path 필드 값이 lib.php 문자열과 일치하는지 확인합니다.

### node-pattern-group

수집 노드에서 분석 노드와 동기화된 패턴 그룹 항목을 조회합니다. 이 명령어는 수집 노드에서만 사용할 수 있습니다.

#### 문법

node-pattern-group [guid=SIG\_GUID]

선택 매개변수

**guid=SIG\_GUID**

패턴 그룹 GUID 식별자

출력 필드

guid 옵션으로 패턴 그룹 GUID 식별자를 지정하면 지정된 GUID에 해당하는 패턴 그룹정보를 보여줍니다. GUID를 지정하지 않으면 수집 노드에 동기화된 모든 패턴 그룹 목록을 보여줍니다.

**패턴 그룹 GUID 지정 시 출력 필드**

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| expr | 문자열 | aho-corasick 멀티 패턴 검색용 키워드 불린 조합 |
| expr2 | 문자열 | expr 매칭 후 2차 검사 표현식 |
| rule | 문자열 | 패턴 명칭 (매칭 시 출력에 태깅되는 이름) |

**패턴 그룹 GUID 미지정 시 출력 필드**

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| id | 정수 | 정수 식별자 |
| guid | 문자열 | 패턴 그룹 GUID 식별자 |
| name | 문자열 | 패턴 그룹 이름 |
| description | 문자열 | 패턴 그룹 설명 |
| pattern\_count | 정수 | 패턴 항목 갯수 |
| company\_guid | 문자열 | 회사 GUID 식별자 |
| company\_name | 문자열 | 회사 이름 |
| user\_name | 문자열 | 계정 이름 |
| user\_guid | 문자열 | 계정 GUID 식별자 |

#### 사용 예

수집 서버에서 동기화된 패턴 그룹 목록 조회

node-pattern-group

수집 서버에서 동기화된 특정 패턴 그룹 항목 조회

node-pattern-group guid=b5ce2e95-67b9-4d64-8f6e-2746264a58d2

#### 호환성

node-pattern-group 명령은 SNR #1118 2019-08-01\_18-34 버전부터 사용 가능합니다.

# 포렌식 명령어

## 윈도우 아티팩트

### automatic-destinations-file

점프 목록 파일 중 automaticDestinations-ms 파일 유형을 조회합니다.

#### 문법

automatic-destinations-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

automaticDestinations-ms 파일 경로. 이 파일은 일반적으로 C:\Users<username>\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations 위치에 있습니다.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_file | 문자열 | 파일 이름 |  |
| app\_id | 문자열 | 앱 ID | 예: Microsoft Paint 6.1 |
| file\_ctime | 날짜 | 점프 파일 생성 일시 |  |
| file\_mtime | 날짜 | 점프 파일 수정 일시 |  |
| file\_atime | 날짜 | 점프 파일 파일 접근 일시 |  |
| target\_file\_size | 64비트 정수 | 대상 파일 크기 |  |
| target\_file\_attrs | 배열 | 대상 파일 속성 목록 | 예: FILE\_ATTRIBUTE\_ARCHIVE |
| target\_file\_ctime | 날짜 | 대상 파일 생성 일시 |  |
| target\_file\_mtime | 날짜 | 대상 파일 수정 일시 |  |
| target\_file\_atime | 날짜 | 대상 파일 접근 일시 |  |
| drive\_serial | 64비트 정수 | 대상 드라이브 시리얼 |  |
| drive\_type | 문자열 | 대상 드라이브 유형 | 예: DRIVE\_FIXED, DRIVE\_REMOVABLE, DRIVE\_CDROM |
| volume\_label | 문자열 | 대상 볼륨 이름 | 예: OS |
| local\_path | 문자열 | 대상 로컬 경로 | 예: C:\Program Files\Internet Explorer\iexplore.exe |
| local\_path\_unicode | 문자열 | 대상 로컬 경로 (유니코드) |  |
| net\_name | 문자열 | 대상 네트워크 공유 이름 | 예: \FILE\SAMPLE |
| common\_path\_suffix | 문자열 | 대상 네트워크 공유 경로 |  |
| common\_path\_suffix\_unicode | 문자열 | 대상 네트워크 공유 경로 (유니코드) |  |
| show\_window | 문자열 | 윈도우 표시 옵션 | 예: SHOW, SHOW\_NORMAL, SHOW\_MAXIMIZED |
| shortcut\_name | 문자열 | 바로가기 이름 | 예: Run PuTTYgen |
| working\_dir | 문자열 | 작업 디렉터리 경로 |  |
| relative\_path | 문자열 | 상대 경로 |  |
| cmd\_args | 문자열 | 명령줄 인자 | 예: -newtab |
| icon\_location | 문자열 | 아이콘 경로 | 예: C:\Windows\SYSTEM32\IEFRAME.dll |
| hot\_key | 문자열 | 단축키 | 예: CTRL + ALT + R |
| target\_id\_list | 배열 | 대상 ID 목록 |  |

### custom-destinations-file

점프 목록 파일 중 customDestinations-ms 파일 유형을 조회합니다.

#### 문법

custom-destinations-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

customDestinations-ms 파일 경로. 이 파일은 일반적으로 C:\Users<username>\AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations 위치에 있습니다.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_file | 문자열 | 파일 이름 |  |
| app\_id | 문자열 | 앱 ID | 예: Windows Defender |
| file\_ctime | 날짜 | 점프 파일 생성 일시 |  |
| file\_mtime | 날짜 | 점프 파일 수정 일시 |  |
| file\_atime | 날짜 | 점프 파일 파일 접근 일시 |  |
| target\_file\_size | 64비트 정수 | 대상 파일 크기 |  |
| target\_file\_attrs | 배열 | 대상 파일 속성 목록 | 예: FILE\_ATTRIBUTE\_ARCHIVE |
| target\_file\_ctime | 날짜 | 대상 파일 생성 일시 |  |
| target\_file\_mtime | 날짜 | 대상 파일 수정 일시 |  |
| target\_file\_atime | 날짜 | 대상 파일 접근 일시 |  |
| drive\_serial | 64비트 정수 | 대상 드라이브 시리얼 |  |
| drive\_type | 문자열 | 대상 드라이브 유형 | 예: DRIVE\_FIXED, DRIVE\_REMOVABLE, DRIVE\_CDROM |
| volume\_label | 문자열 | 대상 볼륨 이름 | 예: OS |
| local\_path | 문자열 | 대상 로컬 경로 | 예: C:\Program Files\Internet Explorer\iexplore.exe |
| net\_name | 문자열 | 대상 네트워크 공유 이름 | 예: \FILE\SAMPLE |
| common\_path\_suffix | 문자열 | 대상 네트워크 공유 경로 |  |
| show\_window | 문자열 | 윈도우 표시 옵션 | 예: SHOW, SHOW\_NORMAL, SHOW\_MAXIMIZED |
| shortcut\_name | 문자열 | 바로가기 이름 | 예: Run PuTTYgen |
| working\_dir | 문자열 | 작업 디렉터리 경로 |  |
| relative\_path | 문자열 | 상대 경로 |  |
| cmd\_args | 문자열 | 명령줄 인자 | 예: -newtab |
| icon\_location | 문자열 | 아이콘 경로 | 예: C:\Windows\SYSTEM32\IEFRAME.dll |
| hot\_key | 문자열 | 단축키 | 예: CTRL + ALT + R |
| target\_id\_list | 배열 | 대상 ID 목록 |  |

### evtx-file

EVTX 윈도우 이벤트 로그 파일에서 이벤트 채널, 이벤트 공급자, 이벤트 ID, 이벤트 작업 등의 정보를 조회합니다.

#### 문법

evtx-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

EVTX 윈도우 이벤트 로그 파일의 경로. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\evtx\*.evtx) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 EVTX 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로.

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 이벤트 발생 시각 |
| computer | 문자열 | 컴퓨터명 |
| channel | 문자열 | 이벤트 채널 |
| provider | 문자열 | 이벤트 공급자 |
| event\_id | 정수 | 이벤트 ID |
| task | 정수 | 이벤트 작업 |
| level | 정수 | 이벤트 레벨 |
| record\_id | 정수 | 레코드 ID |
| msg | 문자열 | 이벤트 메세지 |
| event\_data | 맵 | 이벤트 데이터 |

#### 사용 예

파일 경로를 입력하여 조회

evtx-file D:\data\evtx\System.evtx

zippath으로 지정한 ZIP 파일 안에 있는 EVTX 이벤트 파일을 조회

evtx-file zippath=D:\data\evtx.zip evtx\System.evtx

이벤트 공급자가 MySQL인 이벤트 조회

evtx-file D:\data\evtx\application.evtx | search provider=="MySQL"

EVTX\_WHITE 메시지 패턴과 일치하지 않는 이벤트 목록만 조회

evtx-file D:\data\evtx\application.evtx | mpsearch msg [ lookuptable EVTX\_WHITE ] | search len(\_mp\_result) == 0

### hive-file

레지스트리 하이브 파일에서 계정 및 그룹, 보안 정책, OS 정보, USB 장치, 프로그램 사용 내역 등의 정보를 조회합니다.

#### 문법

hive-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

레지스트리 파일의 경로. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\registry\*) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 레지스트리 하이브 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

하이브 파일과 출력되는 필드 내용은 다음 표를 참조하십시오.

하이브 파일

|  |  |  |  |
| --- | --- | --- | --- |
| 파일 | 용도 | 레지스트리 경로 | 추출 정보 |
| SAM | 계정 및 접속 기록 | HKEY\_LOCAL\_MACHINE\SAM | 계정, 그룹 |
| SECURITY | 보안 정책 및 권한 | HKEY\_LOCAL\_MACHINE\Security | 보안 정책 |
| SOFTWARE | 설치 프로그램 | HKEY\_LOCAL\_MACHINE\Software | OS 버전, OS 설치일, OS 설치 디렉터리, 소유자 계정 |
| SYSTEM | 시스템 설정 | HKEY\_LOCAL\_MACHINE\System | 호스트명, 시간대, 시스템 종료시각, USB 장치 등 |
| NTUSER.DAT | 사용자 설정 | HKEY\_USERS\.DEFAULT | 열어본 파일 목록 |

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| key | 문자열 | 서브 키 |
| type | 문자열 | 타입 |
| name | 문자열 | 레지스트리 이름 |
| value | 객체 | 레지스트리 데이터 |
| last\_written | 날짜 | 마지막으로 쓰여진 시간 |

#### 사용 예

파일 경로를 입력하여 조회

hive-file D:\data\registry\SYSTEM

zippath 옵션을 입력한 경우, 조회

hive-file zippath=D:\data\registry.zip registry\SYSTEM

윈도우 OS 정보 확인

hive-file D:\data\registry\SOFTWARE | search key=="ROOT\\Microsoft\\Windows NT\\CurrentVersion"

### lnk-file

윈도우 바로 가기 파일 정보를 추출합니다.

#### 문법

lnk-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

바로 가기 파일 경로. 일반적으로 바탕 화면이나 최근 문서 폴더에서 바로 가기 (.lnk 확장자) 파일을 찾을 수 있습니다.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_file | 문자열 | 파일 이름 |  |
| file\_ctime | 날짜 | 점프 파일 생성 일시 |  |
| file\_mtime | 날짜 | 점프 파일 수정 일시 |  |
| file\_atime | 날짜 | 점프 파일 파일 접근 일시 |  |
| target\_file\_size | 64비트 정수 | 대상 파일 크기 |  |
| target\_file\_attrs | 배열 | 대상 파일 속성 목록 | 예: FILE\_ATTRIBUTE\_ARCHIVE |
| target\_file\_ctime | 날짜 | 대상 파일 생성 일시 |  |
| target\_file\_mtime | 날짜 | 대상 파일 수정 일시 |  |
| target\_file\_atime | 날짜 | 대상 파일 접근 일시 |  |
| drive\_serial | 64비트 정수 | 대상 드라이브 시리얼 |  |
| drive\_type | 문자열 | 대상 드라이브 유형 | 예: DRIVE\_FIXED, DRIVE\_REMOVABLE, DRIVE\_CDROM |
| volume\_label | 문자열 | 대상 볼륨 이름 | 예: OS |
| local\_path | 문자열 | 대상 로컬 경로 | 예: C:\Program Files\Internet Explorer\iexplore.exe |
| net\_name | 문자열 | 대상 네트워크 공유 이름 | 예: \FILE\SAMPLE |
| common\_path\_suffix | 문자열 | 대상 네트워크 공유 경로 |  |
| show\_window | 문자열 | 윈도우 표시 옵션 | 예: SHOW, SHOW\_NORMAL, SHOW\_MAXIMIZED |
| shortcut\_name | 문자열 | 바로가기 이름 | 예: Run PuTTYgen |
| working\_dir | 문자열 | 작업 디렉터리 경로 |  |
| relative\_path | 문자열 | 상대 경로 |  |
| cmd\_args | 문자열 | 명령줄 인자 | 예: -newtab |
| icon\_location | 문자열 | 아이콘 경로 | 예: C:\Windows\SYSTEM32\IEFRAME.dll |
| hot\_key | 문자열 | 단축키 | 예: CTRL + ALT + R |
| target\_id\_list | 배열 | 대상 ID 목록 |  |

#### 사용 예

최근 문서 폴더의 모든 바로 가기 파일 조회

lnk-file C:\Users\Logpresso\AppData\Roaming\Microsoft\Windows\Recent\\*.lnk

### ntfs-logfile

NTFS 트랜잭션 로그 파일에서 파일 이름, 생성/수정/접근일시, Redo/Undo 작업 유형을 조회합니다. 이 명령으로 파일 생성, 삭제, 이름 변경 등의 내역을 확인할 수 있습니다.

#### 문법

ntfs-logfile [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

NTFS 트랜잭션 로그 파일의 경로. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\NTFS\*) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 NTFS 트랜잭션 로그 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| access\_at | 날짜 | 접근일시 |
| client\_data\_length | 정수 | 레코드 크기 |
| created\_at | 날짜 | 생성일시 |
| file\_name | 문자열 | 파일 이름 |
| flags | 정수 | 플래그 |
| lsn | 정수 | 로그파일 시퀀스 넘버 |
| mft\_head | 문자열 | mft 헤더 |
| mft\_link\_count | 정수 | 해당 파일을 참조하는 엔트리 갯수 |
| mft\_lsn | 정수 | mft 로그파일 시퀀스 넘버 |
| modified\_at | 날짜 | 수정일시 |
| page | 정수 | 페이지 번호 |
| prev\_lsn | 정수 | 이전 로그파일 시퀀스 넘버 |
| record\_type | 정수 | 레코드 타입 (2=체크포인트 레코드, 1=그외 레코드) |
| redo\_len | 정수 | redo 데이터 크기 |
| redo\_offset | 정수 | redo 데이터 시작 |
| redo\_op | 문자열 | redo 작업 |
| undo\_len | 정수 | undo 데이터 크기 |
| undo\_offset | 정수 | undo 데이터 시작 |
| undo\_op | 문자열 | undo 작업 |

Redo\_op, Undo\_op 작업 코드

redo\_op와 undo\_op 필드에 출력되는 작업 코드는 다음 표를 참조하십시오.

|  |  |
| --- | --- |
| `redo\_op`/`undo\_op` | 16진수 값 |
| noop | 0x00 |
| compensation\_log\_record | 0x1 |
| initialize\_file\_record\_segment | 0x2 |
| deallocate\_file\_record\_segment | 0x3 |
| write\_end\_of\_file\_record\_segment | 0x4 |
| create\_attribute | 0x5 |
| delete\_attribute | 0x6 |
| update\_resident\_value | 0x7 |
| update\_non\_resident\_value | 0x8 |
| update\_mapping\_pairs | 0x9 |
| delete\_dirty\_clusters | 0xa |
| set\_new\_attribute\_size | 0xb |
| add\_index\_entry\_root | 0xc |
| delete\_index\_entry\_root | 0xd |
| add\_index\_entry\_allocation | 0xe |
| delete\_index\_entry\_allocation | 0xf |
| set\_index\_entry\_ven\_allocation | 0x12 |
| update\_file\_name\_root | 0x13 |
| update\_file\_name\_allocation | 0x14 |
| set\_bits\_in\_non\_resident\_bitmap | 0x15 |
| clear\_bits\_in\_non\_resident\_bitmap | 0x16 |
| prepare\_transaction | 0x19 |
| commit\_transaction | 0x1a |
| forget\_transaction | 0x1b |
| open\_non\_resident\_attribute | 0x1c |
| open\_attribute\_table\_dump | 0x1d |
| dirty\_page\_table\_dump | 0x1f |
| transaction\_table\_dump | 0x20 |
| update\_record\_data\_root | 0x21 |

#### 사용 예

파일 경로를 입력하여 조회

ntfs-logfile D:\data\NTFS\test\_LogFile

zippath으로 지정한 ZIP 파일 안에 있는 NTFS 트랜잭션 로그 파일을 조회

ntfs-logfile zippath=D:\data\NTFS.zip NTFS\test\_LogFile

initialize\_file\_record\_segment이거나, delete를 포함하는 redo\_op 로그 조회

ntfs-logfile D:\data\NTFS\test\_LogFile | sort lsn | search redo\_op == "initialize\_file\_record\_segment" or redo\_op == "\*delete\*"

### ntfs-mft

NTFS 마스터 파일에서 파일 경로 및 이름, 파일 크기, 디스크 할당 크기, 파일 생성/수정/접근일시, 디렉터리 여부 등의 정보를 조회합니다. 조회한 데이터로 전체 파일 및 폴더 구조를 분석하거나 삭제된 파일이나 폴더를 추출할 수 있고, ADS(Alternate Data Stream) 은닉 정보를 탐색할 수 있습니다.

#### 문법

ntfs-mft [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

NTFS 마스터 파일의 경로. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\NTFS\*) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 NTFS 마스터 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| no | 정수 | 파일 번호 |
| file\_name | 문자열 | 파일 이름 |
| file\_path | 문자열 | 파일 경로 |
| file\_size | 정수 | 파일 크기 |
| alloc\_size | 정수 | 할당 크기 |
| in\_use | 불리언 | 사용 여부 |
| is\_dir | 불리언 | 디렉터리 여부 |
| link\_count | 정수 | 해당 파일을 참조하는 하드링크 갯수 |
| created\_at | 날짜 | $FILE\_NAME 속성의 생성일시 |
| modified\_at | 날짜 | $FILE\_NAME 속성의 수정일시 |
| access\_at | 날짜 | $FILE\_NAME 속성의 접근일시 |
| mft\_modified\_at | 날짜 | $FILE\_NAME 속성의 MFT 수정일시 |
| std\_created\_at | 날짜 | $STANDARD\_INFORMATION 속성의 생성일시 |
| std\_modified\_at | 날짜 | $STANDARD\_INFORMATION 속성의 수정일시 |
| std\_access\_at | 날짜 | $STANDARD\_INFORMATION 속성의접근일시 |
| std\_mft\_modified\_at | 날짜 | $STANDARD\_INFORMATION 속성의 MFT 수정일시 |
| is\_readonly | 불리언 | 읽기전용 여부 |
| is\_hidden | 불리언 | 숨김 여부 |
| is\_system | 불리언 | 시스템 여부 |
| is\_archive | 불리언 | 아카이브 여부 |
| is\_device | 불리언 | 장치 여부 |
| is\_normal | 불리언 | 일반 여부 |
| is\_temp | 불리언 | 임시 여부 |
| is\_sparse | 불리언 | sparse 파일 여부 |
| is\_reparse | 불리언 | reparse point 여부 |
| is\_compressed | 불리언 | 압축 여부 |
| is\_offline | 불리언 | 오프라인 여부 |
| is\_indexed | 불리언 | 인덱스 여부 |
| is\_encrypted | 불리언 | 암호화 여부 |
| lsn | 정수 | 로그 시퀀스 넘버 |
| seq | 정수 | 시퀀스 |
| file\_ref | 정수 | 파일 레퍼런스 |
| parent\_file\_ref | 정수 | 부모 파일 레퍼런스 |
| parent\_no | 정수 | 부모 파일 번호 |

#### 사용 예

파일 경로를 입력하여 조회

ntfs-mft D:\data\NTFS\test\_MFT

zippath 옵션으로 지정한 ZIP 파일에서 NTFS 마스터 파일을 조회

ntfs-mft zippath=D:\data\NTFS.zip NTFS\test\_MFT

삭제된 숨김 속성 파일 목록 조회

ntfs-mft D:\data\NTFS\test\_MFT | search not(in\_use) and not(is\_dir) and is\_hidden

### ntfs-usnjrnl

USN 저널 파일에서 이벤트 발생 시각, 파일 경로 및 이름, 파일 생성/삭제 등의 작업 정보를 조회합니다. 조회한 데이터로 MFT 파일과 조인하여 타임라인 분석이 가능합니다.

#### 문법

ntfs-usnjrnl [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

USN 저널 파일의 경로. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\NTFS\*) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 USN 저널 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 이벤트 발생 시각 |
| file\_name | 문자열 | 파일 이름 |
| file\_no | 정수 | 파일 번호 |
| file\_ref | 정수 | 파일 레퍼런스 |
| parent\_file\_no | 정수 | 부모 파일 번호 |
| parent\_file\_ref | 정수 | 부모 파일 레퍼런스 |
| reason | 리스트 | 이벤트 동작. 플래그 정보는 아래 표 참고 |
| usn | 정수 | 업데이트 시퀀스 넘버 |

플래그 정보

reason 필드에 출력되는 플래그 정보는 내용은 다음 표를 참조하십시오(USN\_RECORD\_V3 structure (winioctl.h), <https://docs.microsoft.com/ko-kr/windows/win32/api/winioctl/ns-winioctl-usn_record_v2> 참조).

|  |  |  |
| --- | --- | --- |
| 플래그 | 16진수 값 | 설명 |
| DATA\_OVERWRITE | 0x00000001 | 기본 $DATA 속성에 데이터를 겹쳐씀 |
| DATA\_EXTEND | 0x00000002 | 기본 $DATA 속성에 데이터가 추가됨 |
| DATA\_TRUNCATION | 0x00000004 | 기본 $DATA 속성에 데이터가 줄어듬 |
| NAMED\_DATA\_OVERWRITE | 0x00000010 | 이름 있는 $DATA 속성에 데이터를 겹쳐씀 |
| NAMED\_DATA\_EXTEND | 0x00000020 | 이름 있는 $DATA 속성에 데이터가 추가됨 |
| NAMED\_DATA\_TRUNCATION | 0x00000040 | 이름 있는 $DATA 속성에 데이터가 줄어듬 |
| FILE\_CREATE | 0x00000100 | 파일이나 디렉터리가 생성됨 |
| FILE\_DELETE | 0x00000200 | 파일이나 디렉터리가 삭제됨 |
| EA\_CHANGE | 0x00000400 | 파일의 확장된 속성이 변경됨 |
| SECURITY\_CHANGE | 0x00000800 | 접근 권한이 변경됨 |
| RENAME\_OLD\_NAME | 0x00001000 | 객체명 변경 시, 변경 전 이름 |
| RENAME\_NEW\_NAME | 0x00002000 | 객체명 변경 시, 변경 후 이름 |
| INDEXABLE\_CHANGE | 0x00004000 | 인덱스 상태가 변경됨 |
| BASIC\_INFO\_CHANGE | 0x00008000 | 하나 이상의 파일/디렉터리 속성이나 타임스탬프가 변경됨 |
| HARD\_LINK\_CHANGE | 0x00010000 | 하드 링크가 생성되었거나 삭제됨 |
| COMPRESSION\_CHANGE | 0x00020000 | 압축 상태가 변경됨(압축 또는 압축 해제) |
| ENCRYPTION\_CHANGE | 0x00040000 | 암호화 상태가 변경됨(암호화 또는 복호화) |
| OBJECT\_ID\_CHANGE | 0x00080000 | 객체 ID가 변경됨 |
| REPARSE\_POINT\_CHANGE | 0x00100000 | Reparse point가 변경됨 |
| STREAM\_CHANGE | 0x00200000 | 이름 있는 $DATA 속성의 생성, 삭제, 또는 변경됨 |
| TRANSACTED\_CHANGE | 0x00400000 | 주어진 스트림이 TxF 트랜잭션 커밋을 통해 수정됨 |
| INTEGRITY\_CHANGE | 0x00800000 | 무결성 설정이 변경됨 |
| CLOSE | 0x80000000 | 파일/디렉터리를 닫음 |

#### 사용 예

파일 경로를 입력하여 조회

ntfs-usnjrnl D:\data\NTFS\test\_UsnJrnl

zippath 옵션으로 지정한 ZIP 파일 안에 있는 USN 저널 파일을 조회

ntfs-usnjrnl zippath=D:\data\NTFS.zip NTFS\test\_UsnJrnl

실행 파일을 삭제한 이력 조회

ntfs-usnjrnl D:\data\NTFS\test\_UsnJrnl | search file\_name == "\*.exe" and string(reason) == "\*DELETE\*"

NTFS 마스터 파일과 조인하여 타임라인 분석

ntfs-usnjrnl D:\data\NTFS\test\_UsnJrnl | streamjoin type=left file\_no [ ntfs-mft D:\data\NTFS\test\_MFT | rename no as file\_no | fields file\_no, file\_path, in\_use, is\_dir ] | eval reason = strjoin(" | ", reason) | fields \_time, file\_path, reason, in\_use, is\_dir

### prefetch-file

윈도우 프리페치 파일에서 프로그램 실행 흔적을 추출합니다.

#### 문법

prefetch-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

PF 파일 경로. 일반적으로 프리페치 파일은 C:\Windows\Prefetch 경로에 생성됩니다.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_file | 문자열 | PF 파일 이름 | 예: CMD.EXE-0BD30981.pf |
| directories | 배열 | 디렉터리 경로 목록 |  |
| executable\_name | 문자열 | 실행 파일 이름 | 예: CMD.EXE |
| file\_size | 64비트 정수 | 실행 파일 크기 |  |
| hash | 문자열 | 실행 파일 경로 해시 | 예: bd30981 |
| last\_run\_time | 날짜 | 최근 실행 시각 |  |
| loaded\_files | 배열 | DLL 파일 경로 목록 |  |
| prev\_run\_times | 배열 | 이전 실행 시각 목록 | 최대 8개까지 저장됨 |
| run\_count | 32비트 정수 | 실행 횟수 |  |
| version | 32비트 정수 | 프리페치 파일 버전 | 예: 30 |

### reg-bam-entries

윈도우 10 (빌드 1709 이상)의 BAM (Background Activity Moderator)이 SYSTEM 레지스트리 하이브 파일에 기록한 데이터에서 계정별 프로그램 실행 흔적을 추출합니다.

#### 문법

reg-bam-entries [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

레지스트리 파일 경로. 파일 경로에 와일드카드(\*)를 사용하면 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 레지스트리 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_file | 문자열 | 레지스트리 하이브 파일 이름 | 예: SYSTEM |
| sid | 문자열 | 프로그램을 실행한 계정의 SID | 예: S-1-5-18 |
| file\_name | 문자열 | 실행 파일 이름 | 예: cmd.exe |
| file\_path | 문자열 | 실행 파일 경로 | 예: \Device\HarddiskVolume4\Windows\System32\cmd.exe |
| last\_execution | 날짜 | 최근 실행 일시 |  |

#### 사용 예

윈도우 10의 SYSTEM 하이브 파일에서 BAM 아티팩트 조회

reg-bam-entries D:\data\registry\SYSTEM

### reg-network-profiles

SOFTWARE 레지스트리 하이브 파일에서 네트워크 프로필 목록을 조회합니다.

#### 문법

reg-network-profiles [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

레지스트리 파일 경로. 파일 경로에 와일드카드(\*)를 사용하면 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 레지스트리 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_file | 문자열 | 하이브 파일 이름 | 예: SOFTWARE |
| guid | 문자열 | 프로필 GUID |  |
| profile\_name | 문자열 | 프로필 이름 | 예: iPhone |
| description | 문자열 | 설명 | 예: iPhone |
| category | 문자열 | 분류 | 예: Public, Private, Domain Authenticted |
| category\_id | 32비트 정수 | 분류 ID | 예: 0 (Public), 1 (Private), 2 (Domain) |
| name\_type | 문자열 | 이름 유형 | 예: Wired Network, Wireless Network |
| name\_type\_id | 32비트 정수 | 이름 유형 ID | 예: 6 (Wired), 71 (Wireless) |
| date\_created | 날짜 | 생성 일시 |  |
| date\_last\_connected | 날짜 | 최근 접속 일시 |  |
| managed | 불리언 | 자동 관리 여부 | 예: 1 (윈도우 관리), 0 (사용자 관리) |

### reg-opensave-files

탐색기 공용 대화 상자를 통해 최근에 열거나 저장된 파일 정보, 웹 브라우저와 애플리케이션으로 열거나 저장한 파일의 정보를 레지스트리 파일에서 조회합니다. 조회한 데이터로 사용자가 최근에 열거나 저장한 파일을 확인할 수 있습니다.

#### 문법

reg-opensave-files [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

레지스트리 파일 경로. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\registry\*.DAT) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 DAT 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_path | 문자열 | 파일 경로 |
| file\_ext | 문자열 | 파일 확장자 |
| file\_size | 문자열 | 파일 크기 |
| access\_at | 날짜 | 접근일시 |
| created\_at | 날짜 | 생성일시 |
| modified\_at | 날짜 | 수정일시 |
| mft\_entry\_index | 바이너리 | MFT 엔트리 인덱스 |
| ntfs\_seq | 정수 | NTFS 시퀀스 |
| last\_written | 날짜 | 마지막으로 쓰여진 시간 |
| order | 정수 | 확장자별 파일 순서 |

#### 사용 예

파일 경로를 입력하여 조회

reg-opensave-files D:\data\registry\NTUSER.DAT

zippath 옵션으로 지정한 ZIP 파일 안에 있는 DAT 파일을 조회

reg-opensave-files zippath=D:\data\registry.zip registry\NTUSER.DAT

파일 확장자별 순서 정렬

reg-opensave-files D:\data\registry\NTUSER.DAT | sort file\_ext, order

### reg-recent-docs

레지스트리 파일에서 사용자가 탐색기를 통해서 열거나 실행한 파일, 폴더 정보를 조회합니다. 조회한 데이터는 사용자가 열거나 실행한 파일, 폴더 정보 및 문서, 폴더 실행 유무를 확인할 수 있습니다. 또한, 사용자의 행위 파악를 파악하는데 사용할 수 있습니다.

#### 문법

reg-recent-docs [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

레지스트리 파일 경로. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\registry\*.DAT) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 레지스트리 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_name | 문자열 | 파일 이름 |
| file\_ext | 문자열 | 파일 확장자 |
| last\_written | 날짜 | 마지막으로 쓰여진 시간 |
| order | 정수 | 확장자별 파일 순서 |

#### 사용 예

파일 경로를 입력하여 조회

reg-recent-docs D:\data\registry\NTUSER.DAT

zippath 옵션을 입력한 경우, 조회

reg-recent-docs zippath=D:\data\registry.zip registry\NTUSER.DAT

파일 확장자별 순서 정렬

reg-recent-docs D:\data\registry\NTUSER.DAT | sort file\_ext, order

### reg-shellbags

레지스트리 파일에서 사용자가 로컬, 네트워크 및 이동식 저장 장치에서 접근한 폴더 정보를 조회합니다. 조회한 데이터는 사용자가 특정 폴더에 접근한 시간 정보 확인, 존재하는 폴더의 삭제/덮어쓰기에 대한 증거 조사, 탐색기를 통한 폴더 접근에 대한 MAC 타임(Modification, Access, Metadata Change) 추적에 사용할 수 있습니다.

#### 문법

reg-shellbags [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

레지스트리 파일 경로를 입력합니다. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\registry\*) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 레지스트리 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_name | 문자열 | 파일 이름 |
| file\_ext | 문자열 | 파일 확장자 |
| last\_written | 날짜 | 마지막으로 쓰여진 시간 |
| order | 정수 | 확장자별 파일 순서 |

#### 사용 예

파일 경로를 입력하여 조회

reg-shellbags D:\data\registry\NTUSER.DAT

zippath 옵션으로 지정한 ZIP 압축 파일 안에 있는 레지스트리 파일을 조회

reg-shellbags zippath=D:\data\registry.zip registry\NTUSER.DAT

파일 확장자별 순서 정렬

reg-shellbags D:\data\registry\NTUSER.DAT | sort file\_ext, order

### reg-shim-cache

레지스트리 파일에서 AppCompatCache 키(HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache\AppCompatCache)에 저장된 모든 실행 파일의 경로, 크기, 마지막 실행 시간 등의 정보를 조회합니다. 조회한 데이터는 실행 파일의 이름, 경로, 크기 정보 및 마지막 실행 시간을 확인할 수 있고, 침해사고 분석에 활용할 수 있습니다.

#### 문법

reg-shim-cache [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

레지스트리 파일 경로를 입력합니다. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\registry\*) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_path | 문자열 | 실행 파일 경로 |
| modified\_at | 날짜 | 수정일시 |

#### 사용 예

파일 경로를 입력하여 조회

reg-shim-cache D:\data\registry\SYSTEM

zippath 옵션을 입력한 경우, 조회

reg-shim-cache zippath=D:\data\registry.zip registry\SYSTEM

### reg-user-assists

레지스트리 파일에서 최근에 실행한 프로그램 목록, 마지막 실행 시간, 실행 횟수 등의 정보를 조회합니다. 조회한 데이터는 최근 실행한 응용프로그램 이름, 목록을 확인할 수 있고, 마지막으로 실행한 응용프로그램 시간과 실행 횟수를 분석에 활용할 수 있습니다.

#### 문법

reg-user-assists [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

레지스트리 파일 경로를 입력합니다. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\registry\*) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| key | 문자열 | 실행 파일 경로 |
| session\_num | 정수 | 세션 번호 |
| exec\_count | 정수 | 실행 횟수 |
| focus\_time | 정수 | 활성화된 시간 |
| last\_execution | 날짜 | 마지막으로 실행된 시간 |
| last\_written | 날짜 | 마지막으로 쓰여진 시간 |

#### 사용 예

파일 경로를 입력하여 조회

reg-user-assists D:\data\registry\NTUSER.DAT

zippath 옵션을 입력한 경우, 조회

reg-user-assists zippath=D:\data\registry.zip registry\NTUSER.DAT

### recent-file-cache

RecentFileCache.bcf 파일에 저장된 프로그램 경로 목록을 추출합니다.

#### 문법

recent-file-cache [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

RecentFileCache.bcf 파일 경로. 윈도우 7의 C:\Windows\AppCompat\Programs 위치에 있습니다.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_file | 문자열 | BCF 파일 이름 |  |
| file\_path | 문자열 | 실행 프로그램 파일 경로 |  |

### srum-application-resource-usages

SRUM (System Resource Utilization Monitor) 데이터베이스 파일에서 응용 프로그램 리소스 사용 기록을 추출합니다.

#### 문법

srum-application-resource-usages FILE\_PATH

필수 매개변수

**FILE\_PATH**

SRUDB.dat 파일 경로. 이 ESE 데이터베이스 파일은 일반적으로 C:\Windows\System32\sru\SRUDB.dat 위치에 있습니다.

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_time | 날짜 | 시각 |  |
| \_file | 문자열 | SRUDB.dat 파일 이름 |  |
| app\_id | 32비트 정수 | 앱 ID |  |
| app\_name | 문자열 | 앱 이름 | 예: \Device\HarddiskVolume3\Windows\explorer.exe |
| auto\_inc\_id | 32비트 정수 | 자동 증가 번호 |  |
| background\_bytes\_read | 64비트 정수 | 백그라운드 읽은 바이트 |  |
| background\_bytes\_written | 64비트 정수 | 백그라운드 기록 바이트 |  |
| background\_context\_switches | 32비트 정수 | 백그라운드 컨텍스트 스위치 |  |
| background\_cycle\_time | 64비트 정수 | 백그라운드 CPU 사이클 |  |
| background\_num\_read\_operations | 32비트 정수 | 백그라운드 읽기 작업 횟수 |  |
| background\_num\_write\_operations | 32비트 정수 | 백그라운드 쓰기 작업 횟수 |  |
| background\_number\_of\_flushes | 32비트 정수 | 백그라운드 플러시 횟수 |  |
| face\_time | 64비트 정수 | 실행 시간 (ns) |  |
| foreground\_bytes\_read | 64비트 정수 | 포어그라운드 읽은 바이트 |  |
| foreground\_bytes\_written | 64비트 정수 | 포어그라운드 기록 바이트 |  |
| foreground\_context\_switches | 32비트 정수 | 포어그라운드 컨텍스트 스위치 |  |
| foreground\_cycle\_time | 64비트 정수 | 포어그라운드 CPU 사이클 |  |
| foreground\_num\_read\_operations | 32비트 정수 | 포어그라운드 읽기 작업 횟수 |  |
| foreground\_num\_write\_operations | 32비트 정수 | 포어그라운드 쓰기 작업 횟수 |  |
| foreground\_number\_of\_flushes | 32비트 정수 | 포어그라운드 플러시 횟수 |  |
| sid | 문자열 | 프로그램 실행 계정의 SID | 예: S-1-5-18 |
| user\_id | 32비트 정수 | 계정 ID |  |

### srum-energy-usages

SRUM (System Resource Utilization Monitor) 데이터베이스 파일에서 에너지 사용 기록을 추출합니다.

#### 문법

srum-energy-usages FILE\_PATH

필수 매개변수

**FILE\_PATH**

SRUDB.dat 파일 경로. 이 ESE 데이터베이스 파일은 일반적으로 C:\Windows\System32\sru\SRUDB.dat 위치에 있습니다.

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_time | 날짜 | 시각 |  |
| \_file | 문자열 | SRUDB.dat 파일 이름 |  |
| app\_id | 32비트 정수 | 앱 ID | 모두 1 |
| auto\_inc\_id | 32비트 정수 | 자동 증가 번호 |  |
| charge\_level | 32비트 정수 | 배터리 충전 수준 |  |
| configuration\_hash | 64비트 정수 | 설정 해시 값 |  |
| cycle\_count | 32비트 정수 | 사이클 횟수 |  |
| designed\_capacity | 32비트 정수 | 배터리 설계 충전 용량 |  |
| event\_timestamp | 64비트 정수 | 윈도우 FILETIME 값 |  |
| full\_charged\_capacity | 32비트 정수 | 배터리 완전 충전 용량 | 설계 용량보다 작음 |
| state\_transition | 32비트 정수 | 상태 전이 |  |
| user\_id | 32비트 정수 | 계정 ID |  |

윈도우 FILETIME은 1601년 1월 1일 이후 경과한 나노초를 100으로 나눈 값입니다. FILETIME 값에서 116444736000000000를 빼고 10000으로 나누면 밀리초 단위의 유닉스 타임스탬프(epoch)로 변환할 수 있습니다.

### srum-long-term-energy-usages

SRUM (System Resource Utilization Monitor) 데이터베이스 파일에서 장기 에너지 사용 기록을 추출합니다.

#### 문법

srum-long-term-energy-usages FILE\_PATH

필수 매개변수

**FILE\_PATH**

SRUDB.dat 파일 경로. 이 ESE 데이터베이스 파일은 일반적으로 C:\Windows\System32\sru\SRUDB.dat 위치에 있습니다.

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_time | 날짜 | 시각 |  |
| \_file | 문자열 | SRUDB.dat 파일 이름 |  |
| active\_ac\_time | 32비트 정수 |  |  |
| active\_dc\_time | 32비트 정수 |  |  |
| active\_discharge\_time | 32비트 정수 |  |  |
| active\_energy | 32비트 정수 |  |  |
| app\_id | 32비트 정수 | 앱 ID | 모두 1 |
| auto\_inc\_id | 32비트 정수 | 자동 증가 번호 |  |
| configuration\_hash | 64비트 정수 |  |  |
| cs\_ac\_time | 32비트 정수 |  |  |
| cs\_dc\_time | 32비트 정수 |  |  |
| cs\_discharge\_time | 32비트 정수 |  |  |
| cs\_energy | 32비트 정수 |  |  |
| cycle\_count | 32비트 정수 |  |  |
| designed\_capacity | 32비트 정수 | 배터리 설계 용량 |  |
| full\_charged\_capacity | 32비트 정수 | 배터리 완전 충전 용량 | 설계 용량보다 작음 |
| user\_id | 32비트 정수 | 계정 ID |  |

### srum-network-connectivities

SRUM (System Resource Utilization Monitor) 데이터베이스 파일에서 네트워크 연결 기록을 추출합니다.

#### 문법

srum-network-connectivities FILE\_PATH

필수 매개변수

**FILE\_PATH**

SRUDB.dat 파일 경로. 이 ESE 데이터베이스 파일은 일반적으로 C:\Windows\System32\sru\SRUDB.dat 위치에 있습니다.

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_time | 날짜 | 시각 |  |
| \_file | 문자열 | SRUDB.dat 파일 이름 |  |
| app\_id | 32비트 정수 | 앱 ID | 모두 1 |
| auto\_inc\_id | 32비트 정수 | 자동 증가 번호 |  |
| connect\_start\_time | 날짜 | 네트워크 연결 시작 시각 |  |
| connected\_time | 32비트 정수 | 연결 유지 시간 |  |
| interface\_luid | 64비트 정수 | 네트워크 어댑터 식별자 (Local Unit ID) |  |
| l2\_profile\_flags | 32비트 정수 | L2 프로파일 플래그 |  |
| l2\_profile\_id | 32비트 정수 | L2 프로파일 ID |  |
| user\_id | 32비트 정수 | 계정 ID |  |

### srum-network-usages

SRUM (System Resource Utilization Monitor) 데이터베이스 파일에서 프로그램별 네트워크 통신 사용량을 추출합니다.

#### 문법

srum-network-usages FILE\_PATH

필수 매개변수

**FILE\_PATH**

SRUDB.dat 파일 경로. 이 ESE 데이터베이스 파일은 일반적으로 C:\Windows\System32\sru\SRUDB.dat 위치에 있습니다.

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_time | 날짜 | 시각 |  |
| \_file | 문자열 | SRUDB.dat 파일 이름 |  |
| app\_id | 32비트 정수 | 앱 ID |  |
| app\_name | 문자열 | 앱 이름 | 예: \device\harddiskvolume3\windows\system32\taskhostw.exe |
| auto\_inc\_id | 32비트 정수 | 자동 증가 번호 |  |
| bytes\_recvd | 64비트 정수 | 수신 바이트량 |  |
| bytes\_sent | 64비트 정수 | 송신 바이트량 |  |
| interface\_luid | 64비트 정수 | 네트워크 어댑터 식별자 (Local Unit ID) |  |
| l2\_profile\_flags | 32비트 정수 | L2 프로파일 플래그 |  |
| l2\_profile\_id | 32비트 정수 | L2 프로파일 ID |  |
| sid | 문자열 | SID |  |
| user\_id | 32비트 정수 | 계정 ID |  |

### srum-push-notifications

SRUM (System Resource Utilization Monitor) 데이터베이스 파일에서 윈도우 푸시 알림 내역을 추출합니다.

#### 문법

srum-network-usages FILE\_PATH

필수 매개변수

**FILE\_PATH**

SRUDB.dat 파일 경로. 이 ESE 데이터베이스 파일은 일반적으로 C:\Windows\System32\sru\SRUDB.dat 위치에 있습니다.

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_time | 날짜 | 시각 |  |
| \_file | 문자열 | SRUDB.dat 파일 이름 |  |
| app\_id | 32비트 정수 | 앱 ID |  |
| auto\_inc\_id | 32비트 정수 | 자동 증가 번호 |  |
| network\_type | 32비트 정수 | 네트워크 유형 | 예: 0 |
| notification\_type | 32비트 정수 | 알림 유형 | 예: 1, 4 |
| payload\_size | 32비트 정수 | 페이로드 크기 |  |
| sid | 문자열 | 프로그램 실행 계정의 SID |  |
| user\_id | 32비트 정수 | 계정 ID |  |

### timeline-activities

윈도우 10의 ActivitiesCache.db 파일에서 프로그램 실행 이력을 추출합니다.

#### 문법

timeline-activities FILE\_PATH

필수 매개변수

**FILE\_PATH**

ActivitiesCache.db 파일 경로. 이 파일은 C:\Users<username>\AppData\Local\ConnectedDevicesPlatform 디렉터리 내부에 생성됩니다.

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| id | 바이너리 | 액티비티 ID |  |
| executable | 문자열 | 실행 파일 경로 | 예: {1AC14E77-02E7-4E5D-B744-2EB1AE5198B7}\cmd.exe |
| activity\_type | 문자열 | 작업 유형 | 예: ExecuteOpen, InFocus, CopyPaste |
| payload\_app\_display\_name | 문자열 | 앱 표시 이름 | 예: 명령 프롬프트 |
| payload\_display\_text | 문자열 | 표시 텍스트 | 예: 명령 프롬프트 |
| payload\_description | 문자열 | 설명 |  |
| payload\_active\_duration | 32비트 정수 | 활성화 기간 | 초 단위 |
| start\_time | 날짜 | 활성 시작 시각 |  |
| end\_time | 날짜 | 활성 종료 시각 |  |
| last\_modified\_time | 날짜 | 최종 변경 시각 |  |
| last\_modified\_on\_client | 날짜 | 클라이언트의 최종 변경 시각 |  |
| expiration\_time | 날짜 | 만료 시각 |  |
| priority | 32비트 정수 | 중요도 | 예: 1, 3 |
| activity\_status | 32비트 정수 | 액티비티 상태 | 예: 1 |
| user\_action\_state | 32비트 정수 | 사용자 액션 상태 | 예: 0 |
| created\_in\_cloud | 32비트 정수 | 클라우드에서 생성 여부 | 예: 1, 0 |
| is\_read | 32비트 정수 | 읽기 여부 | 예: 1, 0 |
| is\_local\_only | 32비트 정수 | 로컬 전용 여부 | 예: 1, 0 |
| etag | 32비트 정수 | ETag |  |
| group\_app\_activity\_id | 문자열 | 그룹 앱 액티비티 ID |  |
| app\_activity\_id | 문자열 | 앱 액티비티 ID | GUID\경로 형식 |
| parent\_activity\_id | 바이너리 | 부모 액티비티 ID |  |
| package\_id\_hash | 문자열 | 패키지 ID 해시 |  |
| platform\_device\_id | 문자열 | 플랫폼 장치 ID |  |
| payload\_activation\_uri | 문자열 | 페이로드 활성화 URI |  |
| payload\_background\_color | 문자열 | 페이로드 배경색 | 예: light, black |
| payload\_content\_uri | 문자열 | 페이로드 컨텐츠 경로 |  |
| payload\_type | 문자열 | 페이로드 유형 | 예: UserEngaged |
| payload\_user\_timezone | 문자열 | 페이로드 시간대 | 예: Asia/Seoul |
| payload\_reporting\_app | 문자열 | 페이로드 리포팅 앱 | 예: ShellActivityMonitor |
| clipboard\_payload | 문자열 | 클립보드 페이로드 |  |
| tag | 문자열 | 태그 |  |

#### 사용 예

윈도우 10 타임라인 아티팩트 조회

timeline-activities C:\Users\Logpresso\AppData\Local\ConnectedDevicesPlatform\L.Logpresso\ActivitiesCache.db

### wer-file

윈도우 에러 리포팅 서비스가 기록한 오류 보고서 파일을 조회합니다.

#### 문법

wer-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

Report.wer 파일 경로. 일반적으로 WER 파일은 C:\ProgramData\Microsoft\Windows\WER\ReportArchive 경로에 생성됩니다.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| app\_name | 문자열 | 프로그램 이름 | 예: WinSCP: SFTP, FTP, WebDAV, S3 and SCP client |
| app\_path | 문자열 | 프로그램 경로 | 예: C:\Program Files (x86)\WinSCP\WinSCP.exe |
| app\_session\_guid | 문자열 | 세션 GUID |  |
| application\_identity | 문자열 | 프로그램 해시 | 예: 6CBDFE29B2B81F7344F51D705F87701B |
| boot\_id | 문자열 | 부트 ID |  |
| consent | 문자열 | 오류 보고서 전송 동의 여부 | 예: 1 |
| consent\_key | 문자열 | 오류 보고서 동의 키 | 예: APPCRASH |
| dynamic\_sigs | 맵 | 동적 예외 정보 |  |
| etw\_non\_collect\_reason | 문자열 | ETW 미수집 사유 | 예: 1 |
| event\_time | 문자열 | 이벤트 시각 | FILETIME 값. 예: 132993550101388746 |
| event\_type | 문자열 | 이벤트 유형 | 예: APPCRASH |
| friendly\_event\_name | 문자열 | 화면 표시용 이벤트 이름 | 예: 작동이 중지됨 |
| integrator\_report\_identifier | 문자열 | 보고 식별자 |  |
| is\_fatal | 문자열 | 치명적 여부 | 예: 1 |
| loaded\_modules | 배열 | 로드된 DLL 파일 경로 목록 |  |
| metadata\_hash | 문자열 | 메타데이터 해시 | 예: 1414078753 |
| ns\_app\_name | 문자열 | 프로그램 이름 | 예: WinSCP.exe |
| ns\_group | 문자열 | 프로그램 그룹 | 예: windows8 |
| ns\_partner | 문자열 | 프로그램 제조사 | 예: windows |
| original\_filename | 문자열 | 프로그램 파일 이름 | 예: winscp.exe |
| os\_infos | 맵 | 운영체제 정보 |  |
| report\_identifier | 문자열 | 보고서 GUID 식별자 |  |
| report\_status | 문자열 | 보고서 상태 | 예: 268435456 |
| report\_type | 문자열 | 보고서 유형 | 예: 2 |
| response\_bucket\_id | 문자열 | 응답 버킷 ID |  |
| response\_bucket\_table | 문자열 | 응답 버킷 테이블 | 예: 1 |
| response\_legacy\_bucket\_id | 문자열 | 응답 레거시 버킷 ID |  |
| responsetype | 문자열 | 응답 유형 | 예: 4 |
| service\_split | 문자열 |  | 예: 52166656 |
| sigs | 맵 | 예외 정보 | 응용 프로그램 버전, 예외 코드, 예외 오프셋 등 |
| states | 맵 | 상태 값 목록 |  |
| target\_app\_id | 문자열 | 대상 프로그램 ID |  |
| target\_app\_ver | 문자열 | 대상 프로그램 버전 | 예: 2021//10//11:13:40:38!19d8b46!WinSCP.exe |
| target\_as\_id | 문자열 |  | 예: 16815 |
| ui | 문자열 |  | 예: C:\Program Files (x86)\WinSCP\WinSCP.exe |
| upload\_time | 문자열 | 업로드 시각 | FILETIME 값. 예: 132993550104287215 |
| version | 문자열 | WER 포맷 버전 | 예: 1 |
| wow64\_guest | 문자열 |  | 예: 332 |
| wow64\_host | 문자열 |  | 예: 34404 |

### wmi-objects-data

WMI (Windows Management Instrumentation) OBJECTS.DATA 파일에 저장된 CIM 개체를 조회합니다.

#### 문법

wmi-objects-data [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

필수 매개변수

**FILE\_PATH**

OBJECTS.DATA 파일 경로. 이 파일은 C:\WINDOWS\system32\wbem\Repository\OBJECTS.DATA 경로에 위치합니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 OBJECTS.DATA 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_file | 문자열 | 저장소 파일 이름 | 예: OBJECTS.DATA |
| \_time | 날짜 | 시각 |  |
| record\_type | 문자열 | 레코드 유형 | 예: CLASS\_DEFINITION, INSTANCE\_OBJECT, REGISTRATION |
| class\_name | 문자열 | 클래스 이름 | 예: Win32\_PageFileSetting |
| date1 | 날짜 | 날짜 1 |  |
| date2 | 날짜 | 날짜 2 |  |
| hash | 문자열 | 해시 값 |  |
| index\_key | 문자열 | 인덱스 키 |  |
| instance\_name | 문자열 | 인스턴스 이름 | 예: Connection, provider |
| name | 문자열 | 이름 | 예: CIM\_Setting |
| properties | 맵 | 속성 목록 |  |
| qualifiers | 맵 | 한정자 목록 |  |
| strings | 배열 | 문자열 목록 |  |
| super\_class\_name | 문자열 | 상위 클래스 이름 | 예: \_\_SystemClass, \_\_Provider, \_\_Event |

#### 사용 예

WMI 서브스크립션을 통한 실행 탐지 (T1546.003)

wmi-objects-data C:\windows\system32\wbem\Repository\OBJECTS.DATA | eval strings = strjoin("\n", strings) | search in(strings, "\*\_\_EventFilter\*", "\*\_\_EventConsumer\*", "\*\_\_FilterToConsumerBinding\*")

### zipfile-entries

지정된 ZIP 파일 내부의 압축된 파일 및 디렉터리 목록을 가져옵니다.

#### 문법

zipfile-entries ZIPFILE\_PATH

필수 매개변수

**ZIPFILE\_PATH**

ZIP 파일의 절대 경로를 입력합니다. 파일 경로에 와일드카드(\*)를 사용하면 특정 문자열 패턴을 포함한 모든 파일 및 디렉터리를 한 번에 조회할 수 있습니다. 파일을 읽어오려면 로그프레소 실행 계정에 파일 접근 권한이 부여되어 있어야 합니다.

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| zip\_path | 문자열 | ZIP 파일 경로 |
| entry | 문자열 | ZIP 파일 목록 |
| file\_size | 정수 | 파일 크기 |
| compressed\_size | 정수 | 압축 크기 |
| modified\_at | 날짜 | 수정일시 |
| comment | 문자열 | 코멘트 |

#### 사용 예

D:\data 의 entry.zip 파일의 파일 및 디렉터리 목록 조회

zipfile-entries D:\data\entry.zip

D:\data 의 모든 ZIP 파일의 파일 및 디렉터리 목록 조회

zipfile-entries D:\data\\*.zip

entry.zip 파일에서 .evtx 확장자를 가진 파일 엔트리 조회

zipfile-entries D:\data\entry.zip | search entry=="\*.evtx"

## 리눅스 아티팩트

### linux-arp-entries

/proc/net/arp 파일에서 ARP 캐시 정보를 조회합니다.

#### 문법

linux-arp-entries

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| type | 문자열 | 하드웨어 유형. 대부분의 경우, 이더넷(1) |
| ip | IP 주소 | 연관된 IP 주소 |
| mac | 문자열 | MAC 주소 |
| iface | 문자열 | 네트워크 인터페이스 이름 |

#### 사용 예

ARP 캐시 정보를 조회

linux-arp-entries

### linux-connections

TCP/IP 네트워크 연결 정보를 조회합니다.

#### 문법

linux-connections

#### 설명

이 명령어는 다음 파일에서 TCP/IP 네트워크 연결 정보를 수집합니다:

* /proc/net/icmp
* /proc/net/icmp6
* /proc/net/raw
* /proc/net/raw6
* /proc/net/tcp
* /proc/net/tcp6
* /proc/net/udp
* /proc/net/udp6

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| pid | 정수 | 프로세스 ID |
| rx\_queue | 정수 | 수신 큐 길이 |
| tx\_queue | 정수 | 송신 큐 길이 |
| protocol | 문자열 | 프로토콜 |
| local\_ip | IP 주소 | 로컬 IP 주소 |
| local\_port | 정수 | 로컬 포트 번호 |
| remote\_ip | IP 주소 | 원격 IP 주소 |
| remote\_port | 정수 | 원격 포트 번호 |
| state | 문자열 | 상태 (LISTEN, ESTABLISHED, TIME\_WAIT 등) |

### linux-cron-jobs

예약된 Cron job을 조회합니다.

#### 문법

linux-cron-jobs

#### 설명

이 명령어는 다음 파일에서 예약된 작업 정보를 수집합니다:

* /var/cron/tabs/
* /var/spool/cron/
* /var/spool/cron/crontabs/

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| user | 문자열 | 사용자 ID |
| cron\_schedule | 문자열 | cron 일정 |
| cmd\_line | 문자열 | 실행할 명령 |

### linux-env

리눅스 시스템 환경 변수를 조회합니다.

#### 문법

linux-env

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| name | 문자열 | 환경 변수 이름 |
| value | 문자열 | 환경 변수 값 |

### linux-failed-logins

/var/log/btmp 파일에서 로그인 실패 이력을 조회합니다.

#### 문법

linux-failed-logins [ignore-error=BOOL]

선택 매개변수

**ignore-error=BOOL**

/var/log/btmp 파일을 읽을 수 없는 경우 오류 처리(기본값: f)

* t: 오류가 발생해도 정상 종료를 반환
* f: 오류가 발생하면 실패를 반환

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 로그인 실패 시각 |
| src\_ip | IPv4 주소 | 로그인을 시도한 원격지 IP 주소 |
| user | 문자열 | 사용자 ID |

### linux-hidden-files

/tmp, /dev, /home 및 하위 디렉터리에 숨겨진 파일 목록을 보여줍니다.

#### 문법

linux-hidden-files

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_path | 문자열 | 파일 경로 |
| file\_name | 문자열 | 파일 이름 |
| file\_type | 문자열 | 파일 유형 |
| permissions | 문자열 | 접근 권한 |
| file\_size | 정수 | 파일 크기 |
| file\_ctime | 날짜 | 파일 생성 시각 |
| file\_mtime | 날짜 | 파일 수정 시각 |
| file\_atime | 날짜 | 파일 접근 시각 |
| owner\_read | 불리언 | 소유자 읽기 권한 여부 |
| owner\_write | 불리언 | 소유자 쓰기 권한 여부 |
| owner\_execute | 불리언 | 소유자 실행 권한 여부 |
| group\_read | 불리언 | 그룹 읽기 권한 여부 |
| group\_write | 불리언 | 그룹 쓰기 권한 여부 |
| group\_execute | 불리언 | 그룹 실행 권한 여부 |
| others\_read | 불리언 | 기타 읽기 권한 여부 |
| others\_write | 불리언 | 기타 쓰기 권한 여부 |
| others\_execute | 불리언 | 기타 실행 권한 여부 |

### linux-logins

/var/log/wtmp 파일에서 사용자의 로그인 및 로그아웃 이력을 조회합니다.

#### 문법

linux-logins

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 로그인 시각 |
| src\_ip | IPv4 주소 | 접속 IP 주소 |
| host | 문자열 | 접속 호스트 이름 (IP 혹은 도메인) |
| pid | 정수 | 프로세스 ID |
| user | 문자열 | 로그인 계정 |
| tty | 문자열 | 터미널 |
| login\_time | 날짜 | 로그인 시각 |
| logout\_time | 날짜 | 로그아웃 시각 |

### linux-network-interfaces

네트워크 인터페이스 설정과 통계 정보를 조회합니다.

#### 문법

linux-network-interfaces

#### 설명

이 명령어는 /sys/class/net/의 인터페이스별 flags, statistics 디렉터리의 각 통계 값과 ip address show 실행 결과를 기반으로 네트워크 인터페이스의 설정 및 통계 정보를 수집합니다.

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| iface | 문자열 | 인터페이스 이름 |
| status | 문자열 | 활성 상태 |
| mtu | 32비트 정수 | 최대 전송 단위 |
| ip\_addr | IPv4 주소 | IPv4 주소 |
| netmask | IPv4 주소 | IPv4 넷마스크 |
| ip\_addr6 | IPv6 주소 | IPv6 주소 |
| prefix\_len | 32비트 정수 | IPv6 프리픽스 길이 |
| rx\_pkts | 64비트 정수 | 수신 패킷 수 |
| rx\_bytes | 64비트 정수 | 수신 바이트 수 |
| rx\_errors | 64비트 정수 | 수신 오류 (체크섬 불일치 등) |
| rx\_drops | 64비트 정수 | 수신 버림 (목적지 불일치 등) |
| rx\_overruns | 64비트 정수 | 수신 손실 (수신 큐 초과 등) |
| tx\_pkts | 64비트 정수 | 송신 패킷 수 |
| tx\_bytes | 64비트 정수 | 송신 바이트 수 |
| tx\_errors | 64비트 정수 | 오류가 발생해 보내지 않은 패킷 |
| tx\_drops | 64비트 정수 | 송신 버림 패킷 수 |
| tx\_overruns | 64비트 정수 | 송신 손실 패킷 수 |
| tx\_carrier | 64비트 정수 | 링크 단절 수 |
| tx\_collisions | 64비트 정수 | 충돌 패킷 수 |

### linux-no-owner-files

소유자가 없는 리눅스 파일을 조회합니다.

#### 문법

linux-no-owner-files

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_path | 문자열 | 파일 경로 |
| file\_name | 문자열 | 파일 이름 |
| file\_type | 문자열 | 파일 유형 |
| permissions | 문자열 | 접근 권한 |
| file\_size | 64비트 정수 | 파일 크기 |
| file\_ctime | 날짜 | 파일 생성 시각 |
| file\_mtime | 날짜 | 파일 수정 시각 |
| file\_atime | 날짜 | 파일 접근 시각 |
| owner\_read | 불리언 | 소유자 읽기 권한 여부 |
| owner\_write | 불리언 | 소유자 쓰기 권한 여부 |
| owner\_execute | 불리언 | 소유자 실행 권한 여부 |
| group\_read | 불리언 | 그룹 읽기 권한 여부 |
| group\_write | 불리언 | 그룹 쓰기 권한 여부 |
| group\_execute | 불리언 | 그룹 실행 권한 여부 |
| others\_read | 불리언 | 기타 읽기 권한 여부 |
| others\_write | 불리언 | 기타 쓰기 권한 여부 |
| others\_execute | 불리언 | 기타 실행 권한 여부 |

### linux-non-device-files

/dev 디렉터리에 있는 파일 중에서 장치 유형이 아닌 파일을 조회합니다.

#### 문법

linux-non-device-files

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_path | 문자열 | 파일 경로 |
| file\_name | 문자열 | 파일 이름 |
| file\_type | 문자열 | 파일 유형 |
| permissions | 문자열 | 접근 권한 |
| file\_size | 정수 | 파일 크기 |
| file\_ctime | 날짜 | 파일 생성 시각 |
| file\_mtime | 날짜 | 파일 수정 시각 |
| file\_atime | 날짜 | 파일 접근 시각 |
| owner\_read | 불리언 | 소유자 읽기 권한 여부 |
| owner\_write | 불리언 | 소유자 쓰기 권한 여부 |
| owner\_execute | 불리언 | 소유자 실행 권한 여부 |
| group\_read | 불리언 | 그룹 읽기 권한 여부 |
| group\_write | 불리언 | 그룹 쓰기 권한 여부 |
| group\_execute | 불리언 | 그룹 실행 권한 여부 |
| others\_read | 불리언 | 기타 읽기 권한 여부 |
| others\_write | 불리언 | 기타 쓰기 권한 여부 |
| others\_execute | 불리언 | 기타 실행 권한 여부 |

### linux-open-files

/proc/프로세스ID/fd 파일 목록을 수집하여 리눅스 프로세스별 열린 파일 목록을 조회합니다.

#### 문법

linux-open-files

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| pid | 정수 | 프로세스 ID |
| cmd\_line | 문자열 | 프로세스에서 실행하는 명령줄 |
| user | 문자열 | 사용자 ID |
| fd | 정수 | 파일 디스크립터 |
| type | 문자열 | 파일 유형 (REG = 일반 파일) |
| file\_size | 정수 | 파일 크기 |
| target | 문자열 | 파일 경로 혹은 소켓 정보 |

### linux-partitions

fdisk -l 명령 실행 결과를 파싱하여 리눅스 디스크 파티션 목록을 조회합니다.

#### 문법

linux-partitions

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| device | 문자열 | 스토리지 장치 이름 |
| start | 정수 | 섹터 시작 번호 |
| end | 정수 | 섹터 끝 번호 |
| sector\_count | 정수 | 전체 섹터 수 |
| total\_bytes | 정수 | 전체 파티션 크기 (바이트 단위) |
| type | 문자열 | 파티션 유형 |

### linux-pipes

find / -type p 명령 실행 결과를 파싱하여 파이프 파일 목록을 조회합니다.

#### 문법

linux-pipes

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_path | 문자열 | 파일 경로 |
| file\_name | 문자열 | 파일 이름 |
| file\_type | 문자열 | 파일 유형 |
| permissions | 문자열 | 접근 권한 |
| file\_size | 정수 | 파일 크기 |
| file\_ctime | 날짜 | 파일 생성 시각 |
| file\_mtime | 날짜 | 파일 수정 시각 |
| file\_atime | 날짜 | 파일 접근 시각 |
| owner\_read | 불리언 | 소유자 읽기 권한 여부 |
| owner\_write | 불리언 | 소유자 쓰기 권한 여부 |
| owner\_execute | 불리언 | 소유자 실행 권한 여부 |
| group\_read | 불리언 | 그룹 읽기 권한 여부 |
| group\_write | 불리언 | 그룹 쓰기 권한 여부 |
| group\_execute | 불리언 | 그룹 실행 권한 여부 |
| others\_read | 불리언 | 기타 읽기 권한 여부 |
| others\_write | 불리언 | 기타 쓰기 권한 여부 |
| others\_execute | 불리언 | 기타 실행 권한 여부 |

### linux-processes

/proc 파일 시스템에서 리눅스 프로세스 정보와 시스템 자원 사용량을 조회합니다.

#### 문법

linux-processes

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| user | 문자열 | 프로세스 실행 계정 |
| pid | 정수 | 프로세스 아이디 |
| ppid | 정수 | 부모 프로세스 아이디 |
| cpu\_usage | 64비트 실수 | CPU 사용율 (%) |
| mem\_usage | 64비트 실수 | 메모리 사용율 (%) |
| vsz | 32비트 정수 | 가상 메모리 크기(KiB) |
| rss | 32비트 정수 | 상주 메모리 크기(KiB) |
| status | 문자열 | 상태 |
| is\_deleted | 불리언 | 실행 파일 삭제 여부 |
| cmd\_line | 문자열 | 명령줄 |
| start\_time | 날짜 | 프로세스 시작 시각 |

### linux-recent-files

/ 이하의 디렉토리에서 리눅스에서 최근 생성되거나 수정된 파일 목록을 조회합니다. from이나 span 옵션을 주지 않으면 최근 1일간 생성되거나 수정된 파일 목록을 조회합니다.

#### 문법

linux-recent-files [OPTION]

선택 매개변수

**from=yyyyMMddHHmmss**

yyyyMMddHHmmss 포맷으로 범위의 시작을 지정합니다. 뒷자리를 쓰지 않으면 0으로 채워집니다.

yyyyMMddHHmmss 형식으로 검색할 기간의 시작 날짜와 시각을 지정합니다(기본값: 없음). 입력한 시각부터 검색을 시작합니다. 앞부분만 입력하면 나머지 자리는 0으로 인식합니다. 예를 들어 20130605를 입력하면 20130605000000(2013년 6월 5일 0시 0분 0초)으로 인식합니다. 이 옵션을 span과 함께 사용할 수 없습니다.

**span=INT{y|mon|w|d|h|m|s}**

현재 시각으로부터 일정 시간 범위 이내의 변경 파일로 한정하여 조회합니다. 미지정 시 기본값은 1일 범위를 사용합니다. y(연), mon(월), w(주), d(일), h(시), m(분), s(초) 단위로 지정할 수 있습니다. 예를 들면, 10s의 경우 현재 시각으로부터 10초 이전까지의 범위를 의미합니다.

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_path | 문자열 | 파일 경로 |
| file\_name | 문자열 | 파일 이름 |
| file\_type | 문자열 | 파일 유형 |
| permissions | 문자열 | 접근 권한 |
| file\_size | 정수 | 파일 크기 |
| file\_ctime | 날짜 | 파일 생성 시각 |
| file\_mtime | 날짜 | 파일 수정 시각 |
| file\_atime | 날짜 | 파일 접근 시각 |
| owner\_read | 불리언 | 소유자 읽기 권한 여부 |
| owner\_write | 불리언 | 소유자 쓰기 권한 여부 |
| owner\_execute | 불리언 | 소유자 실행 권한 여부 |
| group\_read | 불리언 | 그룹 읽기 권한 여부 |
| group\_write | 불리언 | 그룹 쓰기 권한 여부 |
| group\_execute | 불리언 | 그룹 실행 권한 여부 |
| others\_read | 불리언 | 기타 읽기 권한 여부 |
| others\_write | 불리언 | 기타 쓰기 권한 여부 |
| others\_execute | 불리언 | 기타 실행 권한 여부 |

#### 사용 예

최근 하루 동안 생성되거나 수정된 파일 목록을 조회

linux-recent-files

최근 7일 동안 생성되거나 수정된 파일 목록을 조회

linux-recent-files span=7d

2021년 1월 1일 이후로 생성되거나 수정된 파일 목록을 조회

linux-recent-files from=20210101

### linux-rkhunter

rkhunter 실행하고 결과를 조회합니다. 이 명령어를 사용하려면 rkhunter가 시스템에 설치되어 있어야 합니다. [rhhunter](http://rkhunter.sourceforge.net/)는 POSIX 호환 시스템에서 루트킷을 모니터링하는 오픈소스 소프트웨어입니다.

#### 문법

linux-rkhunter [ignore-error=t]

선택 매개변수

**ignore-error=BOOL**

오류 처리 옵션(기본값: f)

* t: 오류가 발생해도 정상 종료를 반환 (예: rkhunter가 설치되어 있지 않은 경우)
* f: 오류가 발생하면 실패를 반환

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| target | 문자열 | 대상 |
| reason | 문자열 | 진단 사유 |
| description | 문자열 | 상세정보 |

### linux-routes

/proc/net/route으로부터 라우팅 정보를 조회합니다.

#### 문법

linux-routes

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| destination | IPv4 주소 | 목적 네트워크 주소 |
| mask | IPv4 주소 | 넷마스크 |
| forward | IPv4 주소 | 게이트웨이 주소 |
| flags | 문자열 | 플래그 (C: 캐시 엔트리, D: 다이나믹, G: 게이트웨이, H: 호스트, M: 변경됨, U: 활성) |
| mss | 32비트 정수 | 최대 세그먼트 크기 |
| irtt | 32비트 정수 | 초기 라운드 트립 시간 |
| iface | 문자열 | 네트워크 인터페이스 이름 |

### linux-setuid-files

find / -user root -perm -4000 -print 명령을 실행하고 setuid가 설정된 파일 목록을 조회합니다.

#### 문법

linux-setuid-files [md5=BOOL]

선택 매개변수

**md5=BOOL**

파일의 MD5 해시 출력 여부(기본값: f)

* t: 파일의 MD5 해시를 출력
* f: 파일의 MD5 해시를 출력하지 않음

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_path | 문자열 | 파일 경로 |
| file\_name | 문자열 | 파일 이름 |
| file\_type | 문자열 | 파일 유형 |
| permissions | 문자열 | 접근 권한 |
| file\_size | 정수 | 파일 크기 |
| file\_ctime | 날짜 | 파일 생성 시각 |
| file\_mtime | 날짜 | 파일 수정 시각 |
| file\_atime | 날짜 | 파일 접근 시각 |
| md5 | 문자열 | 파일의 MD5 해시 값 |
| owner\_read | 불리언 | 소유자 읽기 권한 여부 |
| owner\_write | 불리언 | 소유자 쓰기 권한 여부 |
| owner\_execute | 불리언 | 소유자 실행 권한 여부 |
| group\_read | 불리언 | 그룹 읽기 권한 여부 |
| group\_write | 불리언 | 그룹 쓰기 권한 여부 |
| group\_execute | 불리언 | 그룹 실행 권한 여부 |
| others\_read | 불리언 | 기타 읽기 권한 여부 |
| others\_write | 불리언 | 기타 쓰기 권한 여부 |
| others\_execute | 불리언 | 기타 실행 권한 여부 |

### linux-shell-sessions

/var/run/utmp 로그를 파싱하여 리눅스에 현재 로그인한 상태의 셸 세션 목록을 조회합니다.

#### 문법

linux-shell-sessions

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| user | 문자열 | 사용자 ID |
| tty | 문자열 | 터미널 |
| host | 문자열 | 호스트 이름 |
| src\_ip | IPv4 주소 | 로그인을 수행한 원격 IP 주소 |
| idle\_time | 정수 | 유휴 시간 (초 단위) |
| jcpu | 64비트 실수 | 터미널에서 실행된 모든 프로세스가 사용한 CPU 시간 (초 단위) |
| pcpu | 64비트 실수 | 현재 프로세스가 사용한 CPU 시간 (초 단위) |
| cmd\_line | 문자열 | 실행 중인 명령줄 |

### linux-system-files

/usr/bin 디렉터리에 있는 파일들의 접근 권한과 MD5 해시를 조회합니다.

#### 문법

linux-system-files [md5=BOOL]

선택 매개변수

**md5=BOOL**

파일의 MD5 해시 출력 여부(기본값: f)

* t: 파일의 MD5 해시를 출력
* f: 파일의 MD5 해시를 출력하지 않음

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_path | 문자열 | 파일 경로 |
| file\_name | 문자열 | 파일 이름 |
| file\_type | 문자열 | 파일 유형 |
| permissions | 문자열 | 접근 권한 |
| file\_size | 정수 | 파일 크기 |
| file\_ctime | 날짜 | 파일 생성 시각 |
| file\_mtime | 날짜 | 파일 수정 시각 |
| file\_atime | 날짜 | 파일 접근 시각 |
| md5 | 문자열 | 바이너리에 대한 MD5 해시 값 |
| owner\_read | 불리언 | 소유자 읽기 권한 여부 |
| owner\_write | 불리언 | 소유자 쓰기 권한 여부 |
| owner\_execute | 불리언 | 소유자 실행 권한 여부 |
| group\_read | 불리언 | 그룹 읽기 권한 여부 |
| group\_write | 불리언 | 그룹 쓰기 권한 여부 |
| group\_execute | 불리언 | 그룹 실행 권한 여부 |
| others\_read | 불리언 | 기타 읽기 권한 여부 |
| others\_write | 불리언 | 기타 쓰기 권한 여부 |
| others\_execute | 불리언 | 기타 실행 권한 여부 |

### linux-shell-commands

/etc/passwd에 등록된 사용자별 bash\_history, zsh\_history 로그를 조회하여 사용자별로 실행한 명령 이력을 조회합니다.

#### 문법

linux-shell-commands

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| user | 문자열 | 사용자 ID |
| cmd\_line | 문자열 | 실행한 명령줄 |

### linux-system-info

리눅스 시스템 정보를 조회합니다. 호스트명, 커널 버전, 가동 시간, 평균 부하, UMASK 정보를 포함합니다.

#### 문법

linux-system-info

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| hostname | 문자열 | 호스트 이름 |
| kernel\_ver | 문자열 | 커널 버전 |
| kernel\_build\_time | 날짜 | 커널 시작 시각 |
| uptime | 정수 | 시스템 가동 시간 |
| load\_avg\_1m | 64비트 실수 | 1분 평균 부하 |
| load\_avg\_5m | 64비트 실수 | 5분 평균 부하 |
| load\_avg\_15m | 64비트 실수 | 15분 평균 부하 |
| umask | 문자열 | 파일 또는 디렉터리 생성 시 적용되는 권한 마스크 |
| console\_banner | 문자열 | 콘솔 배너 |
| telnet\_banner | 문자열 | 텔넷 배너 |
| system\_banner | 문자열 | 시스템 배너 |

### linux-systemd-services

systemctl -at service 명령을 실행하여 리눅스 systemd 서비스 목록을 조회합니다.

#### 문법

linux-systemd-services

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| service | 문자열 | 서비스 유닛 |
| load\_status | 문자열 | 서비스 로드 상태 |
| active\_status | 문자열 | 고수준 활성 상태 |
| sub\_status | 문자열 | 저수준 활성 상태 |

### linux-systemd-timers

systemctl list-unit-files --type=timer 명령을 실행하여 리눅스 systemd 타이머 목록을 조회합니다.

#### 문법

linux-systemd-timers

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| timer | 문자열 | 타이머 명칭 |
| status | 문자열 | 활성화 상태 |

### linux-tmp-files

/tmp, /dev, /home 디렉터리 이하의 숨겨진 파일(.으로 시작하는 파일 이름) 목록을 조회합니다.

#### 문법

linux-tmp-files

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| file\_path | 문자열 | 파일 경로 |
| file\_name | 문자열 | 파일 이름 |
| file\_type | 문자열 | 파일 유형 |
| permissions | 문자열 | 접근 권한 |
| file\_size | 정수 | 파일 크기 |
| file\_ctime | 날짜 | 파일 생성 시각 |
| file\_mtime | 날짜 | 파일 수정 시각 |
| file\_atime | 날짜 | 파일 접근 시각 |
| owner\_read | 불리언 | 소유자 읽기 권한 여부 |
| owner\_write | 불리언 | 소유자 쓰기 권한 여부 |
| owner\_execute | 불리언 | 소유자 실행 권한 여부 |
| group\_read | 불리언 | 그룹 읽기 권한 여부 |
| group\_write | 불리언 | 그룹 쓰기 권한 여부 |
| group\_execute | 불리언 | 그룹 실행 권한 여부 |
| others\_read | 불리언 | 기타 읽기 권한 여부 |
| others\_write | 불리언 | 기타 쓰기 권한 여부 |
| others\_execute | 불리언 | 기타 실행 권한 여부 |

### linux-users

/etc/passwd 파일을 파싱하여 사용자 목록을 조회합니다.

#### 문법

linux-users

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| user | 문자열 | 사용자 ID |
| uid | 정수 | UID |
| gid | 정수 | GID |
| description | 문자열 | 설명 |
| home\_path | 문자열 | 홈 디렉터리 경로 |
| shell | 문자열 | 로그인 셸 |

### linux-user-groups

root나 wheel 그룹에 속한 사용자를 식별하여 사용자 그룹 목록을 조회합니다.

#### 문법

linux-user-groups

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| gid | 정수 | 그룹 아이디 |
| group | 문자열 | 그룹 이름 |
| users | 문자열 | 계정 목록 |

### linux-vmstats

시스템의 입력과 출력 현황을 조회합니다. 이 쿼리를 사용하려면 시스템에서 vmstat 명령을 실행할 수 있어야 합니다.

#### 문법

linux-vmstats

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| swap\_size | 정수 | 가상 메모리 크기 |
| free\_size | 정수 | 유휴 메모리 크기 |
| buffer\_size | 정수 | 파일 메타데이터 버퍼 크기 |
| cache\_size | 정수 | 파일 데이터 캐시 크기(페이지 캐시) |
| swap\_in | 정수 | 디스크에서 읽어온 초당 스왑 크기 |
| swap\_out | 정수 | 디스크로 내보낸 초당 스왑 크기 |
| block\_in | 정수 | 디스크에서 메모리로 읽어온 초당 블록 수 |
| block\_out | 정수 | 메모리에서 디스크로 내보낸 초당 블록 수 |

## 웹 브라우저 아티팩트

### chrome-cookies

크롬 브라우저의 'Cookies' SQLite 파일에서 쿠키 목록을 추출합니다.

#### 문법

chrome-cookies FILE\_PATH

필수 매개변수

**FILE\_PATH**

크롬 브라우저의 'Cookies' SQLite 파일 경로. 크롬 브라우저(96 버전 이상)의 쿠키 파일은 C:\Users<username>\AppData\Local\Google\Chrome\User Data\Default\Network\Cookies 위치에 있습니다.

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| \_time | 시간 | 생성 일시 |  |
| host\_key | 문자열 | 호스트 키 |  |
| top\_frame\_site\_key | 문자열 | 최상위 도메인 |  |
| name | 문자열 | 이름 |  |
| value | 문자열 | 값 | 최신 버전에서는 사용되지 않음 |
| encrypted\_value | 바이너리 | 암호화 |  |
| path | 경로 | 경로 |  |
| expires | 날짜 | 쿠키 만료 일시 |  |
| priority | 32비트 정수 | 중요도 | 0~3 범위. 숫자가 작은 경우 디스크 용량 부족 시 먼저 삭제됨 |
| is\_secure | 32비트 정수 | Secure 쿠키 여부 | 1 또는 0 |
| is\_httponly | 32비트 정수 | HttpOnly 쿠키 여부 | 1 또는 0 |
| has\_expires | 32비트 정수 | 만료 일시 지정 여부 |  |
| is\_persistent | 32비트 정수 | 지속 여부 | 1 또는 0 |
| is\_same\_party | 32비트 정수 | 1사 쿠키 여부 | 1 또는 0 (3사 쿠키) |
| samesite | 32비트 정수 | SameSite 설정 | -1 (unspecified), 0 (no\_restriction), 1 (lax), 2 (strict) |
| source\_scheme | 32비트 정수 | 원본 스킴 | 1 (HTTP), 2 (HTTPS) |
| source\_port | 32비트 정수 | 원본 포트 |  |
| created | 날짜 | 생성 일시 |  |
| updated | 날짜 | 수정 일시 |  |
| last\_access | 날짜 | 최근 접근 일시 |  |

쿠키 분류

* Secure 쿠키: HTTPS 통신인 경우에만 값을 전송하는 쿠키
* HttpOnly 쿠키: 자바스크립트 API (document.cookie)로 접근 불가능한 쿠키
* SameSite 쿠키: CSRF 공격을 차단하기 위해 크로스 사이트 전송을 차단하는 쿠키

#### 사용 예

크롬 브라우저의 쿠키 목록 추출

chrome-cookies C:\Users\Logpresso\AppData\Local\Google\Chrome\User Data\Default\Network\Cookies

### chrome-downloads

크롬 'History' SQLite 파일에서 파일을 다운로드한 이력을 추출합니다.

#### 문법

chrome-downloads FILE\_PATH

필수 매개변수

**FILE\_PATH**

크롬 'History' SQLite 파일 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 시간 | 방문 시각 |
| start\_time | 시간 | 다운로드 시작 시각 |
| end\_time | 시간 | 다운로드 종료 시각 |
| mime\_type | 문자열 | MIME 타입 |
| file\_open | 불리언 | 파일 열기 여부 |
| file\_path | 문자열 | 파일 경로 |
| file\_size | 정수 | 파일 크기 |
| url | 문자열 | 다운로드 URL |
| referer | 문자열 | 참조 URL |

#### 사용 예

/opt/logpresso/History 파일에서 파일 다운로드 이력 추출

chrome-downloads /opt/logpresso/History

### chrome-plugin-artifacts

크롬 브라우저의 'manifest.json' 파일에서 크롬 확장 프로그램 정보를 추출합니다.

#### 문법

chrome-plugin-artifacts FILE\_PATH

필수 매개변수

**FILE\_PATH**

크롬 브라우저의 'manifest.json' 파일 경로. 이 파일은 일반적으로 C:\Users<username>\AppData\Local\Google\Chrome\User Data\Default\Extensions<extension\_id><version>\manifest.json 위치에 있습니다.

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

**출력 필드**

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 설명 |
| name | 문자열 | 확장 프로그램 이름 |  |
| version | 문자열 | 확장 프로그램 버전 |  |
| description | 문자열 | 설명 |  |
| minimum\_browser\_version | 문자열 | 브라우저 최소 버전 |  |
| permissions | 배열 | 권한 목록 | 예: alarms, storage |
| update\_url | 문자열 | 업데이트 URL |  |
| author | 맵 | 작성자 정보 | email 키 포함 |
| background | 맵 | 백그라운드 스크립트 |  |
| content\_capabilities | 맵 | 기능 권한 목록 | 예: clipboardWrite |
| content\_security\_policy | 문자열 | 스크립트 실행 정책 |  |
| default\_locale | 문자열 | 기본 로케일 | 예: en-US |
| differential\_fingerprint | 문자열 | 확장 프로그램 해시 값 |  |
| externally\_connectable | 맵 | 통신 허용 URL 패턴 목록 |  |
| file\_atime | 날짜 | manifest.json 파일 접근 시각 |  |
| file\_ctime | 날짜 | manifest.json 파일 생성 시각 |  |
| file\_mtime | 날짜 | manifest.json 파일 수정 시각 |  |
| icons | 맵 | 아이콘 이미지 파일 매핑 |  |
| key | 문자열 | 확장 프로그램 고유 ID |  |
| manifest\_version | 32비트 정수 | 매니페스트 문법 버전 | 예: 2 |
| storage | 맵 | 데이터 스키마 등 스토리지 설정 |  |
| web\_accessible\_resources | 배열 | 접근 허용 웹 리소스 파일 목록 |  |

#### 사용 예

크롬 확장 매니페스트 파일 조회

chrome-plugin-artifacts C:\Users\Logpresso\AppData\Local\Google\Chrome\User Data\Default\Extensions\ghbmnnjooekpmoecnnnilnnbdlolhkhi\1.53.0\_0\manifest.json

### chrome-search-terms

크롬 'History' SQLite 파일에서 검색 이력을 추출합니다.

#### 문법

chrome-search-terms FILE\_PATH

필수 매개변수

**FILE\_PATH**

크롬 'History' SQLite 파일 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 방문 시각 |
| keywords | 문자열 | 검색한 단어 |
| title | 문자열 | 웹페이지 제목 |
| url | 문자열 | 웹페이지 URL |

#### 사용 예

/opt/logpresso/History 파일에서 검색 이력 추출

chrome-search-terms /opt/logpresso/History

### chrome-visits

크롬 History DB 파일에서 웹 사이트 방문 기록을 추출합니다.

#### 문법

chrome-visits FILE\_PATH

필수 매개변수

**FILE\_PATH**

크롬 'History' SQLite 파일 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 방문 시각 |
| title | 정수 | 웹페이지 제목 |
| visit\_count | 정수 | 웹페이지 방문 횟수 |
| typed\_count | 정수 | 주소창에 직접 입력하여 접속한 횟수 |
| hidden | 불리언 | iframe 여부 |
| url | 문자열 | 방문 URL |

#### 사용 예

/opt/logpresso/History 파일에서 웹사이트 방문 기록 추출

chrome-visits /opt/logpresso/History

### esedb-columns

ESE(Extensible Storage Engine) 데이터베이스 파일에서 테이블별 컬럼 정의 목록을 조회합니다.

#### 문법

esedb-columns table=TABLE FILE\_PATH

필수 매개변수

**FILE\_PATH**

ESE 데이터베이스 파일 경로

**table=TABLE**

ESE 데이터베이스 파일에 포함된 테이블 이름

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_file | 문자열 | 파일 이름 |
| table\_name | 문자열 | 테이블 이름 |
| column\_id | 정수 | 컬럼 아이디 |
| column\_name | 문자열 | 컬럼 이름 |
| column\_type | 문자열 | 컬럼 데이터 타입 |

**column\_type** 필드는 다음 중 하나의 값을 표시합니다: bit, unsigned byte, short, unsigned short, long, unsigned long, long long, binary, long binary, text

#### 사용 예

/opt/logpresso/WebCacheV01.dat 파일에서 MSysObjects 테이블의 컬럼 목록을 조회

esedb-columns table=MSysObjects /opt/logpresso/WebCacheV01.dat

### esedb-records

ESE(Extensible Storage Engine) 데이터베이스 파일에서 지정된 테이블의 데이터를 조회합니다.

#### 문법

esedb-records table=TABLE FILE\_PATH

필수 매개변수

**table=TABLE**

ESE 데이터베이스에서 레코드를 조회할 테이블 이름

**FILE\_PATH**

ESE 데이터베이스 파일 경로

#### 사용 예

/opt/logpresso/WebCacheV01.dat 파일에서 MSysObjects 테이블의 레코드 데이터를 조회

esedb-records table=MSysObjects /opt/logpresso/WebCacheV01.dat

### esedb-tables

ESE(Extensible Storage Engine) 데이터베이스 파일에서 전체 테이블 목록 및 컬럼 구성을 조회합니다.

#### 문법

esedb-tables FILE\_PATH

필수 매개변수

**FILE\_PATH**

ESE 데이터베이스 파일 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_file | 문자열 | 설명 |
| table\_name | 문자열 | 테이블 이름 |
| columns | 리스트 | 컬럼 목록 |

#### 사용 예

/opt/logpresso/WebCacheV01.dat 파일의 테이블 목록 및 컬럼 구성 조회

esedb-tables /opt/logpresso/WebCacheV01.dat

### eml-file

[RFC 822](https://www.rfc-editor.org/rfc/rfc822)를 준수하는 EML 파일에서 헤더, 제목, 본문 등 이메일 정보를 추출합니다.

#### 문법

eml-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] [raw=BOOL] FILE\_PATH

필수 매개변수

**FILE\_PATH**

EML 파일의 경로. 파일 경로에 와일드카드(\*)를 사용하면(예: D:\data\eml\*.eml) 패턴과 일치하는 모든 파일을 한 번에 조회할 수 있습니다. zippath 옵션과 함께 사용하는 경우, ZIP 파일에 포함된 EML 파일의 경로를 입력하십시오.

선택 매개변수

**zipcharset=CHARSET**

ZIP 엔트리 인코딩 형식 (기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

ZIP 파일의 경로.

**raw=BOOL**

mail\_content 필드에 출력할 메일 본문의 형식(기본값: f)

* t: HTML
* f: 일반 텍스트

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_file | 문자열 | EML 파일 이름 |
| \_time | 날짜 | 보낸 시각. RFC 822의 Date 필드 |
| mail\_from | 문자열 | 보낸 사람 주소 |
| mail\_from\_name | 문자열 | 보낸 사람 이름 |
| mail\_to | 문자열 | 받는 사람 주소. 개행으로 구분 |
| mail\_to\_name | 문자열 | 받는 사람 이름. 개행으로 구분 |
| mail\_cc | 문자열 | 참조 주소. 개행으로 구분 |
| mail\_cc\_name | 문자열 | 참조 이름. 개행으로 구분 |
| mail\_bcc | 문자열 | 숨은 참조 주소. 개행으로 구분 |
| mail\_bcc\_name | 문자열 | 숨은 참조 이름. 개행으로 구분 |
| mail\_subject | 문자열 | 메일 제목 |
| mail\_content | 문자열 | 텍스트 혹은 HTML 형식의 메일 본문 |
| mail\_attachments | 문자열 | 첨부 파일 목록. 개행으로 구분 |
| mail\_headers | 문자열 | 메일 헤더 목록. 개행으로 구분 |
| attachments | 배열 | 각 요소는 file\_name, file\_size 포함 |

#### 사용 예

EML 파일 경로를 입력하여 조회

eml-file sample.eml

ZIP 파일 내에 압축된 EML 파일 조회

eml-file zippath=image.zip sample.eml

### firefox-downloads

파이어폭스 places.sqlite DB 파일에서 다운로드 이력을 추출합니다.

#### 문법

firefox-downloads FILE\_PATH

필수 매개변수

**FILE\_PATH**

파이어폭스 places.sqlite DB 파일 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 방문 시각 |
| start\_time | 날짜 | 다운로드 시작 시각 |
| end\_time | 날짜 | 다운로드 종료 시각 |
| file\_path | 문자열 | 파일 경로 |
| file\_size | 정수 | 파일 크기 |
| url | 문자열 | 다운로드 URL |

#### 사용 예

/opt/logpresso/places.sqlite 파일에서 파일 다운로드 이력 조회

firefox-downloads /opt/logpresso/places.sqlite

### firefox-visits

파이어폭스 places.sqlite DB 파일에서 웹 사이트 방문 이력을 추출합니다.

#### 문법

firefox-visits FILE\_PATH

필수 매개변수

**FILE\_PATH**

파이어폭스 places.sqlite DB 파일 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 방문 시각 |
| title | 문자열 | 웹페이지 제목 |
| visit\_count | 정수 | 웹페이지 방문 횟수 |
| typed\_count | 정수 | 주소창에 직접 입력하여 접속한 횟수 |
| hidden | 불리언 | iframe여부 |
| url | 문자열 | 방문 URL |

#### 사용 예

/opt/logpresso/places.sqlite 파일에서 웹사이트 방문 이력 조회

firefox-visits /opt/logpresso/places.sqlite

### ie-cache-files

ESE 데이터베이스 파일에서 캐시 파일 데이터를 추출합니다.

#### 문법

ie-cache-files FILE\_PATH

필수 매개변수

**FILE\_PATH**

ESE 데이터베이스 파일 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 방문 시각 |
| container\_id | 정수 | 컨테이너 ID |
| entry\_id | 정수 | 엔트리 ID |
| cache\_id | 정수 | 캐시 ID |
| visit\_count | 정수 | 웹사이트 방문 횟수 |
| url | 문자열 | 다운로드 URL |
| file\_name | 문자열 | 파일 이름 |
| file\_size | 정수 | 파일 크기 |
| sync\_time | 날짜 | 접근 시각 |
| creation\_time | 날짜 | 생성 시각 |
| expiry\_time | 날짜 | 만료 시각 |
| modified\_time | 날짜 | 수정 시각 |
| request\_headers | 문자열 | HTTP 요청 헤더 |
| response\_headers | 문자열 | HTTP 응답 헤더 값 |
| group | 바이너리 |  |
| url\_hash | 정수 | URL 해시 값 |
| secure\_dir | 정수 | 디렉토리 인덱스 |

#### 사용 예

/opt/logpresso/WebCacheV01.dat 파일에서 캐시 파일 데이터를 추출

ie-cache-files /opt/logpresso/WebCacheV01.dat

### ie-cookies

ESE 데이터베이스 파일에서 쿠키 데이터를 추출합니다.

#### 문법

ie-cookies FILE\_PATH

필수 매개변수

**FILE\_PATH**

ESE 데이터베이스 파일 경로

#### 사용 예

/opt/logpresso/WebCacheV01.dat 파일에서 쿠키 데이터 추출

ie-cookies /opt/logpresso/WebCacheV01.dat

### ie-downloads

ESE 데이터베이스 파일에서 다운로드 기록을 추출합니다.

#### 문법

ie-downloads FILE\_PATH

필수 매개변수

**FILE\_PATH**

ESE 데이터베이스 파일 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 방문 시각 |
| container\_id | 정수 | 컨테이너 ID |
| entry\_id | 정수 | 엔트리 ID |
| cache\_id | 정수 | 캐시 ID |
| visit\_count | 정수 | 웹페이지 방문 횟수 |
| url | 문자열 | 다운로드 URL |
| file\_path | 문자열 | 파일 경로 |
| file\_size | 정수 | 파일 크기 |
| sync\_time | 날짜 | 다운로드 시각 |
| response\_headers | 문자열 | HTTP 응답 헤더 |
| url\_hash | 정수 | URL 해시 |
| secure\_dir | 정수 | 디렉토리 인덱스 |

#### 사용 예

/opt/logpresso/WebCacheV01.dat 파일에서 파일 다운로드 기록 추출

ie-downloads /opt/logpresso/WebCacheV01.dat

### ie-visits

ESE 데이터베이스 파일에서 웹사이트 방문 기록을 추출합니다.

#### 문법

ie-visits FILE\_PATH

필수 매개변수

**FILE\_PATH**

ESE 데이터베이스 파일 경로

#### 설명

출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| \_time | 날짜 | 방문 시각 |
| container\_id | 정수 | 컨테이너 ID |
| entry\_id | 정수 | 엔트리 ID |
| cache\_id | 정수 | 캐시 ID |
| visit\_count | 정수 | 웹사이트 방문 횟수 |
| user | 문자열 | NT 계정 |
| url | 문자열 | 방문 URL |
| file\_size | 정수 | 파일 크기 |
| sync\_time | 날짜 | 접근 시각 |
| expiry\_time | 날짜 | 만료 시각 |
| modified\_time | 날짜 | 수정 시각 |
| response\_headers | 문자열 | HTTP 응답 헤더 값 |
| url\_hash | 정수 | URL 해시 값 |
| secure\_dir | 정수 | 디렉토리 인덱스 |

#### 사용 예

/opt/logpresso/WebCacheV01.dat 파일에서 웹사이트 방문 기록 추출

ie-visits /opt/logpresso/WebCacheV01.dat

### sqlite-records

SQLite 데이터베이스 파일에서 지정된 테이블 레코드를 조회합니다.

#### 문법

sqlite-records table=TABLE FILE\_PATH

필수 매개변수

**table=TABLE**

레코드를 조회할 SQLite 데이터베이스 테이블 이름

**FILE\_PATH**

SQLite 데이터베이스 파일 경로

#### 사용 예

/opt/logpresso/sqlite 파일에서 visits 테이블의 레코드를 조회

sqlite-records table=visits /opt/logpresso/sqlite

### sqlite-tables

SQLite 데이터베이스 파일에서 테이블 스키마 목록을 조회합니다.

#### 문법

sqlite-tables FILE\_PATH

필수 매개변수

**FILE\_PATH**

SQLite 데이터베이스 파일 경로

#### 설명

명령을 실행한 다음 출력되는 필드 내용은 다음 표를 참조하십시오.

출력 필드

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| type | 문자열 | 객체 구분 |
| name | 문자열 | 객체 구분 |
| table\_name | 문자열 | 테이블 이름 |
| root\_page | 정수 | 루트 페이지 번호 |
| sql | 문자열 | 테이블 스키마 |

#### 사용 예

/opt/logpresso/sqlite 파일의 테이블 스키마 조회

sqlite-tables /opt/logpresso/sqlite

# 시스템 명령어

## 시스템 명령어

### confdb

로그프레소의 설정을 저장하는 confdb의 설정값 데이터를 조회합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

confdb databasesconfdb cols DB\_NAMEconfdb docs DB\_NAME COL\_NAMEconfdb logs DB\_NAME

필수 매개변수

**databases**

confdb의 데이터베이스 인스턴스 목록 조회

**cols DB\_NAME**

특정 설정 데이터베이스 내부의 모든 컬렉션 목록 조회

**docs DB\_NAME COL\_NAME**

특정 설정 컬렉션 내부의 모든 문서 개체 조회

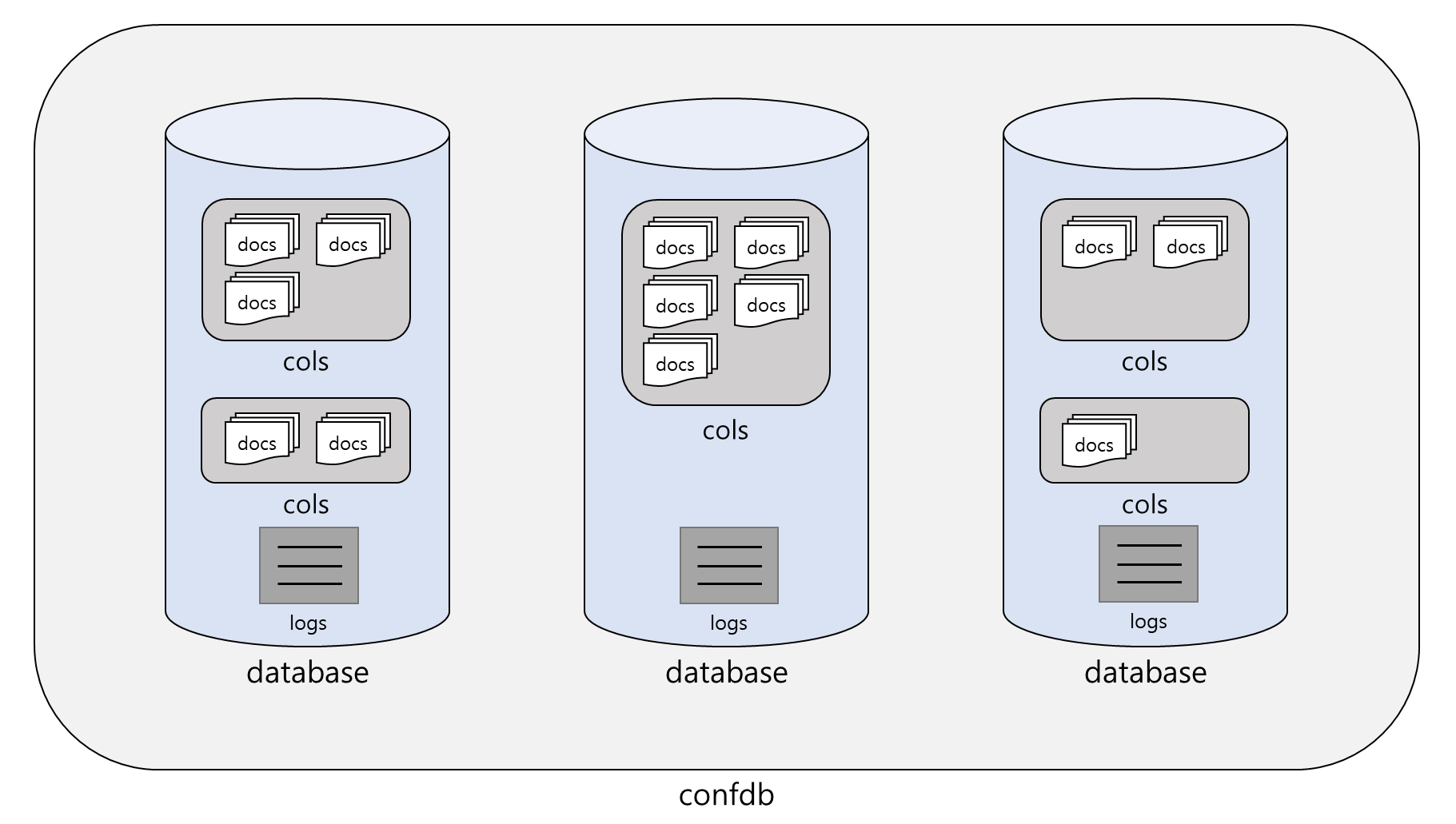
**logs DB\_NAME**

특정 데이터베이스의 변경 로그 조회

#### 설명

confdb는 다음과 같은 구조로 구성되어 있습니다. 설정 항목들은 개별 데이터베이스로 분류되며, 각 데이터베이스는 컬렉션의 집합으로 구성됩니다. 각 컬렉션은 문서의 집합으로 구성됩니다.

confdb databases 명령어로 최상단 데이터베이스 목록을 조회할 수 있으며, confdb cols 명령어로 특정 데이터베이스의 컬렉션 목록을 조회할 수 있습니다. confdb docs 명령어를 통해 특정 컬렉션에 있는 문서 목록을 조회할 수 있습니다.



#### 사용 예

모든 데이터베이스 인스턴스 목록 조회

confdb databases

araqne-cron 데이터베이스 인스턴스의 컬렉션 목록 조회

confdb cols araqne-cron

araqne-cron 데이터베이스 인스턴스의 schedule 컬렉션 조회

confdb docs araqne-cron schedule

araqne-cron 데이터베이스 인스턴스의 로그 조회

confdb logs araqne-cron

## 로그 수집기 관리

### system loggers

현재 설정되어 있는 모든 로그 수집기의 상태를 조회합니다. 관리자 권한이 필요합니다.

#### 문법

system loggers

#### 설명

명령문을 실행했을 때 보여주는 필드는 다음과 같습니다.

* namespace: 로거가 속한 집합의 이름(로그프레소 로컬에서 실행하는 로거는 local 집합을 의미하며, 그 외에는 로거가 위치한 센트리의 이름(guid)으로 표시)
* name: 로거의 이름
* factory\_namespace: 로거 팩토리가 속한 집합의 이름
* factory\_name: 로거 팩토리의 이름입니다.
* status: 현재 로거의 상태입니다(running: 실행 중, stopped: 멈춤).
* interval: 로거 실행 주기(단위: ms)
* cron\_schedule: 예약된 실행 일정
* log\_count: 현재까지 저장된 로그의 개수
* drop\_count: 드롭된 로그의 개수
* log\_volume: 로그 데이터 파일의 전체 용량(byte)
* drop\_volume: 드롭된 로그 데이터의 전체 용량(byte)
* last\_start\_at: 로거가 시작된 최근 시간입니다.
* last\_run\_at: 로거가 실행된 최근 시간입니다.
* last\_log\_at: 마지막으로 들어온 로그의 유입 시간
* last\_write\_at: 마지막으로 로그를 받은 시간입니다.

로거 팩토리는 각각 다른 기능을 수행하는 로거의 유형 그룹을 의미합니다. 로거 팩토리는 노드(로그프레소(local), 또는 센트리(guid)마다 영역이 구분됩니다. 같은 로거 팩토리라 하더라도 local 네임스페이스에 속하는 것과 센트리의 네임스페이스에 속하는 것은 서로 다른 로거 팩토리입니다.

## 테이블 및 데이터 관리

### system tables

시스템의 테이블 목록을 조회합니다. 관리자는 시스템의 모든 테이블 목록을 조회할 수 있습니다. 일반 사용자 계정은 자신에게 권한이 부여된 테이블 목록만 조회할 수 있습니다.

#### 문법

system tables

#### 설명

명령문을 실행했을 때 보여주는 필드는 다음과 같습니다.

* **table**: 테이블 이름
* **metadata**: 테이블의 메타데이터
* **primary\_configs**: 주 스토리지 설정 정보
* **replica\_configs**: 백업 스토리지 설정 정보
* **lock\_owner**: 락 소유자
* **lock\_purpose**: 락을 건 목적
* **lock\_reentcnt**: 락을 건 프로세스가 재진입하면서 잠근 횟수
* **retention\_policy**: 테이블 보관 주기(단위: 일)
* **data\_path**: 테이블 데이터 파일 경로
* **is\_locked**: 테이블 잠금 여부 (true: 잠김, false: 잠기지 않음)
* **privileges**: 사용자 계정 테이블 접근 권한
* **security\_groups**: 사용자 그룹 테이블 접근 권한

테이블에 저장된 레코드를 확인하려면 [table](https://docs.logpresso.com/ko/query/table-command) 명령어를 사용하십시오.

### system count

쿼리 시점까지 저장된 모든 레코드 개수를 일자별로 조회합니다.

#### 문법

system count [from=yyyyMMdd] [to=yyyyMMdd] [diskonly=BOOL] [TABLE, ...]

선택 매개변수

**from=yyyyMMdd**

조회 범위 시작 일자(시작 일자 포함하여 조회)를 yyyyMMdd 형식으로 지정합니다.

**to=yyyyMMdd**

조회 범위 마지막 일자를 yyyyMMdd 형식으로 지정합니다. [table](https://docs.logpresso.com/ko/query/table-command) 명령어의 to 옵션과 달리, 지정한 일자도 조회 범위에 포함합니다.

**diskonly=BOOL**

디스크에 기록된 건수만 조회 여부(기본값: f)

* t: 디스크에 기록된 건수만 조회
* f: 메모리에 버퍼링된 데이터 건수도 포함해서 조회

**[TABLE, ...]**

레코드 개수를 확인할 테이블을 쉼표(,)로 구분하여 지정합니다. 테이블을 지정하지 않으면 사용자에게 읽기 권한이 부여된 모든 테이블의 레코드 개수를 확인합니다.

#### 설명

출력 필드는 다음과 같습니다.

* **\_time**: 파티션 일자
* **table**: 테이블 이름
* **count**: 테이블에 기록된 레코드 개수

### checktable

지정한 날짜 범위의 테이블 데이터 무결성을 검사합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

checktable [from=yyyyMMdd] [to=yyyyMMdd] [trace=BOOL] [TABLE, ...]

선택 매개변수

**from=yyyyMMdd**

무결성 검사 시작 일자(시작 일자 포함하여 검사)를 yyyyMMdd 형식으로 지정합니다.

**to=yyyyMMdd**

무결성 검사 마지막 일자 (마지막 일자 포함하여 검사)를 yyyyMMdd 형식으로 지정합니다.

**trace=BOOL**

무결성에 이상이 없는 데이터 블럭 정보 출력 여부(기본값: f)

* t: 무결성에 이상이 없는 정상 데이터 블럭 정보도 출력합니다.
* f: 무결성이 훼손된 데이터 블럭 정보만 보여줍니다.

**[TABLE, ...]**

무결성을 검사할 테이블을 쉼표(,)로 구분하여 지정합니다. 테이블을 지정하지 않으면 사용자에게 읽기 권한이 부여된 모든 테이블의 무결성을 확인합니다. 테이블 이름은 와일드카드(\*)를 지원합니다.

#### 설명

대상 테이블이 다이제스트 알고리즘이 설정된 암호화 프로파일을 이용하는 경우에만 무결성 검사가 수행되며, 무결성 검사에 필요한 HMAC 시그니처를 포함하지 않는 테이블은 검사에서 자동으로 제외됩니다.

출력 필드는 다음과 같습니다.

* table: 테이블 이름
* day: 날짜 파티션 이름
* block\_id: 블럭 ID
* last\_block\_id: 마지막으로 블럭 ID로, 무결성이 손상된 경우에만 나타납니다.
* signature: 데이터 생성 시점에 계산한 해시 값
* hash: 무결성 검사 시점에 계산한 해시 값. 이 값이 signature 필드 값과 다르면 변조된 것으로 간주합니다.
* msg: valid, modified, corrupted 문자열 중 하나의 값으로 표시합니다. 데이터가 변조되거나 손상된 경우, 데이터 조회 쿼리를 실행할 때 해당 데이터 블럭을 읽을 수 없으므로 건너뜁니다.
* valid: 무결성이 검증되었음
* modified: 데이터가 변조됨
* corrupted: 파일 구조가 손상됨

무결성 검사 시 이상이 없을 경우 별도 출력 결과가 없습니다.

#### 사용 예

모든 테이블의 2014년 9월 데이터 무결성 검사

checktable from=20140901 to=20140930 \*

syslog\_ 로 시작하는 모든 테이블 데이터 무결성 검사

checktable syslog\_\*

### copytable

지정한 날짜 범위의 테이블 데이터 및 인덱스 데이터 파일을 지정된 경로에 복사하거나 이동합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

copytable [from=yyyyMMdd] [to=yyyyMMdd] [incremental=t|overwrite=t|worm=t] [move=t] [tables="TABLE\_1, TABLE\_2, ..."] [indexpath="PATH"] path="PATH"

필수 매개변수

**path="PATH"**

테이블 백업 경로를 큰 따옴표 쌍(" ")으로 감싸서 지정합니다. 백업 경로에 백슬래시()나 공백문자와 같은 특수문자가 있으면 백슬래시()를 이용한 이스케이프 처리가 필요합니다.

선택 매개변수

**from=yyyyMMdd**

백업 시작 일자(시작 일자 포함하여 백업)를 yyyyMMdd 형식으로 지정합니다.

**to=yyyyMMdd**

백업 마지막 일자(마지막 일자 포함하여 백업)를 yyyyMMdd 형식으로 지정합니다.

**incremental=t**

백업 미디어의 경로에 동일한 파일이 있으면 기존 파일 끝에 데이터를 추가합니다. 이 옵션은 worm, overwrite 옵션과 함께 사용할 수 없습니다.

**overwrite=t**

백업 미디어의 경로에 동일한 파일이 있으면 기존 파일을 대체합니다. 확장자가 .transfer인 임시 파일에 데이터를 기록하고, 이 파일에 기록이 완료되면 기존 파일의 이름으로 변경하고 원본 파일을 삭제하는 방식을 사용합니다. 백업 실행 중에 취소하더라도 기존 백업 파일을 유지할 수 있습니다. 이 옵션은 incremental, worm 옵션과 함께 사용할 수 없습니다. 백업 방식을 아무것도 지정하지 않으면 overwrite 옵션을 적용합니다.

**worm=t**

WORM 스토리지(Write Once Read Many), CD에 테이블을 복사할 때 지정합니다. 이 옵션을 지정하면 확장자가 .transfer인 임시 파일을 생성하지 않고 백업 미디어에 복사를 실행합니다. 미지정 시 .transfer 확장자의 임시 파일을 생성한 후, 파일 이름을 변경하고 삭제합니다. 이 옵션은 incremental, overwrite 옵션과 함께 사용할 수 없습니다.

**move=t**

t로 설정하면 복사 완료 후 원본 파일을 삭제합니다. 백업 미디어의 파일 크기와 원본의 파일 크기가 일치하지 않는 경우에는 원본 파일을 삭제하지 않습니다.

**tables="TABLE\_1, TABLE\_2, ..."**

복제 대상 테이블 목록을 쉼표(,)로 구분하여 지정합니다. 전체 목록은 큰 따옴표 쌍(" ")으로 감싸서 입력합니다. 이 옵션을 지정하지 않으면 전체 테이블을 백업합니다.

**indexpath="PATH"**

풀텍스트 인덱스 파일을 지정된 경로를 큰 따옴표 쌍(" ")으로 감싸서 지정합니다. 지정하지 않으면 테이블 데이터 파일만 백업됩니다.

#### 설명

이 명령어는 일반적으로 테이블 데이터와 인덱스 데이터 파일을 주기적으로 NAS와 같은 스토리지에 백업할 때 사용합니다.

각 데이터 파일의 복사를 수행하면서 현재 진행 상황을 출력합니다. 이미 같은 이름의 파일이 존재하거나, 이름 변경이 실패하거나, 백업 미디어의 용량 부족과 같은 예외 상황이 발생하면 error\_msg 필드에 오류 내용을 표시합니다. 이를 이용해 오류가 발생했을 때 경보 메일을 전송하는 등 별도의 후처리를 수행할 수 있습니다. 중간에 일부 데이터 파일의 백업에 실패하더라도 사용자가 명시적으로 취소하기 전까지 쿼리는 중단되지 않고 실행됩니다.

#### 사용 예

모든 테이블 데이터 파일을 /backup 경로에 복사

copytable path="/backup"

2015년 6월 24일부터 2015년 6월 25일까지 모든 테이블 데이터 파일을 e:\backup 경로에 복사

copytable from=20150624 to=20150625 path="e:\\backup"

2015년 6월 24일부터 2015년 6월 25일까지 모든 테이블 데이터 파일을 /backup 경로에 이동

copytable from=20150624 to=20150625 move=t path="/backup"

2015년 6월 24일부터 2015년 6월 25일까지 test 테이블과 인덱스 데이터 파일을 /backup 경로에 복사

copytable from=20150624 to=20150625 tables="test" path="/backup" indexpath="/backup"

### purge

테이블에서 지정한 날짜 범위의 데이터 파일을 파기합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

purge from=yyyyMMdd to=yyyyMMdd TABLE\_1, TABLE\_2, ...

**from=yyyyMMdd**

파기 대상의 시작 일자(시작 일자 포함하여 파기)를 yyyyMMdd 형식으로 지정합니다.

**to=yyyyMMdd**

파기 대상의 마지막 일자(마지막 일자 포함하여 파기)를 yyyyMMdd 형식으로 지정합니다.

**TABLE\_1, TABLE\_2, ...**

데이터를 파기할 테이블 목록을 쉼표(,)로 구분하여 지정합니다. 쿼리를 실행할 때마다 기존 데이터를 파기하고 새로운 데이터를 입력하려는 경우에 사용합니다.

이 명령은 로그프레소 부팅 옵션으로 -Daraqne.logdb.purge=enabled 스위치를 추가한 경우에만 사용할 수 있습니다.

ENT-3.10.2006.0-u2352 이전 버전은 테이블 이름 사이에 공백이 있으면 purge 명령어가 제대로 동작하지 않습니다.

#### 사용 예

2014년 9월 10일과 2014년 9월 11일의 sample 테이블 데이터를 파기

purge from=20140910 to=20140911 sample

### system logdisk

압축된 테이블 데이터 파일들의 디스크 사용량을 일자별로 조회합니다.

#### 문법

system logdisk [from=yyyyMMdd] [to=yyyyMMdd] [TABLE\_1, TABLE\_2, ...]

**from=yyyyMMdd**

조회 대상의 시작 일자(시작 일자 포함)를 yyyyMMdd 형식으로 지정합니다.

**to=yyyyMMdd**

조회 대상의 마지막 일자(마지막 일자 포함)를 yyyyMMdd 형식으로 지정합니다.

**[TABLE\_1, TABLE\_2, ...]**

테이블 데이터 파일의 디스크 사용량을 조회할 테이블 목록을 쉼표(,)로 구분하여 지정합니다. 테이블 목록을 생략하면 명령문을 실행하는 사용자 계정에게 읽기 권한이 부여된 모든 테이블의 사용량을 조회합니다.

#### 설명

출력 필드는 다음과 같습니다.

* \_time: 파티션 일자
* table: 테이블 이름
* disk\_usage: 디스크 사용 바이트 수

### system indexdisk

쿼리 시점까지 저장된 모든 인덱스 파일들의 디스크 사용량을 일자 및 유형별로 조회합니다.

#### 문법

system indexdisk [from=yyyyMMdd] [to=yyyyMMdd] [TABLE\_1, TABLE\_2, ...]

**from=yyyyMMdd**

조회 대상의 시작 일자(시작 일자 포함)를 yyyyMMdd 형식으로 지정합니다.

**to=yyyyMMdd**

조회 대상의 마지막 일자(마지막 일자 포함)를 yyyyMMdd 형식으로 지정합니다.

**[TABLE\_1, TABLE\_2, ...]**

인덱스 파일의 디스크 사용량을 조회할 테이블 목록을 쉼표(,)로 구분하여 지정합니다. 테이블 목록을 생략하면 명령문을 실행하는 사용자 계정에게 읽기 권한이 부여된 모든 테이블에서 인덱스 파일의 디스크 사용량을 조회합니다.

#### 설명

출력 필드는 다음과 같습니다.

* \_time: 파티션 일자
* table: 테이블 이름
* index: 인덱스 이름
* type: 인덱스 유형
* disk\_usage: 디스크 사용 바이트 수

## 룩업 관리

### system lookups

현재 시스템에 등록된 모든 룩업(lookup) 이름 목록을 조회합니다.

#### 문법

system lookups

#### 설명

출력 필드는 다음과 같습니다.

* name: 룩업 이름

룩업은 데이터 매핑에 사용되는 파일 또는 데이터베이스 형태의 데이터를 의미합니다. ENT, STD 웹 콘솔 **쿼리 > 룩업** 메뉴에서 룩업을 조회할 수 있고, 이 명령어가 보여주는 것과 동일한 정보를 확인할 수 있습니다.

관련 명령어는 다음을 참조하십시오.

* geocode\_kr: 대한민국 행정구역 코드 룩업 조회
* lookup: 매핑 테이블을 조회하여 특정한 필드 값을 다른 값으로 변환
* lookuptable: 파일 형태로 추가한 룩업 테이블의 내용을 조회
* memlookup: 인메모리(in-memory) 매핑 테이블의 생성 및 메타데이터 조회

## 쿼리 관리

### system queries

현재 실행 중인 모든 쿼리의 상태를 조회합니다. 관리자는 시스템에서 실행된 모든 쿼리를, 일반 사용자 계정은 자신이 실행한 쿼리만 조회할 수 있습니다.

#### 문법

system queries

#### 설명

이 명령은 쿼리 실행 내역을 출력합니다. 레코드는 다음과 같은 필드로 구성됩니다.

* id(정수): 쿼리 식별자
* query\_string(문자열): 쿼리 문자열
* is\_eof(불리언): 쿼리 종료 여부(true: 종료됨, false: 실행 중)
* is\_end(불리언): 쿼리 종료 여부(true: 종료됨, false: 실행 중)
* is\_cancelled(불리언): 쿼리 취소 여부(true: 취소됨, false: 취소되지 않음)
* start\_time(정수): 쿼리 시작 시각 (단위: epoch)
* finish\_time(정수): 쿼리 종료 시각 (단위: epoch)
* last\_started(시간): 마지막 리프레시 시각
* elapsed(정수): 쿼리 실행 소요 시간 (단위: ms), 쿼리가 시작되지 않은 경우, null
* background(불리언): 백그라운드 실행 여부(true: 백그라운드 쿼리, false: 백그라운드 쿼리 아님)
* commands(개체): 세부 명령어별 실행 상태
* sub\_queries(배열): 서브 쿼리 목록
* is\_scheduled\_query(불리언): 예약된 쿼리 여부(true: 예약된 쿼리, false: 예약된 쿼리 아님)
* login\_name(문자열): 쿼리 실행 계정
* remote\_ip(문자열): 쿼리 실행 계정의 접속 IP 주소
* rows(정수): 쿼리 실행 결과 반환된 레코드 개수

쿼리문에서 system queries 명령 결과를 참조할 때 is\_end 대신에 is\_eof를 이용하십시오. is\_end는 하위호환성을 위해 남겨둔 레거시 필드입니다.

### system streams

시스템의 스트림 목록을 조회합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

system streams

#### 설명

출력 필드는 다음과 같습니다.

* source\_type: 데이터 원본 유형으로, logger(로그 수집기), table(테이블), stream (스트림 쿼리) 중 하나의 값으로 표시
* name: 스트림 쿼리 이름
* running: 실행 여부(true: 실행 중, false: 멈춤)
* enabled: 활성화 여부(true: 활성화 상태, false: 비활성화 상태)
* async: 비동기 모드(true: 비동기 모드, false: 동기 모드)
* description: 스트림에 대한 설명
* interval: 리프레시 주기(단위: ms)
* query\_string: 쿼리 문자열
* input\_count: 입력 건수
* output\_count: 출력 건수
* owner: 쿼리 실행 계정
* created\_at: 생성일시
* modified\_at: 수정일시
* last\_refresh\_at: 마지막 리프레시 시각
* input\_tables: 입력으로 들어오는 테이블 목록
* input\_loggers: 입력으로 들어오는 로그 수집기 목록
* input\_streams: 입력으로 들어오는 스트림 목록

스트림으로 들어오는 데이터는 다음 명령어들로 확인할 수 있습니다: logger, table, stream.

### system ceptopics

현재 등록된 이벤트 컨텍스트 주제별 통계를 조회합니다. CEP는 복합 이벤트 처리(Complex Event Processing)의 약어입니다.

#### 문법

system ceptopics

#### 설명

출력 필드는 다음과 같습니다.

* **topic**: 이벤트 컨텍스트 주제
* **count**: 현재 존재하는 이벤트 컨텍스트 수량

#### 더 알아보기

* [evtctxlist](https://docs.logpresso.com/ko/query/evtctxlist-command)
* [evtctxadd](https://docs.logpresso.com/ko/query/evtctxadd-command)
* [evtctxdel](https://docs.logpresso.com/ko/query/evtctxdel-command)
* [evtctxdrop](https://docs.logpresso.com/ko/query/evtctxdrop-command)
* [evtctxget()](https://docs.logpresso.com/ko/query/evtctxget-function)
* [evtctxgetvar()](https://docs.logpresso.com/ko/query/evtctxgetvar-function)
* [evtctxsetvar()](https://docs.logpresso.com/ko/query/evtctxsetvar-function)

### system cepclocks

로그 타임스탬프를 이용한 외부 클럭을 CEP 컨텍스트 만료 시각과 타임아웃 기준으로 사용하는 경우, 현재까지 등록된 호스트별 클럭을 조회합니다.

#### 문법

system cepclocks

#### 설명

출력 필드는 다음과 같습니다.

* host: 호스트 이름
* time: 호스트 클럭
* timeout\_queue\_len: 타임아웃 대기열 길이
* expire\_queue\_len: 만료 대기열 길이

## PCAP 디바이스 관리

### system pcapdevices

로그프레소에서 사용 할 수 있는 PCAP 네트워크 인터페이스 목록을 조회합니다.

#### 문법

system pcapdevices

#### 설명

출력 필드는 다음과 같습니다.

* name: 네트워크 인터페이스 이름(문자열)
* description: 네트워크 인터페이스에 대한 설명(문자열)
* ip: 네트워크 인터페이스에 할당된 IP 주소(IP 주소 개체)
* mac: 네트워크 인터페이스의 MAC 주소
* subnet: 연결된 네트워크의 네트워크 주소
* netmask: 연결된 네트워크의 넷마스크

## 센트리 관리

### system sentries

로그프레소 서버에 등록된 모든 센트리 상태 정보를 조회합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

system sentries

#### 설명

센트리 프로세스 기본 정보(guid, 연결 상태, 설치 경로, Java 버전), 설치 서버의 정보 및 성능치 스냅샷(OS, CPU, Memory, 네트워크) 등을 확인할 수 있습니다.

출력 필드

이 명령어의 출력 필드는 다음과 같습니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자 |
| host\_name | 문자열 | 호스트 이름 |
| remote\_ip | IP 주소 | 센트리의 IP 주소 |
| is\_connected | 불리언 | 접속된 경우 true, 단절된 경우 false |
| description | 문자열 | 센트리에 대한 설명 정보(있는 경우) |
| cpu\_usage | 문자열 | cpu 사용량 (cpu\_kernel + cpu\_user) |
| mem\_usage | 실수 | 현재 메모리 사용률(%) |
| disk\_usage | 실수 | 디스크 사용량(byte) |
| nic\_rx\_usage | 실수 | 네트워크 수신 패킷량(receive) |
| nic\_tx\_usage | 실수 | 네트워크 송신 패킷량(transmit) |
| user\_dir | 문자열 | 실행 디렉터리 위치 |
| cpu\_kernel | 실수 | 커널의 CPU 사용률(%) |
| cpu\_user | 실수 | 유저 프로세스의 CPU 사용률(%) |
| phy\_used | 정수 | 물리 메모리 사용량(byte) |
| phy\_free | 정수 | 물리 메모리 잔여량(byte) |
| phy\_total | 정수 | 물리 메모리 총 사이즈(byte) |
| swap\_used | 정수 | 스왑 메모리 사용량(byte) |
| swap\_free | 정수 | 스왑 메모리 잔여량(byte) |
| swap\_total | 정수 | 스왑 메모리 총 사이즈(byte) |
| last\_connect\_at | 날짜 | 마지막 연결 시각 |
| os | 문자열 | 운영체제 이름 |
| os\_ver | 문자열 | 운영체제 버전 |
| arch | 문자열 | 아키텍처 |
| jvm\_name | 문자열 | 자바 가상 머신(JVM) 이름 |
| jvm\_version | 문자열 | 자바 가상 머신(JVM) 버전 |
| ip\_addrs | 배열 | 전체 할당된 IP 주소 배열 |
| disks | 배열 | 전체 디스크 사용량을 total, available, used로 구분해 표시(byte) |
| nics | 배열 | 네트워크 카드 리스트 및 송/수신량 |

### sentry-arp-cache

센트리의 ARP 캐시를 조회합니다.

#### 문법

sentry-arp-cache [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 guid를 입력 레코드로 받아서 센트리의 ARP 캐시 정보를 요청하는 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |

출력 필드

이 명령어는 원본 레코드의 필드에 아래의 필드를 추가하여 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| type | 문자열 | ARP 캐시 엔트리 유형(static, dynamic, invalid) |
| ip | IP 주소 | IP 주소 |
| mac | 문자열 | MAC 주소 |

오류가 발생하면 \_error 필드에 오류 내용을 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 ARP 캐시 조회

sentry | fields guid | sentry-arp-cache

### sentry-bundles

센트리의 번들 목록을 조회합니다.

#### 문법

sentry-bundles [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 guid를 입력 레코드로 받아서 센트리의 번들 목록 정보를 요청하는 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |

출력 필드

이 명령어는 원본 레코드의 필드에 아래의 필드를 추가하여 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| bundle\_id | 64비트 정수 | 번들 ID |
| symbolic\_name | 문자열 | 번들 심볼릭 이름 |
| version | 문자열 | 번들 버전 |
| state | 문자열 | 번들 상태 |
| last\_modified | 날짜 | 마지막 변경 일시 |
| integrity | 문자열 | 무결성 상태 |

state 필드의 번들 상태 값은 다음과 같이 분류됩니다:

|  |  |
| --- | --- |
| 번들 상태 | 의미 |
| ACTIVE | 번들이 정상적으로 시작됨 |
| INSTALLED | 번들의 의존성이 확보되지 않음 |
| RESOLVED | 번들의 의존성이 확보되었으나 시작되지 않음 |
| STARTING | 번들을 시작하는 중 |
| STOPPING | 번들을 정지하는 중 |
| UNINSTALLED | 번들이 삭제됨 |

integrity 필드의 무결성 상태 값은 다음과 같이 분류됩니다:

|  |  |
| --- | --- |
| 무결성 상태 | 의미 |
| no signature | 디지털 서명된 해시 값이 존재하지 않음 |
| hash error | 번들 파일에 대한 SHA-512 해시 실패 (I/O 오류 등) |
| verified | 디지털 서명된 해시 값과 일치함 |
| modified | 디지털 서명된 해시 값과 일치하지 않음 (변조됨) |

오류가 발생하면 \_error 필드에 오류 내용을 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 번들 목록 조회

sentry | fields guid | sentry-bundles

### sentry-jstack

센트리의 스레드별 스택 상태를 조회합니다. 수집기의 내부 동작 상태 등을 원격으로 진단하는데 사용합니다.

#### 문법

sentry-jstack [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 guid를 입력 레코드로 받아서 센트리의 스레드별 스택 상태를 요청하는 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |

출력 필드

이 명령어는 원본 레코드의 필드에 아래의 필드를 추가하여 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| tid | 64비트 정수 | 스레드 ID |
| name | 문자열 | 스레드 이름 |
| state | 문자열 | 스레드 상태(RUNNABLE, BLOCKED, TIMED\_WAITING, WAITING) |
| stacktrace | 문자열 | 스레드 스택 |

state 필드의 스레드 상태 값은 다음과 같이 분류됩니다:

|  |  |
| --- | --- |
| 스레드 상태 | 의미 |
| RUNNABLE | 언제든 실행 가능한 스레드 |
| BLOCKED | 사용할 객체의 락이 풀릴 때까지 대기 중인 스레드 |
| TIMED\_WAITING | 주어진 시간 동안 대기 중인 스레드 |
| WAITING | 다른 스레드와 동기화를 위해 대기 중인 스레드 |

오류가 발생하면 \_error 필드에 오류 내용을 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 스레드 스택 조회

sentry | fields guid | sentry-jstack

### sentry-logger-configs

센트리의 로거 설정을 조회합니다.

#### 문법

sentry-logger-configs [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)와 로거 이름(name)을 입력 레코드로 받아서 센트리의 로거 설정 정보를 요청하는 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid, name 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |
| name | 문자열 | 센트리 이름 공간 내 유일한 로거 식별자 |

출력 필드

이 명령어는 원본 레코드의 필드에 아래의 필드를 추가하여 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| configs | 맵 | 수집 설정 내역에 대한 키-값 쌍 |

오류가 발생하면 \_error 필드에 오류 내용을 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| name should be not null | 입력 레코드의 name 필드 값이 null |
| name should be string | 입력 레코드의 name 필드 값이 문자열이 아님 |
| name should be non empty string | 입력 레코드의 name 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 로거 설정 조회

sentry | fields guid | sentry-loggers | sentry-logger-configs

### sentry-logger-connect

센트리의 특정한 로거에서 수집된 로그를 로그프레소 서버로 전송하도록 설정합니다.

#### 문법

sentry-logger-connect [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)와 로거 이름(name)을 입력 레코드로 받아서 센트리가 지정된 로거에서 수집된 데이터를 로그프레소 서버로 전송하도록 센트리에게 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid, name 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |
| name | 문자열 | 센트리 이름 공간 내 유일한 로거 식별자 |

출력 필드

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| name should be not null | 입력 레코드의 name 필드 값이 null |
| name should be string | 입력 레코드의 name 필드 값이 문자열이 아님 |
| name should be non empty string | 입력 레코드의 name 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

접속되어 있는 모든 리눅스 센트리에 wtmp 수집기를 생성하고 로그를 원격 전송하도록 설정합니다.

sentry| search os == "Linux" and is\_connected| eval name = "wtmp\_linux"| eval factory\_name = "wtmp"| eval configs = dict("path", "/var/log/wtmp", "server", "linux", "dst\_ip", remote\_ip)| fields guid, name, factory\_name, configs| sentry-logger-create| sentry-logger-connect

### sentry-logger-create

센트리에 로거를 생성합니다.

#### 문법

sentry-logger-create [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리에 로거를 생성합니다. 입력 레코드에 센트리 식별자 guid, 수집 유형 이름 factory\_name, 로거 이름 name, 설명 description, 설정 configs, 테이블 table\_name, 호스트 태그 host\_tag 필드가 필요합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid, name, factory\_name, configs 필드 값이 반드시 필요합니다.

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 필수 | 설명 |
| guid | 문자열 | O | 센트리 고유 식별자(자바의 GUID 형식과 무관) |
| name | 문자열 | O | 센트리 이름 공간 내 유일한 로거 식별자 |
| description | 문자열 | X | 로거 설명 |
| factory\_name | 문자열 | O | 로거 팩토리 식별자 |
| configs | 맵 | O | 로거 설정 |
| table\_name | 문자열 | X | 수집된 데이터를 저장할 테이블의 이름 |
| host\_tag | 문자열 | X | 호스트 태그(\_host 필드 태깅 값) |

* factory\_name: 로그프레소 셸에서 logapi.loggerFactories 명령으로 로거 팩토리 목록을 조회할 수 있습니다. 각 시스템은 운영체제, 설치된 앱에 따라 사용 가능한 로거 팩토리가 다를 수 있습니다.
* configs: [sentry-logger-configs](https://docs.logpresso.com/ko/query/sentry-logger-configs-command) 명령으로 기존에 설정된 로거의 설정을 참고하여 지정합니다.
* table\_name: 테이블 이름을 지정하지 않은 로거는 스트리밍 전용으로 사용됩니다. 스트림 쿼리에서 데이터를 실시간으로 가공한 후에 테이블에 저장할 수도 있습니다.

출력 필드

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| name should be not null | 입력 레코드의 name 필드 값이 null |
| name should be string | 입력 레코드의 name 필드 값이 문자열이 아님 |
| name should be non empty string | 입력 레코드의 name 필드 값이 빈 문자열임 |
| factory\_name should be not null | 입력 레코드의 factory\_name 필드 값이 null |
| factory\_name should be string | 입력 레코드의 factory\_name 필드 값이 문자열이 아님 |
| factory\_name should be non empty string | 입력 레코드의 factory\_name 필드 값이 빈 문자열임 |
| unsupported factoryname: factory\_name | factory\_name 로거 팩토리는 지원되지 않음 |
| configs should be not null | 입력 레코드의 configs 필드 값이 null |
| configs should be dict type | 입력 레코드의 configs 필드 값이 맵 타입이 아님 |
| all values of configs should be string type | 입력 레코드의 configs 맵의 키/값 쌍이 문자열이 아님 |
| missing config key: name | 필수 설정 키(name)가 누락되었음 |
| table\_name should be non empty string | 입력 레코드의 table\_name 필드 값이 빈 문자열임 |
| host\_tag should be non empty string | 입력 레코드의 host\_tag 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

접속되어 있는 모든 리눅스 센트리에 wtmp 로거를 생성합니다.

sentry| search os == "Linux" and is\_connected| eval name = "wtmp\_linux"| eval factory\_name = "wtmp"| eval configs = dict("path", "/var/log/wtmp", "server", "linux", "dst\_ip", remote\_ip)| fields guid, name, factory\_name, configs| sentry-logger-create

### sentry-logger-deploy

센트리에 로거 프로비저닝 프로파일에 정의된 로거 집합을 일괄 생성합니다.

#### 문법

sentry-logger-deploy

#### 설명

로그프레소는 클라우드에서 동적으로 생성되는 인스턴스의 로그를 수집할 수 있도록, 센트리 접속 시 로거를 자동으로 설정하는 로거 프로비저닝 프로파일 기능을 지원합니다. 하지만 접속 시 로거 프로비저닝을 자동으로 실행하려면, 센트리를 시작할 때 환경 변수 logpresso.sentry.logger\_provisioning\_profile를 지정해야 합니다. 이 명령어는 로그프레소 센트리 부팅 시 로거 프로비저닝 프로파일을 지정하지 않더라도 특정 로거 프로비저닝 프로파일에서 정의한 로거 집합을 지정한 로그프레소 센트리에 자동으로 구성합니다.

이 명령어는 프로비저닝 시작 요청만 전달하고 대기하지 않으므로 명령어가 종료된 시점에도 로거 구성은 대기 중이거나 처리 중일 수 있습니다. 또한 로거 프로비저닝을 실행하던 중에 실패하더라도 시스템 로그 기록 외에 별도로 통지되지 않습니다.

로그프레소 셸에서 logpresso.loggerProvisioningTasks 명령으로 대기 중인 로거 프로비저닝 태스크 수를 확인할 수 있습니다.

입력 필드

이 명령어는 입력 레코드에 guid, profile\_guid 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바 GUID 형식과 무관) |
| profile\_guid | 문자열 | 로거 프로비저닝 프로파일 식별자(32자 GUID 형식) |

출력 필드

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| profile\_guid is null | 입력 레코드의 profile\_guid 필드 값이 null |
| profile\_guid should be string | 입력 레코드의 profile\_guid 필드 값이 문자열이 아님 |
| profile\_guid should be non empty string | 입력 레코드의 profile\_guid 필드 값이 빈 문자열임 |
| not connected | 지정된 센트리가 접속된 상태가 아님 |
| profile not found: profile\_guid | 지정된 로거 프로비저닝 프로파일이 존재하지 않음 |

#### 사용 예

윈도우 센트리를 대상으로 윈도우 로거 자동 구성 명령

sentry| search os == "Windows\*"| eval profile\_guid="448c0422-7a30-42ef-b73a-e855e538f779"| sentry-logger-deploy

### sentry-logger-disconnect

센트리의 특정한 로거에서 수집된 로그를 로그프레소 서버로 전송하지 않도록 연결을 해제합니다.

#### 문법

sentry-logger-disconnect [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)와 로거 이름(name)을 입력 레코드로 받아서 센트리가 지정된 로거에서 수집된 데이터를 로그프레소 서버로 전송하지 않도록 요청하는 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid, name 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |
| name | 문자열 | 센트리 이름 공간 내 유일한 로거 식별자 |

출력 필드

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| name should be not null | 입력 레코드의 name 필드 값이 null |
| name should be string | 입력 레코드의 name 필드 값이 문자열이 아님 |
| name should be non empty string | 입력 레코드의 name 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

접속되어 있는 모든 센트리에서 지정된 이름의 로거를 일괄적으로 연결 해제합니다.

sentry | sentry-loggers | search name == "wtmp\_linux" | sentry-logger-disconnect

### sentry-logger-remove

센트리에서 특정한 로거를 삭제합니다.

#### 문법

sentry-logger-remove [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)와 로거 이름(name)을 입력 레코드로 받아서 센트리의 특정한 로거를 삭제하도록 센트리에게 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid, name 필드 값이 필요합니다.

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 이름 | 비고 |
| guid | 문자열 | 센트리 고유 식별자 | GUID 형식과 무관 |
| name | 문자열 | 로거 식별자 | 센트리 이름 공간 내 유일한 로거 식별자 |

출력 필드

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| name should be not null | 입력 레코드의 name 필드 값이 null |
| name should be string | 입력 레코드의 name 필드 값이 문자열이 아님 |
| name should be non empty string | 입력 레코드의 name 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

접속되어 있는 모든 센트리에서 지정된 이름의 로거를 일괄 삭제합니다.

sentry | sentry-loggers | search name == "wtmp\_linux" | sentry-logger-remove

이 작업은 되돌릴 수 없습니다. 마지막 쿼리 명령을 '# sentry-logger-remove'으로 주석 처리하여 삭제 대상을 먼저 확인하세요.

### sentry-logger-set-interval

센트리에 있는 특정한 로거의 수집 주기를 재설정합니다.

#### 문법

sentry-logger-set-interval [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)와 로거 이름(name), 수집 주기(interval)을 입력 레코드로 받아서 센트리의 특정한 로거의 수집 주기를 재설정하도록 센트리에게 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid, name, interval 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |
| name | 문자열 | 센트리 이름 공간 내 유일한 로거 식별자 |
| interval | 32비트 정수 | 밀리초 단위 수집 주기 |

출력 필드

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| name should be not null | 입력 레코드의 name 필드 값이 null |
| name should be string | 입력 레코드의 name 필드 값이 문자열이 아님 |
| name should be non empty string | 입력 레코드의 name 필드 값이 빈 문자열임 |
| interval should be not null | 입력 레코드의 interval 필드 값이 null |
| interval should be integer | 입력 레코드의 interval 필드 값이 정수가 아님 |
| logger not found: name | 지정된 이름의 로거를 찾을 수 없음 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |
| sentry method not found: setLoggerInterval | 센트리가 3.10.2106.0 이전 버전이어서 이 기능을 지원하지 않음 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

접속되어 있는 모든 센트리에서 지정된 이름의 로거를 대상으로 수집 주기를 5초로 변경합니다.

sentry| sentry-loggers| search name == "wtmp\_linux"| eval interval = 5000| sentry-logger-set-interval

### sentry-logger-set-schedule

센트리에 있는 특정한 로거의 크론 형식 수집 일정을 재설정합니다.

#### 문법

sentry-logger-set-schedule [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)와 로거 이름(name), 수집 일정(cron\_schedule)을 입력 레코드로 받아서 센트리에 있는 특정한 로거의 크론 형식 수집 일정을 재설정하도록 센트리에게 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid, name, cron\_schedule 필드 값이 필요합니다.

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 필수 | 설명 |
| guid | 문자열 | O | 센트리 고유 식별자(자바 GUID 형식과 무관) |
| name | 문자열 | O | 센트리 이름 공간 내 유일한 로거 식별자 |
| cron\_schedule | 문자열 | X | 수집 주기(크론 형식의 수집 일정. null인 경우 기존 수집 일정 삭제) |

* cron\_schedule: 입력 방식은 유닉스 계열 운영체제의 크론 설정과 동일합니다. 리눅스 시스템에서 "man 5 crontab" 명령으로 일정 설정에 필요한 항목을 확인해보십시오.

출력 필드

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| name should be not null | 입력 레코드의 name 필드 값이 null |
| name should be string | 입력 레코드의 name 필드 값이 문자열이 아님 |
| name should be non empty string | 입력 레코드의 name 필드 값이 빈 문자열임 |
| cron\_schedule should be string | 입력 레코드의 cron\_schedule 필드 값이 문자열이 아님 |
| wrong cron expression format: expr | 입력 레코드의 cron\_schedule 필드 값이 유효하지 않은 크론 일정 형식임 |
| logger not found: name | 지정된 이름(name)의 로거를 찾을 수 없음 |
| logger is running: name | 실행 중인 로거(name)의 수집 일정을 변경할 수 없음 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |
| sentry method not found: setLoggerSchedule | 센트리가 3.10.2106.0 이전 버전이어서 이 기능을 지원하지 않음 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

접속되어 있는 모든 센트리에서 지정된 이름의 로거를 매시 0분마다 실행하도록 설정

sentry| sentry-loggers| search name == "wtmp\_linux"| eval cron\_schedule="0 \* \* \* \*"| sentry-logger-set-schedule

### sentry-logger-set-time-range

센트리에 있는 특정한 로거의 수집 실행 시간을 재설정합니다.

#### 문법

sentry-logger-set-time-range [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)와 로거 이름(name), 수집 시작 시각(start\_time), 수집 종료 시각(end\_time)을 입력 레코드로 받아서 센트리에 있는 특정한 로거의 크론 형식 수집 주기를 재설정하도록 센트리에게 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid, name, start\_time, end\_time 필드 값이 필요합니다.

|  |  |  |  |
| --- | --- | --- | --- |
| 필드 | 타입 | 필수 | 설명 |
| guid | 문자열 | O | 센트리 고유 식별자(자바 GUID 형식과 무관) |
| name | 문자열 | O | 센트리 이름 공간 내 유일한 로거 식별자 |
| start\_time | 문자열 | X | 수집 시작 시각(형식: HH:mm) |
| end\_time | 문자열 | X | 수집 종료 시각(형식: HH:mm) |

start\_time, end\_time 필드 값이 모두 null이면 대상 로거의 기존 수집 허용 시간 범위 설정을 제거합니다.

출력 필드

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| name should be not null | 입력 레코드의 name 필드 값이 null |
| name should be string | 입력 레코드의 name 필드 값이 문자열이 아님 |
| name should be non empty string | 입력 레코드의 name 필드 값이 빈 문자열임 |
| invalid start\_time format | 입력 레코드의 start\_time 필드 값이 유효한 HH:mm 형식 값이 아님 |
| invalid end\_time format | 입력 레코드의 end\_time 필드 값이 유효한 HH:mm 형식 값이 아님 |
| start\_time is not null but end\_time is null | 입력 레코드에 start\_time이 설정되었으나 end\_time이 설정되지 않음 |
| end\_time is not null but start\_time is null | 입력 레코드에 end\_time이 설정되었으나 start\_time이 설정되지 않음 |
| logger not found: name | 지정된 이름의 로거를 찾을 수 없음 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

접속되어 있는 모든 센트리에서 지정된 이름의 로거를 대상으로 밤 10시부터 오전 6시까지만 수집하도록 설정합니다.

sentry| sentry-loggers| search name == "weblog"| eval start\_time="22:00", end\_time="06:00"| sentry-logger-set-time-range

### sentry-logger-start

센트리에 있는 특정한 로거가 일정 주기마다 수집을 실행하도록 활성화합니다.

#### 문법

sentry-logger-start [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)와 로거 이름(name), 수집 주기(interval)을 입력 레코드로 받아서 센트리에 있는 특정한 로거가 일정 주기마다 수집을 실행할 수 있게 활성화하도록 센트리에게 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에에 guid, name, interval 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |
| name | 문자열 | 센트리 이름 공간 내 유일한 로거 식별자 |
| interval | 32비트 정수 | 밀리초 단위 수집 주기 |

출력 필드

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| name should be not null | 입력 레코드의 name 필드 값이 null |
| name should be string | 입력 레코드의 name 필드 값이 문자열이 아님 |
| name should be non empty string | 입력 레코드의 name 필드 값이 빈 문자열임 |
| interval should be not null | 입력 레코드의 interval 필드 값이 null |
| interval should be integer | 입력 레코드의 interval 필드 값이 정수가 아님 |
| logger is already running | 지정된 로거는 이미 실행 중 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 로거가 5초마다 수집을 실행하도록 활성화합니다.

sentry| sentry-loggers| eval interval = 5000| fields guid, name, interval| sentry-logger-start

### sentry-logger-stop

센트리에 있는 특정 로거를 비활성화합니다.

#### 문법

sentry-logger-stop [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)와 로거 이름(name)을 입력 레코드로 받아서 센트리에 있는 특정한 로거를 비활성화하도록 센트리에게 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid, name 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |
| name | 문자열 | 센트리 이름 공간 내 유일한 로거 식별자 |

출력 필드

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| name should be not null | 입력 레코드의 name 필드 값이 null |
| name should be string | 입력 레코드의 name 필드 값이 문자열이 아님 |
| name should be non empty string | 입력 레코드의 name 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 로거를 비활성화합니다.

sentry | sentry-loggers | fields guid, name | sentry-logger-stop

### sentry-loggers

센트리의 로거 목록을 조회합니다.

#### 문법

sentry-loggers [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)를 입력 레코드로 받아서 해당 센트리의 로거 목록을 요청하도록 센트리에게 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |

출력 필드

이 명령어는 원본 레코드의 필드에 아래의 필드를 추가하여 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| name | 문자열 | 로거 식별자 |
| description | 문자열 | 로거 설명 |
| factory\_name | 문자열 | 로거 유형 식별자(예: syslog, wtmp) |
| status | 문자열 | 로거 상태(예: running, stopped) |
| interval | 32비트 정수 | 수집 주기(단위: 밀리초) |
| cron\_schedule | 문자열 | 크론 형식 수집 일정 |
| transformer | 문자열 | 트랜스포머 식별자 |
| stop\_reason | 문자열 | 로거 중지 사유 |
| log\_count | 64비트 정수 | 수집 건수(버리기 전 수집된 전체 건수) |
| drop\_count | 64비트 정수 | 버린 건수 |
| log\_volume | 64비트 정수 | 수집 용량(버리기 전 수집된 전체 용량. 단위: 바이트) |
| drop\_volume | 64비트 정수 | 버린 용량(단위: 바이트) |
| last\_start\_at | 날짜 | 로거 활성화한 최근 시각 |
| last\_run\_at | 날짜 | 수집을 실행한 최근 시각 |
| last\_log\_at | 날짜 | 마지막으로 수집된 로그의 \_time 필드 값 |
| last\_write\_at | 날짜 | 로그를 출력한 최근 시스템 시각(수집 유휴 탐지용) |
| start\_time | 문자열 | 수집 시작 시각(형식: HH:mm) |
| end\_time | 문자열 | 수집 종료 시각(형식: HH:mm) |
| failure | 문자열 | 수집 중 기록된 로거 중지 사유 |

failure 필드의 로거 중지 사유는 다음과 같이 분류됩니다:

|  |  |
| --- | --- |
| 로거 중지 사유 | 의미 |
| USER\_REQUEST | 사용자의 비활성화 요청에 따라 중지함 |
| SYSTEM\_REQUEST | 이중화 동작 등 시스템 요청에 따라 중지함 |
| LOW\_DISK | 디스크 용량 고갈로 인해 중지함 |
| TRANSFORMER\_DEPENDENCY | 로거가 의존하는 트랜스포머가 시작되지 않음 |
| FACTORY\_DEPENDENCY | 로거가 정의된 로거 팩토리가 시작되지 않음. 번들이 시작되지 않은 경우 |
| STOP\_EXCEPTION | 저장할 테이블이 존재하지 않거나 스토리지가 읽기 전용 모드에 진입한 경우 |
| LOGGER\_EXCEPTION | 수집 도중에 예상하지 못한 예외 발생 |
| LOGGER\_DEPENDENCY | 로거가 의존하는 다른 로거가 시작되지 않음 |

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 현재 로거 목록 조회

sentry | fields guid | sentry-loggers

### sentry-netstat

센트리의 네트워크 통신 현황 정보를 조회합니다.

#### 문법

sentry-netstat [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)를 입력 레코드로 받아서 네트워크 통신 현황 정보를 조회하도록 센트리에게 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |

출력 필드

이 명령어는 원본 레코드의 필드에 아래의 필드를 추가하여 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| pid | 32비트 정수 | 프로세스 ID |
| protocol | 문자열 | 프로토콜(tcp, tcp6, udp, udp6) |
| local\_ip | IP 주소 | 로컬 IP 주소 |
| local\_port | 32비트 정수 | 로컬 포트 |
| remote\_ip | IP 주소 | 원격 IP 주소 |
| remote\_port | 32비트 정수 | 원격 포트 |
| state | 문자열 | 연결 상태(LISTEN, ESTABLISHED, TIMEWAIT) |

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 현재 네트워크 연결 목록 조회

sentry | fields guid | sentry-netstat

### sentry-processes

센트리의 프로세스 목록을 조회합니다.

#### 문법

sentry-processes [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 센트리 식별자(guid)를 입력 레코드로 받아서 프로세스 목록 정보를 조회하도록 센트리에게 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |

출력 필드

이 명령어는 원본 레코드의 필드에 아래의 필드를 추가하여 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| pid | 32비트 정수 | 프로세스 ID |
| name | 문자열 | 프로세스 이름 |
| cpu\_usage | 32비트 정수 | CPU 사용율(단위: %) |
| working\_set | 64비트 실수 | 물리 메모리 사용량(단위: 바이트) |

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 프로세스 목록 조회

sentry | fields guid | sentry-processes

### sentry-routing-table

센트리의 라우팅 테이블 엔트리 목록을 조회합니다.

#### 문법

sentry-routing-table [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 guid를 입력 레코드로 받아서 센트리에게 라우팅 테이블 정보를 요청하는 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |

출력 필드

이 명령어는 원본 레코드의 필드에 아래의 필드를 추가하여 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| type | 문자열 | 라우팅 엔트리 유형(Direct, Indirect) |
| protocol | 문자열 | 프로토콜(Local, NetMgmt) |
| destination | IP 주소 | 목적지 IP 주소 |
| mask | IP 주소 | 넷마스크 |
| forward | IP 주소 | 게이트웨이 IP 주소 |
| metric | 32비트 정수 | 경로 비용 메트릭 |

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 현재 라우팅 테이블 엔트리 목록 조회

sentry | fields guid | sentry-routing-table

### sentry-top-threads

센트리에서 CPU 사용량이 많은 스레드의 스택 상태를 조회합니다. 센트리의 CPU 부하 원인을 원격으로 진단할 때 사용합니다.

#### 문법

sentry-top-threads [timeout=INT]

선택 매개변수

**timeout=INT**

초 단위 RPC 타임아웃(기본값: 30초)

#### 설명

이 명령어는 guid를 입력 레코드로 받아서 센트리에게 CPU 사용량이 많은 스레드의 스택 정보를 요청하는 비동기 RPC 메시지를 전송합니다.

로그프레소 환경 변수 logpresso.core.sentry\_rpc\_parallel은 비동기 RPC 요청 메시지 대기열의 길이를 지정하는데 사용됩니다. 이 환경변수의 기본값은 100이고, 이 환경 변수 값을 변경함으로써 비동기 RPC 요청 대기열의 길이를 조정할 수 있습니다.

RPC 요청 메시지 대기열은 선입선출 방식(first in, first out, 줄여서 FIFO) 방식으로 동작합니다. 예를 들어 로그프레소 서버가 메시지를 보낼 센트리의 개수가 150대라면, 서버는 먼저 100대의 센트리에게 RPC 메시지를 전송하고 RPC 응답을 기다립니다. 이 때 대기 시간이 timeout으로 지정된 시간(기본값은 30초)을 초과하면 해당 센트리로부터 오류가 발생한 것으로 간주합니다. 100대의 센트리 중에서 32대가 응답하거나 타임아웃되면 추가적으로 32대의 센트리에게 RPC 메시지를 전송합니다.

이 명령어는 센트리의 RPC 응답 메시지를 수신하는 순서대로 결과를 출력하기 때문에 출력 레코드의 순서는 입력 레코드의 순서와 다를 수 있습니다. 레코드의 순서에 의존하지 말고 출력 데이터 자체를 이용하십시오.

입력 필드

이 명령어는 입력 레코드에 guid 필드 값이 필요합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| guid | 문자열 | 센트리 고유 식별자(자바의 GUID 형식과 무관) |

출력 필드

이 명령어는 원본 레코드의 필드에 아래의 필드를 추가하여 출력합니다.

|  |  |  |
| --- | --- | --- |
| 필드 | 타입 | 설명 |
| tid | 64비트 정수 | 스레드 ID |
| name | 문자열 | 스레드 이름 |
| state | 문자열 | 스레드 상태(RUNNABLE, BLOCKED, TIMED\_WAITING, WAITING) |
| priority | 32비트 정수 | 실행 우선순위(1-10 범위, 기본값 5) |
| usage | 64비트 정수 | 최근 1초 중 나노 초 단위의 CPU 사용 시간 |
| stacktrace | 문자열 | 스레드 스택 |

이 명령어는 오류가 발생하면 원본 레코드의 필드에 \_error 필드를 추가하여 출력합니다. 발생할 수 있는 오류는 아래와 같습니다:

|  |  |
| --- | --- |
| 오류 메시지 | 의미 |
| guid is null | 입력 레코드의 guid 필드 값이 null |
| guid should be string | 입력 레코드의 guid 필드 값이 문자열이 아님 |
| guid should be non empty string | 입력 레코드의 guid 필드 값이 빈 문자열임 |
| timeout | RPC 요청 타임아웃 시간을 초과함 |
| disconnected | RPC 요청 처리 중에 연결이 단절됨 |
| not connected | 센트리가 접속된 상태가 아님 |

위에 정의된 \_error 문자열 이외에 센트리가 설치된 시스템 상태에 따라 다른 RPC 예외 메시지가 출력될 수 있습니다.

#### 사용 예

모든 센트리의 고부하 스레드 스택 조회

sentry | fields guid | sentry-top-threads

### sentryswap

로그프레소 서버를 운용하는 서버에 센트리가 설치되어 있을 때 사용할 수 있는 명령어로, 센트리 전송 큐에 스왑되어 있는 데이터를 조회합니다.

#### 문법

sentryswap [base=NAME]

선택 매개변수

**base=NAME**

센트리와 연결된 베이스 서버의 고유 식별 이름을 지정합니다. 지정하지 않으면 모든 센트리의 스왑 데이터를 보여줍니다.

베이스 서버는 센트리로부터 로그를 수신하는 로그프레소 서버를 의미합니다. 센트리가 각각 다른 로그프레소 서버로 데이터를 전송할 수 있습니다. 센트리의 전송 큐는 베이스 서버마다 각각 별도로 구분해 운용합니다.

#### 설명

일반적으로 적체 현상 발생 시 전송 큐에서 대기하고 있는 데이터 건수를 확인하거나, 전송 큐에 있는 데이터를 백업한 후에 스왑을 삭제해서 시스템 연결 상태를 즉시 복구하려는 경우 사용합니다.

출력 필드는 다음과 같습니다.

* \_time: 타임스탬프
* \_logger: 로그 수집기 이름
* 전송 대기 버퍼에 있는 레코드의 키-값 쌍

sentryswap은 로그프레소와 센트리가 하나의 호스트에 설치되어 있을 때 사용할 수 있습니다. 센트리만 설치된 호스트에서 스왑을 확인하려면 로그프레소 셸에서 sentry.swapStats 명령으로 확인하십시오.

## 스레드 및 잠금 상태

### system threads

시스템의 모든 스레드 스택과 락 상태를 조회합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

system threads

#### 설명

출력 필드는 다음과 같습니다.

* tid: 스레드 번호
* name: 스레드 이름
* state: 스레드 상태
* stacktrace: 스레드 스택(jstack과 동일한 형식)

### system topthreads

시스템에서 부하가 걸리는 스레드를 조회합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

system topthreads

#### 설명

가장 부하가 높은 순서대로 스레드 정보를 출력합니다. 출력 필드는 다음과 같습니다.

* tid: 스레드 번호
* name: 스레드 이름
* state: 스레드 상태
* priority: 1부터 10 사이의 우선순위 값. 기본 우선순위는 5.
* usage: 나노 초 단위의 CPU 사용 시간
* stacktrace: 스레드 스택 (jstack과 동일한 형식)

## 페더레이션 노드 관리

로그프레소 클러스터를 구성하는 시스템 노드의 상태와 설정을 조회합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

system nodes

#### 설명

페더레이션 노드 상태는 웹 콘솔에서도 확인할 수 있습니다.

출력 필드는 다음과 같습니다.

* name: 노드 이름
* description: 노드 설명
* address: 노드 주소
* port: 노드 포트번호
* failure: 노드 접속 실패 여부
* paired: 노드 페어 여부
* last\_alive: 마지막으로 응답을 받은 시간
* last\_connect: 마지막으로 연결된 시간
* login\_name: 로그인 계정
* secure: 암호화 통신 여부
* skip\_cert\_check: 서버 인증서 검증 안함 여부
* connect\_timeout: 접속 타임아웃 설정
* read\_timeout: 읽기 타임아웃 설정

이 명령어는 ENT #1863 2018-01-31\_17-38 버전부터 지원합니다.

## 라이선스 관리

### system license-usages

시스템 노드의 라이센스 사용량 상태를 조회합니다. 이 명령어를 실행하려면 관리자 권한이 필요합니다.

#### 문법

system license-usages

#### 설명

출력 필드는 다음과 같습니다.

* node: 노드 이름
* volume: 라이센스 사용량 (byte)
* count: 로그 수집 건수

#### 호환성

system license-usages 명령은 ENT #2241 2019-04-23\_17-20 버전부터 지원합니다.

# 함수

## 참조 함수

### $()

쿼리 매개변수의 값을 반환합니다.

#### 문법

$(EXPR, [DEFAULT\_EXPR])

필수 매개변수

**EXPR**

쿼리 매개변수를 반환하는 표현식. 쿼리 매개변수의 이름은 큰 따옴표 쌍(" ")으로 감싸서 입력하세요.

선택 매개변수

**DEFAULT\_EXPR**

EXPR이 null일 때 함수가 반환할 기본값을 정의하는 표현식이나 값

#### 설명

쿼리 매개변수는 쿼리에서 필요할 때 호출해 사용할 수 있는 변수로, 함수 등을 이용한 표현식을 이용함으로써 동적으로 값을 할당해 쿼리를 실행할 때 유용합니다. [set](https://docs.logpresso.com/ko/query/set-command) 또는 [evalc](https://docs.logpresso.com/ko/query/evalc-command) 명령어는 쿼리 매개변수를 정의할 때 사용됩니다. 쿼리 매개변수에 할당된 값을 쿼리나 프로시저에서 참조하려면 $() 함수를 사용하세요.

[프로시저](https://docs.logpresso.com/ko/query/query-syntax)를 정의할 때 이 함수를 이용함으로써 프로시저가 쿼리 매개변수를 받을 수 있습니다. 단, 프로시저를 호출하는 쿼리와 프로시저는 동일한 매개변수 이름을 사용해야 합니다.

#### 사용 예

현재 시각으로부터 최근 7일간 YOUR\_TABLE 테이블에 기록된 데이터 조회

set from=ago("7d") | set to=str(now()) | table from=$("from") to=$("to") YOUR\_TABLE

쿼리 매개변수 \_from, \_to로 지정한 기간 동안 YOUR\_TABLE 테이블에 기록된 데이터 조회. 쿼리 매개변수의 값이 없으면(null), 현재 시각으로부터 최근 1일간 데이터 검색

table from=$("\_from", ago("1d")) to=$("\_to", now()) YOUR\_TABLE

### field()

필드 이름을 표현식으로 받아 필드 값을 반환합니다. 빈 칸이 들어있는 필드 이름을 참조할 때도 사용됩니다.

#### 문법

field(EXPR)

필수 매개변수

**EXPR**

필드 이름을 반환하는 표현식

#### 사용 예

json "[ {'Registered No.': 1, 'Item':'Fender Precision Bass'}, {'Registered No.': 2, 'Item':'Gibson Jazz'}]" | search field("Registered No.") == 2

### whoami()

현재 쿼리를 실행하는 계정 이름을 반환합니다.

#### 문법

whoami()

#### 설명

프로시저는 소유자의 권한으로 실행하므로, 프로시저 내에서 이 함수를 호출하면 소유자 계정의 이름이 반환됩니다.

#### 사용 예

현재 실행 계정 이름 반환

json "{}" | eval user=whoami() => "root"

## 타입 변환 함수

### array()

인자로 지정한 모든 표현식들을 평가하여 생성한 배열을 반환합니다.

#### 문법

array(EXPR, ...)

**EXPR, ...**

각 배열 항목에 입력할 값을 반환하는 표현식

#### 사용 예

숫자를 구성 요소로 갖는 배열 반환

json "{}" | eval array=array(1) | # 반환 값: [1]

문자열을 구성 요소로 갖는 배열 반환

json "{}" | eval array=array("hello", "world") | # 반환 값: ["hello", "world"]

숫자와 문자열을 구성 요소로 갖는 배열 반환

json "{}" | eval array=array(21 \* 2, "the answer to life, the universe, and everything") | # 반환 값: [42, "the answer to life, the universe, and everything"]

구성 요소가 없는 배열 반환

json "{}" | eval array=array(null) | # 반환 값: [null]

### binary()

문자열을 바이너리 값으로 변환합니다.

#### 문법

binary(STR\_EXPR[, CHARSET])

필수 매개변수

**STR\_EXPR**

바이너리로 변환할 대상 문자열 표현식

선택 매개변수

**CHARSET**

문자열 인코딩 형식(기본값: utf-8). IANA Charset에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

#### 사용 예

json "{}" | eval blob=binary("hello, world!") => 68656c6c6f2c20776f726c6421json "{}" | eval blob=binary(null) => null

### date()

문자열을 날짜 타입으로 변환합니다.

#### 문법

date(DATE\_EXPR, DATE\_FMT, [LOCALE])

필수 매개변수

**DATE\_EXPR**

날짜 타입으로 변환할 원본 문자열 표현식

**DATE\_FMT**

문자열을 파싱할 때 사용할 형식으로 다음과 같은 날짜 지시자를 사용할 수 있습니다.

**날짜 지시자**

|  |  |  |
| --- | --- | --- |
| 지시자 | 설명 | 예제 |
| G | 기원 전/후 | AD |
| y | 연도 | 1996; 96 |
| M | 월 | July; Jul; 07 |
| w | 연 단위 주 | 27 |
| W | 월 단위 주 | 2 |
| d | 연 단위 일 | 189 |
| D | 월 단위 일 | 10 |
| F | 요일 번호 | 2 |
| E | 요일 | Tuesday; Tue |
| u | 요일 숫자(1=월요일, …, 7=일요일 | 1 |
| a | 오전/오후 | PM |
| H | 시간(0-23) | 0 |
| k | 시간(1-24) | 24 |
| K | 오전/오후 시간 (0-11) | 0 |
| h | 오전/오후 시간 (1-12) | 12 |
| m | 분 | 30 |
| s | 초 | 55 |
| S | 밀리초 | 978 |
| z | 시간대(일반 표기) | Pacific Standard Time; PST |
| Z | 시간대(RFC 822 표기) | -0800 |
| X | 시간대(ISO 8601 표기) | -08;-0800;08:00 |

선택 매개변수

**LOCALE**

ISO 639에 지정된 2자리 혹은 3자리 코드 로케일. 지정하지 않으면 en으로 설정됩니다. ISO 639 로케일 코드는 다음 링크를 참조하세요: https://iso639-3.sil.org/code\_tables/639/.

#### 설명

STR 표현식이 null 또는 빈 문자열이면 null을 반환합니다. 문자열 이외의 타입인 경우 문자열로 자동 변환 후 날짜 변환을 시도합니다.

로그프레소는 시간 표현식에 엄격한 유효성 검사를 적용하지 않습니다. 예를 들어 DATE 매개변수에 2020년 13월 34일과 같은 값을 받으면 13월은 한해가 넘어가는 2021년 1월로, 34일은 1월 31일에서 3일을 더한 날짜가 되어 2021년 2월 3일로 계산합니다.

#### 사용 예

json "{}" | eval date=date("2013-06-10 00:30:55.978", "yyyy-MM-dd HH:mm:ss.SSS")=> 2013-06-10 00:30:55+0900json "{}" | eval date=date("2020-01-30T10:11:12.123Z", "yyyy-MM-dd'T'HH:mm:ss.SSSX")=> 2020-01-30 19:11:12+0900json "{}" | eval date=date("6월 1 2020 12:34:56", "MMM dd yyyy HH:mm:ss", "ko")=> 2020-06-01 12:34:56+0900

### dict()

키-값 목록을 입력 받아 생성된 맵(map)을 반환합니다.

#### 문법

dict(KEY, VALUE, ...)

필수 매개변수

**KEY, VALUE, ...**

키와 값을 쉼표(,)로 구분하여 순서대로 입력. 여러 개의 키-값 쌍을 같은 방법으로 반복해 입력할 수 있습니다.

#### 설명

키는 null이 아닌 문자열만 입력할 수 있으며, 다른 키와 중복되면 안 됩니다. 키를 중복으로 입력하면 나중에 입력한 값이 할당됩니다. 값은 모든 타입을 입력할 수 있습니다.

또한 키-값 쌍이 맞지 않으면(매개변수 개수가 홀수일 때) 오류가 발생합니다.

맵은 Java에서 사용하는 데이터 타입으로, Python 같은 언어에서 사용하는 딕셔너리를 의미합니다.

#### 사용 예

json "{}" | eval dict=dict() => {}json "{}" | eval dict=dict("a", "aaa") => {"a":"aaa"}json "{}" | eval dict=dict( "name", "John", "age", 30, "host", ip("1.2.3.4"), "hobby", array("music","movie","sports"), "birthday", date("19800101","yyyyMMdd"))=> {"birthday":"1980-01-01 00:00:00+0900","name":"John","host":"/1.2.3.4","age":30,"hobby":["music","movie","sports"]}

### double()

문자열을 64비트 배정도 실수로 변환합니다.

#### 문법

double(STR\_EXPR)

필수 매개변수

**STR\_EXPR**

실수로 변환할 문자열을 반환하는 표현식

#### 설명

표현식이 null이면 null을 반환합니다. 실수 변환에 실패해도 null을 반환합니다. 표현식이 반환한 값이 문자열이 아닌 경우, 문자열로 자동 변환한 다음 실수 변환을 시도합니다.

#### 사용 예

json "{}" | eval numbers=double("1.2") => 1.2json "{}" | eval numbers=double("0") => 0.0json "{}" | eval numbers=double(0) => 0.0json "{}" | eval numbers=double("invalid") => nulljson "{}" | eval numbers=double(null) => null

### frombase64()

BASE64 문자열을 바이너리로 변환하여 반환합니다.

#### 문법

frombase64(BASE64\_STR)

필수 매개변수

**BASE64\_STR**

BASE64로 인코딩된 문자열

#### 사용 예

json "{}" | eval str=decode(frombase64("aGVsbG8sIHdvcmxkIQ=="))=> "hello, world!"

### fromhex()

16진수 문자열을 바이너리로 변환합니다.

#### 문법

fromhex(STR\_EXPR)

필수 매개변수

**STR\_EXPR**

바이너리로 변환할 문자열. 문자열은 대소문자를 구분하지 않습니다.

다음과 같은 상황에서 null을 반환합니다.

* 입력 값이 16진수 문자열이 아닐 때
* 문자열 길이가 홀수일 때

#### 사용 예

json "{}" | eval blob=fromhex("68656c6c6f20776f726c64")=> 68656c6c6f20776f726c64json "{}" | eval blob=fromhex("616263646") => nulljson "{}" | eval blob=fromhex("test") => nulljson "{}" | eval blob=fromhex(null) => null

### groups()

문자열에서 주어진 정규표현식의 그룹에 매칭되는 항목들을 배열로 반환합니다.

#### 문법

groups(STR\_EXPR, REGEX\_PATTERN)

필수 매개변수

**STR\_EXPR**

추출 대상 원본 문자열 표현식

**REGEX\_PATTERN**

그룹을 포함한 정규표현식 문자열

#### 사용 예

json "{}"| eval array=groups("Mar 29 2004 09:54:39", "(.\*?) (.\*?) (.\*?) ") => [Mar, 29, 2004]

### int()

문자열을 정수로 변환합니다.

#### 문법

int(EXPR)

필수 매개변수

**EXPR**

정수로 변환할 문자열을 반환하는 표현식. 인자는 문자열(string), double, float, IP 주소, 배열 중 하나이어야 합니다.

표현식을 평가할 때 다음과 같이 동작합니다.

* null일 때, null을 반환합니다.
* 문자열을 정수로 변환할 수 없을 때에도 null을 반환합니다.
* 배열일 때, 배열의 각 요소를 정수로 변환합니다.
* 이 외에 다른 타입이 인자로 전달되면 자동 변환을 수행한 다음 정수로 변환합니다.

#### 사용 예

json "{}" | eval numbers=int("1234") => 1234json "{}" | eval numbers=int(1234) => 1234json "{}" | eval numbers=int(ip("0.0.0.1")) => 1json "{}" | eval numbers=int(ip("192.168.0.1")) => -1062731775json "{}" | eval numbers=int(12345.6789) => 12345json "{}" | eval numbers=int(null) => nulljson "{}" | eval numbers=int("invalid") => nulljson "{}" | eval numbers=int(array("1", "abc", "2", 3, array(4)))=> [1, null, 2, 3, null]

### ip()

문자열 표현식을 IP 주소 타입으로 변환합니다.

#### 문법

ip(EXPR)

필수 매개변수

**EXPR**

IP 주소로 변환할 문자열을 반환하는 표현식. 인자는 문자열(string), signed int, long 중 하나이어야 합니다.

표현식을 평가할 때 다음과 같이 동작합니다.

* null일 때, null을 반환합니다.
* IP 주소로 변환할 수 없을 때에도 null을 반환합니다.
* 이 외에 다른 타입이 인자로 전달되면 문자열로 변환하고, IP 주소로 변환을 시도합니다.
* IP 주소 타입은 로그프레소의 기본 데이터 타입 중 하나입니다. IP 주소 타입은 '/'로 시작하며, ipv4 및 ipv6 주소를 모두 표현할 수 있습니다.

#### 사용 예

json "{}" | eval ip=ip("1.2.3.4") => /1.2.3.4json "{}" | eval ip=ip("::1") => /0:0:0:0:0:0:0:1json "{}" | eval ip=ip(4294967295) => /255.255.255.255json "{}" | eval ip=ip(-1062731775) => /192.168.0.1json "{}" | eval ip=ip("invalid") => nulljson "{}" | eval ip=ip(null) => null

### long()

문자열을 64비트 정수로 변환합니다.

#### 문법

long(EXPR)

필수 매개변수

**EXPR**

64비트 정수로 변환할 문자열을 반환하는 표현식. 인자는 문자열(string), int, IP 주소 중 하나이어야 합니다.

표현식을 평가할 때 다음과 같이 동작합니다.

* null일 때, null을 반환합니다.
* 64비트 정수로 변환할 수 없을 때에도 null을 반환합니다.
* 이 외에 다른 타입이 인자로 전달되면 문자열로 자동 변환한 다음에 64비트 정수로 변환합니다.

#### 사용 예

json "{}" | eval numbers=long("1234") => 1234json "{}" | eval numbers=long(1234) => 1234json "{}" | eval numbers=long(ip("0.0.0.1")) => 1json "{}" | eval numbers=long(ip("192.168.0.1")) => 3232235521json "{}" | eval numbers=long(null) => nulljson "{}" | eval numbers=long("invalid") => null

### string()

임의의 표현식을 문자열로 변환하거나, 지정한 형식으로 날짜 형식 문자열을 변환합니다.

#### 문법

string(EXPR)string(DATE\_EXPR, DATE\_FMT[, LOCALE])string(DATE\_EXPR, DATE\_FMT[, TIMEZONE])str(EXPR)str(DATE\_EXPR, DATE\_FMT[, LOCALE])str(DATE\_EXPR, DATE\_FMT[, TIMEZONE])

**EXPR**

문자열로 변환할 값을 반환하는 표현식

**DATE\_EXPR**

날짜 타입으로 변환할 원본 문자열 표현식

**DATE\_FMT**

date()에서 사용하는 것과 동일한 날짜 지시자를 사용하여 문자열을 파싱할 때 사용할 형식을 정의합니다.

**TIMEZONE**

시간대는 'GMT+09', 'GMT+0900', 'GMT+09:00', 'GMT+9:00'과 같은 형식으로 입력할 수 있습니다.

**LOCALE**

문자로 된 시간대 약어도 지원하지만, 중의적인 의미를 가질 수 있으므로 주의해야 합니다. 예를 들어 'CST'는 중국 표준시일 수도 있고, 미국 중부 표준시나 쿠바 표준시일 수도 있습니다. 시간대를 지정하지 않으면 로그프레소가 설치된 시스템의 로케일에 따른 시간대를 사용합니다. 시간대 약어는 다음 주소에 있는 Time Zone Abbreviations – Worldwide List를 참조하십시오: https://www.timeanddate.com/time/zones/

**시간대 약어 예시**

|  |  |  |
| --- | --- | --- |
| 약어 | GMT 오프셋 | 설명 |
| UTC | GMT+0 | 협정 세계시 |
| KST | GMT+9 | 대한민국 표준시 |
| CEST | GMT+2 | 중앙유럽 서머타임 |
| MSK | GMT+3 | 모스크바 표준시 |
| PST | GMT-7 | 태평양 표준시 |
| EST | GMT-5 | 미국 동부 표준시 |

#### 사용 예

json "{}" | eval str=string(1) => "1"json "{}" | eval str=string(1.2) => "1.2"json "{}" | eval str=string(true) => "true"json "{}" | eval str=string(null) => nulljson "{}" | eval str=string(now(),"yyyyMMddHHmmss") => "20140807164417"json "{}"| eval str=string(date("20170329","yyyyMMdd"),"yyyy-MM-dd HH:mm:ssZ","GMT+08") => "2017-03-28 23:00:00+0800"

### tobase64()

바이너리 값을 BASE64 문자열로 반환합니다.

#### 문법

tobase64(BLOB\_EXPR)

**BLOB\_EXPR**

바이너리로 평가되는 표현식을 입력합니다. 바이너리가 아닌 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval str=tobase64(binary("hello, world!"))=> "aGVsbG8sIHdvcmxkIQ=="

### tohex()

바이너리 값을 16진수 문자열로 변환합니다. 매개변수 값이 바이너리 타입이 아닌 경우에는 null을 반환합니다.

#### 문법

tohex(BLOB\_EXPR)

**BLOB\_EXPR**

16진수 문자로 변환할 바이너리 값

#### 사용 예

json "{}" | eval hex=tohex(encode("abcde")) => "6162636465"json "{}" | eval hex=tohex(1234) => nulljson "{}" | eval hex=tohex(null) => null

## 타입 검사 함수

### isnum()

인자가 숫자 타입(int, short, long, float, double)인 경우 true, 그렇지 않으면 false를 반환합니다. 입력 값이 null이면 경우 false를 반환합니다.

#### 문법

isnum(EXPR)

필수 매개변수

**EXPR**

검사 대상 값을 반환하는 표현식

#### 사용 예

json "{}" | eval bool=isnum(1) => truejson "{}" | eval bool=isnum(1.2) => truejson "{}" | eval bool=isnum("string") => falsejson "{}" | eval bool=isnum(null) => false

### isnotnull()

인자 값이 null이 아닐 때 true, 인자 값이 null일 때 false를 반환합니다.

#### 문법

isnotnull(EXPR)

필수 매개변수

**EXPR**

검사 대상 값을 반환하는 표현식

#### 사용 예

json "{}" | eval bool=isnotnull(1) => truejson "{}" | eval bool=isnotnull(null) => false

### isnull()

인자 값이 null일 때 true, 인자 값이 null이 아닐 때 false를 반환합니다.

#### 문법

isnull(EXPR)

필수 매개변수

**EXPR**

검사 대상 값을 반환하는 표현식

#### 사용 예

json "{}" | eval bool=isnull(null) => truejson "{}" | eval bool=isnull(1) => false

### isstr()

표현식이 문자열인 경우 true, 그렇지 않으면 false를 반환합니다. 표현식이 null인 경우 false를 반환합니다.

#### 문법

isstr(EXPR)

필수 매개변수

**EXPR**

검사 대상 값을 반환하는 표현식

#### 사용 예

json "{}" | eval bool=isstr("string") => truejson "{}" | eval bool=isstr(0) => falsejson "{}" | eval bool=isstr(null) => false

### typeof()

주어진 표현식의 타입을 문자열 표현으로 반환합니다.

#### 문법

typeof(EXPR)

**EXPR**

타입을 확인할 값을 반환하는 표현식을 지정합니다.

#### 설명

데이터 타입에 따라 다음과 같은 문자열을 반환합니다.

* string: 문자열
* short1 16비트 정수
* int: 32비트 정수
* long: 64비트 정수
* float: 32비트 단정도 소수
* double: 64비트 단정도 소수
* bool: 불리언
* ipv4: IPv4 주소
* ipv6: IPv4 주소
* date: 날짜
* map: 맵
* null: null

#### 사용 예

json "{}" | eval type=typeof(null) => nulljson "{}" | eval type=typeof("sample") => "string"json "{}" | eval type=typeof(1) => "int"json "{}" | eval type=typeof(2147483648) => "long"json "{}" | eval type=typeof(1.2) => "double"json "{}" | eval type=typeof(ip("1.2.3.4")) => "ipv4"json "{}" | eval type=typeof(ip("::1")) => "ipv6"json "{}" | eval type=typeof(true) => "bool"

## 조건 함수

### case()

여러 개의 조건에 따라 분기하여 표현식을 평가합니다.

#### 문법

case(CONDITION, EXPR[, ...], DEFAULT\_EXPR)

필수 매개변수

**CONDITION, EXPR[, ...]**

조건문(CONDITION)이 true이거나, null이 아닐 때 실행할 평가식(EXPR\_N)과 함께 쌍으로 입력합니다.

**DEFAULT\_EXPR**

어느 평가 조건에도 맞지 않을 때 수행할 표현식을 지정합니다.

#### 사용 예

Score가 90이 넘으면 A, 80이 넘으면 B, 70이 넘으면 C, 60이 넘으면 D, 그 외에는 F로 평가

json "[ {'Name': 'Alice', 'Score': 98}, {'Name': 'Bob', 'Score': 65}, {'Name': 'Clark', 'Score': 40} ]" | eval Grade=case( Score > 90, "A", Score > 80, "B", Score > 70, "C", Score > 60, "D", "F") | order Name, Grade, Score

str 필드 값의 문자열 길이가 9자보다 크면 9자로 자르고 말줄임표 적용

json "[{'str': 'Somewhere over the rainbow'}, {'str': 'Wonderful'}]" | eval truncated=case(len(str) > 10, concat(left(str, 10), "…"), str)

### if()

조건문이 평가 결과(참/거짓)에 따라 실행할 표현식을 평가합니다.

#### 문법

if(CONDITION, EXPR\_IF\_TRUE, EXPR\_IF\_FALSE)

필수 매개변수

**CONDITION**

조건문이 true 이거나, null이 아닐 때 참으로 평가됩니다.

**EXPR\_IF\_TRUE**

조건문이 참일 때 평가할 표현식을 입력합니다.

**EXPR\_IF\_FALSE**

조건문이 거짓일 때 평가할 표현식을 입력합니다.

#### 사용 예

status 코드가 200인 경우 ok, 아닌 경우 error 평가

if(status == "200", "ok", "error")

### in()

표현식의 평가값이 이후의 표현식 평가 값 집합 중에 존재하는지 여부를 평가합니다.

#### 문법

in(VAL\_EXPR, EXPR,...)

필수 매개변수

**VALUE\_EXPR**

평가 대상 표현식을 입력합니다.

선택 매개변수

**EXPR,...**

VALUE\_EXPR과 대조할 표현식을 쉼표(,)로 구분하여 입력합니다. 이 중 하나와 일치하면 true, 그렇지 않으면 false를 반환합니다.

#### 사용 예

user\_agent 필드 값이 msie, chrome, safari, firefox 중 하나와 일치하는지 검사

in(user\_agent, "msie", "chrome", "safari", "firefox")

user\_agent 필드 값이 google 혹은 yahoo 문자열을 포함하고 있는지 검사

in(user\_agent, "\*google\*", "\*yahoo\*")

level이 6, 7 중 하나인지 검사

in(level, 6, 7)

### match()

문자열의 일부가 정규표현식과 일치하는지 여부를 반환합니다.

#### 문법

match(VALUE\_EXPR, REGEX)

필수 매개변수

**VALUE\_EXPR**

평가 대상 표현식을 입력합니다. 값이 문자열이 아닌 경우, 문자열로 변환한 다음 REGEX와 비교합니다.

**REGEX**

VAL\_EXPR 값을 비교할 정규표현식을 큰 따옴표 쌍(" ")으로 감싸서 입력합니다. 표현식이 null인 경우 false를 반환합니다.

#### 사용 예

json "{}" | eval match=match("8 miles", "\\d+ [a-z]+") => truejson "{}"| eval match=match(" 8 miles ", "^\\d+ [a-z]+$") => falsejson "{}"| eval match=match("sample", "\\d+ [a-z]+") => falsejson "{}"| eval match=match(123, "\\d+") => truejson "{}"| eval match=match(null, "\\d+") => false

### nvl()

평가 대상 표현식이 null이 아니면 표현식의 값을 반환하고, null이면 기본값을 반환합니다.

#### 문법

nvl(VAL\_EXPR, DEFAULT\_EXPR)

**VALUE\_EXPR**

평가 대상 표현식을 입력합니다. 평가한 값이 null이 아닌 경우, 평가한 값을 반환합니다.

**DEFAULT\_EXPR**

VAL\_EXPR 값이 null일 때 반환할 값을 지정합니다.

#### 사용 예

json "{}" | eval nvl=nvl("hello", "") => "hello"json "{}" | eval nvl=nvl(null, "") => ""

## 문자열 함수

### concat

여러 개의 문자열을 결합하여 하나의 문자열로 만듭니다. 문자열이 아닌 표현식의 경우 문자열로 변환 후 결합됩니다.

#### 문법

concat(EXPR, ...)

필수 매개변수

**EXPR, ...**

문자열을 반환하는 표현식 목록. 구분자로 쉼표(,)를 사용합니다. 표현식이 반환하는 문자열을 모두 하나의 문자열로 결합합니다.

#### 사용 예

json "{}" | eval str=concat("hello", ", ", "world") => "hello, world"

### contains()

대상 문자열이 특정 부분문자열을 포함하는지 여부를 반환합니다.

#### 문법

contains(STR\_EXPR, SEARCH\_STR)

필수 매개변수

**STR\_EXPR**

문자열 표현식을 지정합니다.

**SEARCH\_STR**

STR\_EXPR의 문자열 값에 포함되어 있는지 확인할 문자열을 지정합니다.

#### 사용 예

json "{}" | eval isInclude=contains("foo", "o") => truejson "{}" | eval isInclude=contains("bar", "o") => falsejson "{}" | eval isInclude=contains("baz", null) => falsejson "{}" | eval isInclude=contains(null, null) => false

### format()

주어진 인자들을 이용해 만들어진 새 문자열을 반환합니다.

#### 문법

format(STR\_FMT, {PARAM\_1[, PARAM\_2, ...]|ARRAY\_EXPR})

필수 매개변수

**STR\_FMT**

형식 지시자를 포함한 형식 문자열. 사용 가능한 형식 지시자는 다음 주소에 있는 Class Formatter 문서를 참고하십시오: <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Formatter.html>

**{PARAM\_1[, PARAM\_2, ...]|ARRAY\_EXPR}**

주어진 형식으로 문자열을 표현하는데 이용될 입력값. 배열을 반환하는 표현식(예: [array()](https://docs.logpresso.com/ko/query/array-function), [groups()](https://docs.logpresso.com/ko/query/groups-function)를 이용하는 표현식)을 이용해 형식 문자열에 적용할 인자를 반환하도록 지정할 수 있습니다.

#### 사용 예

json "{}" | eval str=format("date: %04d-%02d-%02d", 2004, 3, 29) => "date: 2004-03-29"json "{}" | eval str=format("%3$s-%1$s-%2$s", groups("Mar 29 2004", "(.\*?) (.\*?) (.\*)")) => "2004-Mar-29"

### guid()

항상 유일한 GUID 문자열 값을 생성하여 반환합니다.

#### 문법

guid()

#### 사용 예

guid() => "a1189dda-870e-4aea-8742-76dcb8398b49"

### indexof()

문자열에서 특정한 문자열의 시작 위치를 검색합니다. 검색할 문자열을 찾을 수 없으면 -1을, 검색 대상 문자열이나 문자열에서 검색할 문자열이 null이면 null을 반환합니다.

#### 문법

indexof(STR\_EXPR, SEARCH\_EXPR[, BEGIN\_EXPR])

필수 매개변수

**STR\_EXPR**

문자열 표현식

**SEARCH\_EXPR**

STR\_EXPR의 문자열 값에 포함되어 있는지 확인할 문자열

선택 매개변수

**BEGIN\_EXPR**

검색을 시작할 위치 인덱스. 위치를 나타내는 인덱스는 '0'부터 시작합니다. 지정된 위치부터 SEARCH\_EXPR 값을 찾습니다.

#### 사용 예

indexof("hello world", "world") => 6indexof("hello world", "foo") => -1indexof("hello world", null) => nullindexof(null, "world") => nullindexof(null, null) => nullindexof("hello world", "o", 5) => 7

### kvjoin()

모든 키와 값을 결합해 하나의 문자열을 만듭니다.

#### 문법

kvjoin(KV\_DELIMIT, PAIR\_DELIMIT[, REGEX])

필수 매개변수

**KV\_DELIMIT**

키와 값을 구분하는 문자를 지정합니다.

**PAIR\_DELIMIT**

각 키-값 쌍들을 구분하는 문자를 지정합니다.

선택 매개변수

**REGEX**

키가 정규표현식과 일치하는 경우만 키와 값을 결합합니다. 지정하지 않으면 모든 키-값 쌍을 결합합니다.

#### 사용 예

키-값 구분자는 콜론(:)으로, 키-값 쌍 구분자는 ^을 사용해 문자열 결합

json "{}" | eval name="Kim", age=30 | eval result=kvjoin(":", "^") => "name:Kim^age:30"

src.\* 정규표현식과 일치하는 필드만 추출하여 키-값 구분자는 콜론(:), 쌍 구분자는 ^을 사용해 문자열 결합

json "{'src\_ip':'1.2.3.4', 'src\_port':45667, 'dst\_ip':'5.6.7.8', 'dst\_port':80, 'protocol':'TCP'}" | eval result=kvjoin(":", "^", "src.\*") => "src\_ip:1.2.3.4^src\_port:45667"

### lastindexof()

문자열에서 특정한 문자열이 나타나는 마지막 위치의 인덱스 값을 반환합니다. 검색할 문자열을 찾을 수 없으면 -1을, 검색 대상 문자열이나 문자열에서 검색할 문자열이 null이면 null을 반환합니다.

#### 문법

lastindexof(STR\_EXPR, SEARCH\_EXPR, [FROM\_EXPR])

필수 매개변수

**STR\_EXPR**

원본 문자열 표현식

**SEARCH\_EXPR**

검색할 문자열

선택 매개변수

**FROM\_EXPR**

STR\_EXPR에서 SEARCH\_EXPR 검색의 시작점으로 사용할 인덱스. FROM\_EXPR에서 역방향으로(인덱스가 0인 방향) SEARCH\_EXPR과 일치하는 문자열을 검색합니다.

SEARCH\_EXPR 문자열이 반드시 FROM\_EXPR부터 인덱스 0 사이에 모두 포함되어 있어야 하는 것은 아닙니다. SEARCH\_EXPR 문자열의 첫 인덱스가 인덱스 0과 FROM\_EXPR 사이에 포함되어 있으면 검색됩니다. 사용 예 5에서 FROM\_EXPR가 30("L"에 해당)이고, SEARCH\_EXPR 문자열 "Lo"의 인덱스 범위는 30 ~ 31이지만 "Lo"가 검색되어 인덱스 30을 반환하고 있습니다.

#### 사용 예

단일 문자의 마지막 인덱스 찾기

json "{}" | eval STR\_EXPR="Life is short. Use Logpresso." | eval LAST\_INDEX=lastindexof(STR\_EXPR, " ") | # 결괏값: LAST\_INDEX = 18

문자열의 마지막 위치 인덱스 찾기

json "{}" | eval STR\_EXPR="Life is short. Use Logpresso." | eval LAST\_INDEX=lastindexof(STR\_EXPR, "Logpresso") | # 결괏값: LAST\_INDEX = 19

존재하지 않는 문자열의 검색을 시도한 예

json "{}" | eval STR\_EXPR="Life is short. Use Logpresso." | eval LAST\_INDEX=lastindexof(STR\_EXPR, "Python") | # 결괏값 LAST\_INDEX = -1

검색을 시작할 인덱스를 찾은 후 마지막 인덱스 찾기(FROM\_INDEX 값을 수정하면서 실행해보세요)

# 문자열: Life is short. Use Logpresso. Long live Logpresso! 인덱스: 01234567891123456789212345678931234567894123456789 검색범위: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ (0-30) | json "{}" | eval STR\_EXPR="Life is short. Use Logpresso. Long Live Logpresso!" | eval SEARCH\_EXPR="Lo" | eval FROM\_INDEX=30 | eval LAST\_INDEX=lastindexof(STR\_EXPR, SEARCH\_EXPR, FROM\_INDEX) | # 결괏값 LAST\_INDEX = 30

[indexof()](https://docs.logpresso.com/ko/query/indexof-function) 함수와 비교

json "{}" | eval STR\_EXPR = "Life is short. Use Logpresso." | eval FIRST\_INDEX = indexof(STR\_EXPR, " ") | eval LAST\_INDEX = lastindexof(STR\_EXPR, " ") | # 결괏값: FIRST\_INDEX = 4, LAST\_INDEX = 18

#### 호환성

lastindexof() 함수는 버전 4.0.2312.0부터 사용 가능합니다.

### left()

문자열에서 지정한 길이만큼 왼쪽부터 문자열 잘라서 반환합니다. 지정된 길이보다 문자열의 길이가 짧으면 문자열 전체를 반환합니다. 인자가 null이면 null을 반환하고, 문자열이 아닌 값을 받으면 문자열로 변환한 다음 평가합니다.

#### 문법

left(STR\_EXPR, CHAR\_LENGTH)

필수 매개변수

**STR\_EXPR**

문자열 표현식

**CHAR\_LENGTH**

입력한 개수만큼 문자열의 왼쪽부터 문자를 잘라서 반환합니다. '0' 이상의 상수만 입력할 수 있습니다.

#### 사용 예

left("0123456789", 4) => "0123"left("0123456789", 11) => "0123456789"left("0123456789", 0) => ""left(1234, 2) => "12"left(1.23, 3) => "1.2"left(null, 3) => null

### len()

문자열의 길이를 반환합니다. 인자가 null이면 0을 반환하고, 문자열이 아닌 값을 받으면 문자열로 변환한 다음 평가합니다.

#### 문법

len(STR\_EXPR)

필수 매개변수

**STR\_EXPR**

문자열 표현식

#### 사용 예

json "{}" | eval length=len("sample") => 6json "{}" | eval length=len(null) => 0json "{}" | eval length=len(123) => 3json "{}" | eval length=len(1.2) => 3

### lower()

문자열을 소문자로 변환합니다. 인자가 null이면 null을 반환합니다. 문자열이 아닌 값을 받으면 문자열로 변환한 다음 평가합니다.

#### 문법

lower(STR\_EXPR)

필수 매개변수

**STR\_EXPR**

문자열 표현식

#### 사용 예

json "{}" | eval str=lower("Hello World") => "hello world"json "{}" | eval str=lower(1234) => "1234"json "{}" | eval str=lower(null) => null

### lpad()

문자열의 왼쪽에 패딩문자를 삽입하여 주어진 길이의 문자열을 만듭니다. 인자가 null이면 0을 반환하고, 문자열이 아닌 값을 받으면 문자열로 변환한 다음 평가합니다.

#### 문법

lpad(STR\_EXPR, OUTPUT\_LENGTH, [PADDING\_EXPR])

필수 매개변수

**STR\_EXPR**

문자열 표현식

**OUTPUT\_LENGTH**

반환할 문자열의 길이 표현식. STR\_EXPR 값이 OUTPUT\_LENGTH보다 길면 OUTPUT\_LENGTH에 맞춰 문자열을 잘라서 반환합니다.

선택 매개변수

**PADDING\_EXPR**

문자열의 왼쪽에 채울 문자열 표현식. 지정하지 않으면 공백문자로 채웁니다.

#### 사용 예

json "{}" | eval lpadded=lpad("string", 10) => " string"json "{}" | eval lpadded=lpad("string", 10, "p") => "ppppstring"json "{}" | eval lpadded=lpad("string", 10, "pad") => "padpstring"json "{}" | eval lpadded=lpad("string", 3, "pad") => "str"json "{}" | eval lpadded=lpad("string", null, "pad") => nulljson "{}" | eval lpadded=lpad("string", 3, null) => null

### replace()

문자열에서 주어진 패턴을 모두 찾아 지정한 문자열로 변경해 반환합니다.

#### 문법

replace(STR\_EXPR, PATTERN, REPLACE\_WITH\_THIS[, REGEX\_FLAG])

**STR\_EXPR**

원본 문자열 표현식

**PATTERN**

검색할 문자열. REGEX\_FLAG에 "re"를 지정하면 정규표현식 패턴을 입력합니다.

**REPLACE\_WITH\_THIS**

매칭된 문자열과 바꿀 문자열

**REGEX\_FLAG**

정규표현식 패턴 플래그로 "re"를 입력하면 정규표현식을 이용해 패턴 검색

#### 사용 예

json "{}" | eval new=replace("hello world", "world" , "logpresso") => "hello logpresso"json "{}" | eval new=replace("123412345", "12" , "!") => "!34!345"json "{}" | eval new=replace("google", "^g" , "b", "re") => "boogle"json "{}" | eval new=replace( "A:2 B:3 C:5 hahaha A:12 B:13 C:15", "A:(\\d+) B:\\d+ C:(\\d+)", "$1 $2 \\$1", "re" ) => "2 5 $1 hahaha 12 15 $1"

### reverseip()

IPv4 주소 문자열을 구성하는 각 숫자를 역순으로 조합한 문자열을 반환합니다. 예를 들어, 127.0.0.1을 매개변수로 입력하면 1.0.0.127을 반환합니다. 유효하지 않은 IPv4 주소 문자열에 대해서는 null을 반환합니다.

#### 문법

reverseip(EXPR)

필수 매개변수

**EXPR**

IPv4 주소 형식의 문자열 혹은 IP 주소 타입의 값

#### 사용 예

아래는 IP 주소를 뒤집고 .in-addr.arpa 문자열을 결합하여 리버스 도메인 조회하는 예시입니다.

json "{}" | eval ip = "172.217.14.238" | eval domain = concat(reverseip(ip), ".in-addr.arpa") | nslookup ns="1.1.1.1" type=PTR domain output status, answers

출력 필드는 다음과 같습니다:

* ip: 172.217.14.238
* domain: 238.14.217.172.in-addr.arpa
* status: NO\_ERROR
* answers: "PTR sea30s02-in-f14.1e100.net"

#### 더 알아보기

* [concat()](https://docs.logpresso.com/ko/query/concat-function)
* [nslookup](https://docs.logpresso.com/ko/query/nslookup-command)

### right()

문자열에서 지정한 길이만큼 오른쪽부터 문자열을 잘라서 반환합니다. 지정된 길이보다 문자열의 길이가 짧으며 문자열 전체를 반환합니다. 인자가 null이면 null을 반환하고, 문자열이 아닌 값을 받으면 문자열로 변환한 다음 평가합니다.

#### 문법

right(EXPR, LENGTH)

**STR\_EXPR**

원본 문자열

**LENGTH**

입력한 개수만큼 문자열의 오른쪽부터 문자를 잘라서 반환합니다. 지정된 길이보다 문자열의 길이가 짧은 경우 문자열 전체를 반환합니다. '0' 이상의 상수만 입력할 수 있습니다.

#### 사용 예

json "{}" | eval right=right("0123456789", 4) => "6789"json "{}" | eval right=right("0123456789", 11) => "0123456789"json "{}" | eval right=right("0123456789", 0) => ""json "{}" | eval right=right(1234, 2) => "34"json "{}" | eval right=right(1.23, 3) => ".23"json "{}" | eval right=right(null, 3) => null

### rpad()

문자열의 오른쪽에 패딩 문자를 삽입하여 주어진 길이의 문자열을 만듭니다. 인자가 null이면 '0'을 반환하고, 문자열이 아닌 값을 받으면 문자열로 변환한 다음 평가합니다.

#### 문법

rpad(STR\_EXPR, OUTPUT\_LENGTH, [PADDING\_EXPR])

**STR\_EXPR**

원본 문자열 표현식

**OUTPUT\_LENGTH**

반환할 문자열의 길이 표현식. STR\_EXPR 값이 OUTPUT\_LENGTH보다 길면 OUTPUT\_LENGTH에 맞춰 문자열을 잘라서 반환합니다.

**PADDING\_EXPR**

문자열의 오른쪽에 채울 문자열 표현식. 지정하지 않으면 공백문자로 채웁니다.

#### 사용 예

json "{}" | eval rpadded=rpad("string", 10) => "string "json "{}" | eval rpadded=rpad("string", 10, "p") => "stringpppp"json "{}" | eval rpadded=rpad("string", 10, "pad") => "stringpadp"json "{}" | eval rpadded=rpad("string", 3, "pad") => "str"json "{}" | eval rpadded=rpad("string", null, "pad") => nulljson "{}" | eval rpadded=rpad("string", 3, null) => null

### split()

원본 문자열에서 특정 문자열을 제거하고 나머지 문자열을 배열로 반환합니다.

#### 문법

split(STR\_EXPR, DELIMITER\_EXPR, [LIMIT\_EXPR])

**STR\_EXPR**

원본 문자열 표현식

**DELIMITER\_EXPR**

STR\_EXPR에서 배열 요소의 구분자로 사용할 문자열 표현식. 검색된 문자열을 구분자로 하여 문자열들을 분리합니다.

**LIMIT\_EXPR**

배열 요소의 최대 개수 표현식. 인자는 2 이상의 정수이어야 합니다.

* LIMIT\_EXPR을 지정하지 않으면 DELIMITER\_EXPR로 구분되는 모든 문자열을 배열 요소로서 분리합니다.
* LIMIT\_EXPR을 지정하면(숫자 n) STR\_EXPR 문자열의 처음부터 LIMIT\_EXPR로 지정한 숫자보다 1 작은 개수(n-1 개)만큼 DELIMITER\_EXPR로 식별되는 구분자를 찾아서 문자열을 분리합니다.

#### 사용 예

json "{'url': 'ko.logpresso.com/documents'}" | eval array=split(field("url"), "/")| # 반환 값: ["ko.logpresso.com", "documents"]

json "{}" | eval array=split("logpresso", "a")| # 반환 값: ["logpresso"]

json "{}" | eval array=split("a,b,c,d", ",")| # 반환 값: ["a","b","c","d"]

json "{'url': 'ko.logpresso.com/documents/test1/test2/test3'}" | eval array=split(field("url"), "/", 2)| # 반환 값: ["ko.logpresso.com", "documents/test1/test2/test3"]

### strjoin()

주어진 배열을 지정된 구분자로 구분된 하나의 문자열로 병합합니다.

#### 문법

strjoin(DELIMIT\_CHAR, ARRAY)

**DELIMIT\_CHAR**

배열 요소를 병합할 때 각 요소의 구분자로 사용할 문자열 상수. 구분자에 상수가 아닌 표현식이 들어오면 문법 오류가 발생합니다.

**ARRAY**

요소를 병합할 배열 표현식. 배열이 null이면 null을 반환하고, 배열의 요소가 null일 때에는 병합된 문자열에 null이 표시됩니다.

#### 사용 예

json "{}" | eval merged=strjoin(",", null) => nulljson "{}" | eval merged=strjoin(",", array(1, 2, 3)) => "1,2,3"

### substr()

원본 문자열에서 지정된 위치의 문자열만 반환합니다. 문자열이 아닌 값을 받으면 문자열로 변환한 다음 평가합니다.

#### 문법

substr(STR\_EXPR, B\_INDEX[, E\_INDEX])

**STR\_EXPR**

원본 문자열 표현식. 표현식이 null이면 null을 반환합니다.

**B\_INDEX**

반환할 문자열의 시작 문자 인덱스로, 인덱스는 '0'부터 시작합니다. 음수일 경우 문자열의 끝에서부터 계산합니다. 시작 위치가 문자열 길이보다 크면 null을 반환합니다.

**E\_INDEX**

반환할 문자열의 마지막 문자 인덱스. 인덱스는 '0'부터 시작합니다. 생략하면 문자열 끝을 의미합니다. 음수일 경우 문자열끝에서부터 위치를 계산합니다. 끝 위치가 문자열 길이보다 크면 시작 위치부터 문자열 끝까지 반환합니다.

#### 사용 예

json "{}" | eval partion\_str=substr("0123456789", 2) => "23456789"json "{}" | eval partion\_str=substr("0123456789", -2) => "89"json "{}" | eval partion\_str=substr("0123456789", 0, 3) => "012"json "{}" | eval partion\_str=substr("0123456789", 4, 12) => "456789"json "{}" | eval partion\_str=substr("0123456789", 5, 5) => ""json "{}" | eval partion\_str=substr("0123456789", 10, 11) => nulljson "{}" | eval partion\_str=substr("0123456789", -1, 11) => "9"json "{}" | eval partion\_str=substr(null, 0, 3)=> null

### trim()

문자열의 좌우에서 공백 문자(탭, 개행 포함)를 제거합니다. 문자열이 아닌 값을 받으면 문자열로 변환한 다음 평가합니다.

#### 문법

trim(STR\_EXPR)

**STR\_EXPR**

원본 문자열 표현식. 표현식이 null이면 null을 반환합니다.

#### 사용 예

json "{}" | eval trimed=trim(" hello world ") => "hello world"json "{}" | eval trimed=trim(123) => "123"json "{}" | eval trimed=trim(null) => null

### upper()

문자열을 대문자로 변환합니다. 문자열이 아닌 값을 받으면 문자열로 변환한 다음 평가합니다.

#### 문법

upper(EXPR)

필수 매개변수

**STR\_EXPR**

원본 문자열 표현식. 표현식이 null이면 null을 반환합니다.

#### 사용 예

json "{}" | eval UPPER=upper("Hello World") => "HELLO WORLD"json "{}" | eval UPPER=upper(1234) => "1234"json "{}" | eval UPPER=upper(null) => null

### urldecode()

주어진 URL을 디코드하여 반환합니다. 예를 들어 %20은 공백으로 변환됩니다.

#### 문법

urldecode(STR\_EXPR[, CHARSET])

**STR\_EXPR**

원본 문자열 표현식. 표현식이 null이면 null을 반환합니다.

**CHARSET**

파일의 인코딩 형식(기본값: utf-8). 인코딩 형식은 IANA Charset Registry에 등록된 Preferred MIME Name 또는 Aliases에 등록된 이름을 사용합니다: https://www.iana.org/assignments/character-sets/character-sets.xhtml

#### 사용 예

json "{ 'url': 'ko.logpresso.com/documents/%B7%CE%B1%D7%BA%D0%BC%AE'}" | eval decode=urldecode(field("url"), "EUC-KR") => ko.logpresso.com/documents/로그분석json "{ 'url': 'ko.logpresso.com/documents/%EB%A1%9C%EA%B7%B8%EB%B6%84%EC%84%9D'}" | eval decode=urldecode(field("url")) => ko.logpresso.com/documents/로그분석json "{}" | eval \_line=urldecode(null) => null

### urlencode()

주어진 문자열을 퍼센트(%) 인코딩하여 반환합니다.

#### 문법

urlencode(STR\_EXPR[, CHARSET])

**STR\_EXPR**

원본 문자열 표현식. 표현식이 null이면 null을 반환합니다.

**CHARSET**

문자 집합 (기본값: utf-8). 인코딩 형식은 IANA Charset Registry에 등록된 Preferred MIME Name 또는 Aliases에 등록된 이름을 사용합니다: https://www.iana.org/assignments/character-sets/character-sets.xhtml

#### 사용 예

json "{'uri': '퍼센트\_인코딩'}" | eval decode=concat( "https://ko.wikipedia.org/wiki/", urlencode(field("uri"), "UTF-8") ) => https://ko.wikipedia.org/wiki/%ED%8D%BC%EC%84%BC%ED%8A%B8\_%EC%9D%B8%EC%BD%94%EB%94%A9json "{}" | eval \_line=urlencode(null) => null

## 수치 함수

### abs()

임의의 숫자의 절대값을 계산합니다.

#### 문법

abs(NUM\_EXPR)

필수 매개변수

**NUM\_EXPR**

int, short, long, float, double 타입을 반환하는 표현식. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval abs=abs(-1) => 1json "{}" | eval abs=abs(1) => 1json "{}" | eval abs=abs(-1.234) => 1.234json "{}" | eval abs=abs(1 – 43) => 42

### acos()

코사인을 적용했을 때 지정된 숫자가 나오는 각도를 반환합니다.

#### 문법

acos(RADIAN\_EXPR)

필수 매개변수

**RADIAN\_EXPR**

라디안(radian) 표현식. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval acos=acos(0.866158094405463) => 0.5233333333333333 (30 \* 3.14 / 180)json "{}" | eval acos=acos(1) => 0json "{}" | eval acos=acos("0") => null

### asin()

사인을 적용했을 때 지정된 숫자가 나오는 각도를 반환합니다.

#### 문법

asin(RADIAN\_EXPR)

필수 매개변수

**RADIAN\_EXPR**

라디안(radian) 표현식. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval asin=asin(0.4997701026431024)=> 0.5233333333333333 (30 \* 3.14 / 180)json "{}" | eval asin=asin(0.8657598394923444)=> 1.0466666666666666 (60 \* 3.14 / 180)json "{}" | eval asin=asin(0.9999996829318346)=> 1.5699999999999876 (90 \* 3.14 / 180)json "{}" | eval asin=asin(0) => 0json "{}" | eval asin=asin("0") => null

### atan()

탄젠트를 적용했을 때 지정된 숫자가 나오는 각도를 반환합니다.

#### 문법

atan(RADIAN\_EXPR)

필수 매개변수

**RADIAN\_EXPR**

라디안(radian) 표현식. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval atan=atan(0.5769964003928729)=> 0.5233333333333333 (30 \* 3.14 / 180)json "{}" | eval atan=atan(1.7299292200897902)=> 1.0466666666666666 (60 \* 3.14 / 180)json "{}" | eval atan=atan(0) => 0json "{}" | eval atan=atan("0") => null

### ceil()

자리올림 함수로, 실수의 소수점 아래 자리를 올림하여 정수로 만듭니다. 소수점 아래 자릿수를 지정하면 자릿수에서 자리를 올립니다. 정수가 인자 값이면 입력을 그대로 반환합니다. 처리할 수 없는 인자값이나 처리할 수 없는 자릿수는 null을 반환합니다.

#### 문법

ceil(NUM\_EXPR[, NUM\_DIGITS])

필수 매개변수

**NUM\_EXPR**

int, short, long, float, double 타입을 반환하는 표현식.

선택 매개변수

**NUM\_DIGITS**

소숫점 아래 자릿수입니다. 음수일 경우 소숫점 위 자릿수로 처리해 자리올림합니다.

#### 사용 예

json "{}" | eval ceiling=ceil(1.1) => 2json "{}" | eval ceiling=ceil(1.6) => 2json "{}" | eval ceiling=ceil(1.61, 1) => 1.7json "{}" | eval ceiling=ceil(1.0) => 1json "{}" | eval ceiling=ceil(5) => 5json "{}" | eval ceiling=ceil(297.5, -2) => 300json "{}" | eval ceiling=ceil("asdf") => nulljson "{}" | eval ceiling=ceil("1.1") => nulljson "{}" | eval ceiling=ceil(1.1, "eediom") => null

### cos()

지정된 각도의 코사인을 반환합니다.

#### 문법

cos(RADIAN\_EXPR)

필수 매개변수

**RADIAN\_EXPR**

라디안(radian) 표현식. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval cos=cos(0) => 1json "{}" | eval cos=cos(30 \* 3.14 / 180) => 0.866158094405463json "{}" | eval cos=cos(60 \* 3.14 / 180) => 0.5004596890082058json "{}" | eval cos=cos(90 \* 3.14 / 180) => 0.0007963267107332633json "{}" | eval cos=cos("0") => null

### exp()

상수 𝑒를 지정된 수만큼 거듭제곱하여 반환합니다. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 문법

exp(NUM\_EXPR)

필수 매개변수

**NUM\_EXPR**

거듭제곱 지수를 반환하는 표현식

#### 사용 예

json "{}" | eval exp=exp(1) => 2.718281828459045json "{}" | eval exp=exp(2) => 7.38905609893065json "{}" | eval exp=exp(-1) => 0.36787944117144233json "{}" | eval exp=exp("2") => null

### floor()

자리내림 함수로, 실수의 소수점 아래 자리를 버림하여 정수로 만들고, 소수점 아래 자릿수를 받으면 그 아래 자리에서 버림합니다. 정수가 인자로 들어오면 입력을 그대로 반환합니다. 처리할 수 없는 인자값이나 처리할 수 없는 자릿수는 null을 반환합니다.

#### 문법

floor(NUM\_EXPR[, NUM\_DIGITS])

필수 매개변수

**NUM\_EXPR**

int, short, long, float, double 타입을 반환하는 표현식

선택 매개변수

**NUM\_DIGITS**

소숫점 아래 자릿수. 음수일 경우 소숫점 위 자릿수로 처리해 자리내림합니다.

#### 사용 예

json "{}" | eval floor=floor(1.1) => 1json "{}" | eval floor=floor(1.61, 1) => 1.6json "{}" | eval floor=floor(531, -1) => 530json "{}" | eval floor=floor(5) => 5json "{}" | eval floor=floor("1.1") => nulljson "{}" | eval floor=floor("asdf") => nulljson "{}" | eval floor=floor(4.3, "eediom") => null

### log()

지정된 숫자의 자연 로그를 반환합니다. 매개변수에 음수를 전달하는 경우, NaN(Not a Number)을 반환합니다. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 문법

log(NUM\_EXPR)

필수 매개변수

**NUM\_EXPR**

int, short, long, float, double 타입을 반환하는 표현식

#### 사용 예

json "{}" | eval log=log(10) => 2.302585092994046

### log10()

지정된 숫자의 상용 로그를 반환합니다. 매개변수에 음수를 전달하는 경우, NaN 값을 반환합니다. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 문법

log10(NUM\_EXPR)

필수 매개변수

**NUM\_EXPR**

int, short, long, float, double 타입을 반환하는 표현식

#### 사용 예

json "{}" | eval result=log10(10) => 1json "{}" | eval result=log10(100) => 2json "{}" | eval result=log10(-10) => NaN

### max()

주어진 표현식 중에 최대값을 반환합니다. null인 표현식은 무시합니다.

#### 문법

max(NUM\_EXPR\_1, NUM\_EXPR\_2, ...)

**NUMBER\_1, NUMBER\_2, ...**

int, short, long, float, double 타입을 반환하는 표현식 목록.

#### 사용 예

json "{}" | eval max=max(1) => 1json "{}" | eval max=max(1, 2) => 2json "{}" | eval max=max(1, 2, null) => 2json "{}" | eval max=max(null) => null

### min()

주어진 표현식 중에 최소값을 반환합니다. null인 표현식은 무시합니다.

#### 문법

min(NUM\_EXPR\_1, NUM\_EXPR\_2, ...)

**NUMBER\_1, NUMBER\_2, ...**

int, short, long, float, double 타입을 반환하는 표현식 목록.

#### 사용 예

json "{}" | eval min=min(1) => 1json "{}" | eval min=min(1, 2) => 1json "{}" | eval min=min(1, 2, null) => 1json "{}" | eval min=min(null) => null

### mod()

나눗셈의 나머지 값을 반환합니다.

#### 문법

mod(NUM\_EXPR, DIVISOR)

**NUM\_EXPR**

int, short, long, float, double 타입을 반환하는 표현식

**DIVISOR**

나누는 수 표현식

#### 사용 예

json "{}" | eval mod=mod(5, 2) => 1json "{}" | eval mod=mod(5, 0) => nulljson "{}" | eval mod=mod(null, 3) => nulljson "{}" | eval mod=mod("test", 3) => null

### pow()

밑수를 지정한 만큼 거듭제곱한 결과를 구합니다.

#### 문법

pow(NUM\_EXPR, POWER)

**NUM\_EXPR**

거듭제곱할 수를 반환하는 표현식.

**POWER**

거듭제곱 지수를 반환하는 표현식

#### 사용 예

json "{}" | eval pow=pow(2, 0) => 1json "{}" | eval pow=pow(2, 1) => 2json "{}" | eval pow=pow(2, 2) => 4

### round()

수를 지정한 자릿수로 반올림합니다. 처리할 수 없는 인자값이나 자릿수는 null을 반환합니다.

#### 문법

round(NUM\_EXPR[, NUM\_DIGITS])

**NUM\_EXPR**

int, short, long, float, double 타입을 반환하는 표현식.

**NUM\_DIGITS**

소숫점 아래 자릿수입니다. 음수일 경우 소숫점 위 자릿수로 처리해 반올림합니다.

#### 사용 예

json "{}" | eval round=round(1.0) => 1json "{}" | eval round=round(1.6) => 2json "{}" | eval round=round(1.47, 1) => 1.5json "{}" | eval round=round(1837, -2) => 1800json "{}" | eval round=round(5) => 5

### seq()

호출될 때마다 1부터 순차적으로 증가하는 번호를 반환합니다.

#### 문법

seq()

### sin()

지정된 각도의 사인을 반환합니다.

#### 문법

sin(RADIAN\_EXPR)

**RADIAN\_EXPR**

라디안(radian) 표현식. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval sin=sin(0) => 0json "{}" | eval sin=sin(30 \* 3.14 / 180) => 0.4997701026431024json "{}" | eval sin=sin(60 \* 3.14 / 180) => 0.8657598394923444json "{}" | eval sin=sin(90 \* 3.14 / 180) => 0.9999996829318346json "{}" | eval sin=sin("0") => null

### sqrt()

지정된 숫자의 제곱근을 반환합니다. 매개변수에 음수를 전달하는 경우, NaN을 반환합니다. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 문법

sqrt(NUM\_EXPR)

**NUM\_EXPR**

int, short, long, float, double 타입을 반환하는 표현식

#### 사용 예

sqrt(4) => 2sqrt(-1) => NaN

### tan()

지정된 각도의 탄젠트를 반환합니다.

#### 문법

tan(RADIAN\_EXPR)

**RADIAN\_EXPR**

라디안(radian) 표현식. 숫자가 아닌 인자 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval tan=tan(0) => 0json "{}" | eval tan=tan(30 \* 3.14 / 180) => 0.5769964003928729json "{}" | eval tan=tan(60 \* 3.14 / 180) => 1.7299292200897902json "{}" | eval tan=tan("0") => null

## 날짜 함수

### ago()

현재 시간 기준으로 입력된 시간 단위값 만큼의 이전 시간을 반환합니다.

#### 문법

ago("NUM{y|mon|w|d|h|m|s}")

필수 매개변수

**NUM{y|mon|w|d|h|m|s}**

s(초), m(분), h(시), d(일), w(주), mon(월), y(연) 단위로 지정할 수 있습니다.

#### 사용 예

2019-04-26 14:31:21 기준 예시입니다.

json "{}" | eval adjusted\_time=ago("3d") => 2019-04-23 14:31:21json "{}" | eval adjusted\_time=ago("5m") => 2019-04-26 14:26:21json "{}" | eval adjusted\_time=ago("13h") => 2019-04-26 01:31:21json "{}" | eval adjusted\_time=ago("1y") => 2018-04-26 14:31:21

### dateadd()

특정한 시각을 기준으로 주어진 시간과 단위만큼 시각을 계산합니다.

#### 문법

dateadd(DATE, "{year|mon|day|hour|min|sec|msec}", INT)

필수 매개변수

**DATE**

시간 타입을 반환하는 표현식

**"{year|mon|day|hour|min|sec|msec}"**

INT 값을 계산할 시간 단위. 각 시간 유형 지시자의 의미는 아래 표를 참조하십시오.

**시간 유형 지시자**

|  |  |
| --- | --- |
| 단위 | 설명 |
| year | 연 |
| mon | 월 |
| day | 일 |
| hour | 시 |
| min | 분 |
| sec | 초 |
| msec | 밀리초 |

**INT**

시간 단위에 더하거나 뺄 시간을 정수로 지정

#### 사용 예

json "{}" | eval time=dateadd( date("2013-09-28 11:47:00", "yyyy-MM-dd HH:mm:ss"), "year", 1) => Sun Sep 28 11:47:00 KST 2014

### datediff()

주어진 시작 시각과 마지막 시각 사이의 기간을 지정한 시간 단위로 계산합니다.

#### 문법

datediff(START\_DATE, END\_DATE, "{year|mon|day|hour|min|sec|msec}")

필수 매개변수

**START\_DATE**

시작 시각을 반환하는 표현식. 다른 타입 값을 받으면 null을 반환합니다.

**END\_DATE**

마지막 시각을 반환하는 표현식. 다른 타입 값을 받으면 null을 반환합니다.

**"{year|mon|day|hour|min|sec|msec}"**

START\_DATE와 END\_DATE의 기간을 계산할 시간 단위를 큰 따옴표로 감싸서 지정합니다. 각 시간 유형 지시자의 의미는 시간 유형 지시자 표를 참조하십시오.

**시간 유형 지시자**

|  |  |
| --- | --- |
| 단위 | 설명 |
| year | 연 |
| mon | 월 |
| day | 일 |
| hour | 시 |
| min | 분 |
| sec | 초 |
| msec | 밀리초 |

#### 사용 예

2014년 9월 29일에서 2013년 9월 29일의 차이를 계산

json "{}" | set start=date("2013-09-29", "yyyy-MM-dd") | set end=date("2014-09-29", "yyyy-MM-dd") | eval year = datediff($("start"), $("end"), "year"), mon = datediff($("start"), $("end"), "mon"), day = datediff($("start"), $("end"), "day"), hour = datediff($("start"), $("end"), "hour"), min = datediff($("start"), $("end"), "min"), sec = datediff($("start"), $("end"), "sec"), msec = datediff($("start"), $("end"), "msec")

잘못된 입력 예시

json "{}" | eval error0 = datediff(null, date("2014-09-29", "yyyy-MM-dd"), "sec"), error1 = datediff(date("2013-09-29", "yyyy-MM-dd"), null, "min"), error2 = datediff("invalid", date("2014-09-29", "yyyy-MM-dd"), "min")

### datepart()

특정 시각에서 지정한 시간 단위(세기, 연, 월, 일, 요일, 등)에 해당하는 정수값을 추출합니다.

#### 문법

datepart(DATE, DATEPART)

필수 매개변수

**DATE**

시간 타입을 반환하는 표현식. 다른 타입 값을 받으면 null을 반환합니다.

**DATEPART**

상수 문자열. 유형 지시자의 목록은 다음 표를 참조하십시오.

**유형 지시자**

|  |  |  |  |
| --- | --- | --- | --- |
| 유형 지시자 | 타입 | 설명 | 예시 값 |
| century | int | 세기 | 21 |
| day | int | 날짜 (1~31) | 12 |
| decade | int | 연도를 10으로 나눈 몫 | 201 |
| dow | int | 한 주 내에서의 날 수. 일요일 (0), 토요일 (6) | 1 |
| doy | int | 한 해 내에서의 날 수 | 163 |
| epoch | long | 1970년 1월 1일부터 날짜까지 경과한 초 | 1497269156 |
| hour | int | 24시간 기준 시간 (0-23) | 21 |
| isodow | int | ISO 8601 기준 한 주 내에서의 날 수. 월요일 (1), 일요일 (7) | 1 |
| isoyear | int | ISO 8601 기준 한 해의 첫 월요일을 새해 첫 날로 인식한 년도 | 2017 |
| microseconds | int | 초를 포함한 마이크로초 | 56371000 |
| millenium | int | 밀레니엄 (천 년 단위) | 3 |
| milliseconds | int | 초를 포함한 밀리초 | 56371 |
| min, minute | int | 분 (0-59) | 5 |
| mon, month | int | 월 (1-12) | 6 |
| msec | int | 초를 포함하지 않은 밀리초 | 377 |
| quarter | int | 분기 (1-4) | 2 |
| sec, seconds | int | 초 (0-59) | 56 |
| timezone | int | UTC 기준 시간대 (초) | 32400 |
| timezone\_hour | int | UTC 기준 시간대 (시) | 9 |
| timezone\_minute | int | UTC 기준 시간대 (분) | 0 |
| week | int | ISO 8601 기준 한 해의 첫 월요일을 새해 첫 날로 인식한 주의 수 | 24 |
| year | int | 년도 | 2017 |

#### 사용 예

"6월 1 2020 12:34:56"에서 연도에 해당하는 값을 추출

json "{}" | eval time= datepart( date("6월 1 2020 12:34:56", "MMM dd yyyy HH:mm:ss", "ko"), "year" )

"6월 1 2020 12:34:56"에서 월에 해당하는 값을 추출

json "{}" | eval time= datepart( date("6월 1 2020 12:34:56", "MMM dd yyyy HH:mm:ss", "ko"), "mon" )

"6월 1 2020 12:34:56"에서 유닉스 시간에 해당하는 값을 추출

json "{}" | eval time=datepart( date( "6월 1 2020 12:34:56", "MMM dd yyyy HH:mm:ss", "ko"), "epoch" ) => 1590982496

### daterange()

시작 날짜 표현식의 값으로부터 끝 날짜 표현식에 이르기 전까지 시간 단위만큼 건너뛰면서 날짜 값을 생성하여 리스트로 반환합니다. 반환하는 리스트에 끝 날짜는 포함하지 않습니다.

#### 문법

daterange(START\_DATE, END\_DATE[, INTERVAL{y|mon|w|d|h|m|s}])

필수 매개변수

**START\_DATE**

시작 날짜를 반환하는 표현식을 입력합니다. 다른 타입 값을 받으면 null을 반환합니다.

**END\_DATE**

끝 날짜를 반환하는 표현식을 입력합니다. 다른 타입 값을 받으면 null을 반환합니다.

선택 매개변수

**INTERVAL{y|mon|w|d|h|m|s}**

시간 단위를 지정합니다. s(초), m(분), h(시), d(일), w(주), mon(월), y(연) 단위로 지정할 수 있습니다. 단위를 지정하지 않으면 1d를 사용합니다.

시스템 과부하를 방지하기 위해 daterange() 함수의 결과가 10만건을 초과하면 예외를 발생시켜 쿼리를 실패시킵니다.

#### 사용 예

json "{}" | eval mark\_days= daterange( date("20150901", "yyyyMMdd"), date("20150908", "yyyyMMdd") ) => ["2015-09-01 00:00:00+0900","2015-09-02 00:00:00+0900","2015-09-03 00:00:00+0900","2015-09-04 00:00:00+0900","2015-09-05 00:00:00+0900","2015-09-06 00:00:00+0900","2015-09-07 00:00:00+0900"]json "{}" | eval mark\_days= daterange( date("20150901", "yyyyMMdd"), date("20150902", "yyyyMMdd"), "4h" ) => ["2015-09-01 00:00:00+0900","2015-09-01 04:00:00+0900","2015-09-01 08:00:00+0900","2015-09-01 12:00:00+0900","2015-09-01 16:00:00+0900","2015-09-01 20:00:00+0900"]json "{}" | eval mark\_days=daterange("20150901", "20150908") => null

### datetrunc()

지정된 시간날짜 표현식을 지정된 시간 단위를 기준으로 절사하여 반환합니다.

#### 문법

datetrunc(DATE, DATEPART{y|mon|w|d|h|m|s})

필수 매개변수

**DATE**

시간 타입을 반환하는 표현식을 입력합니다. 다른 타입 값을 받으면 null을 반환합니다.

**DATEPART**

시간 단위를 s(초), m(분), h(시), d(일), w(주), mon(월), y(연) 단위로 지정합니다.

#### 사용 예

json "{}" | eval date=datetrunc( date("2014-07-14 11:13:23", "yyyy-MM-dd HH:mm:ss"),"1m") => 2014-07-14 11:13:00+0900json "{}" | eval date=datetrunc( date("2014-07-14 11:13:23", "yyyy-MM-dd HH:mm:ss"),"5m") => 2014-07-14 11:10:00+0900json "{}" | eval date=datetrunc( date("2014-07-14 11:13:23", "yyyy-MM-dd HH:mm:ss"),"1mon") => 2014-07-01 00:00:00+0900

### epoch()

1970년 1월 1일 이후 경과된 초 또는 밀리초를 입력받아 date 타입으로 변환합니다. 매개변수 값을 초 단위로 해석했을 때 9999년 1월 1일보다 크다면 밀리초로 해석합니다.

#### 문법

epoch(NUM\_EXPR)

필수 매개변수

**NUM\_EXPR**

초 또는 밀리초 단위의 자연수를 반환하는 표현식을 입력합니다.

#### 사용 예

json "{}" | eval time=epoch(1435196373492) => 2015-06-25 10:39:33+0900

### now()

함수가 호출되는 시점의 현재 시스템 시각을 반환합니다.

#### 문법

now()

#### 사용 예

json "{}" | eval time=now() => Sat Sep 28 23:58:41 KST 2013

## IP 주소 함수

### ip2int()

임의의 IPv4 주소 문자열을 숫자(signed int) 타입으로 변환합니다.

#### 문법

ip2int(IP4\_ADDR)

필수 매개변수

**IP4\_ADDR**

IPv4 주소 문자열 표현식. 유효하지 않은 다른 타입 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval ip2int=ip2int("192.168.0.1") => -1062731775json "{}" | eval ip2int=ip2int("127.0.0.1") => 2130706433json "{}" | eval ip2int=ip2int("255.255.255.255") => -1json "{}" | eval ip2int=ip2int("256.256.256.256") => null

### ip2long()

#### 문법

ip2long(IP4\_ADDR)

필수 매개변수

**IP4\_ADDR**

IPv4 주소 문자열 표현식. 유효하지 않은 다른 타입 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval ip2long=ip2long("192.168.0.1") => 3232235521json "{}" | eval ip2long=ip2long("127.0.0.1") => 2130706433json "{}" | eval ip2long=ip2long("255.255.255.255") => 4294967295json "{}" | eval ip2long=ip2long("256.256.256.256") => null

### long2ip()

임의의 정수를 IPv4 주소 문자열로 변환합니다.

#### 문법

long2ip(LONG\_INT)

필수 매개변수

**LONG\_INT**

정수 표현식. 유효하지 않은 다른 타입 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval long2ip=long2ip(3232235521) => "192.168.0.1"json "{}" | eval long2ip=long2ip(2130706433) => "127.0.0.1"json "{}" | eval long2ip=long2ip(-1) => "255.255.255.255"json "{}" | eval long2ip=long2ip(-1062731775) => "192.168.0.1"

### network()

주어진 IPv4/IPv6 주소와 CIDR로 네트워크 주소 값을 반환합니다.

#### 문법

network(IP\_ADDR, CIDR)

**IP\_ADDR**

IPv4 또는 IPv6 형식의 문자열 혹은 IP 타입 값. IPv6 주소는 대소문자를 구분하지 않습니다.

**CIDR**

CIDR 정수 값. IPv4의 경우 0~32, IPv6의 경우 0~128 범위로 지정합니다.

#### 사용 예

json "{}" | eval network=network(null, 32) => nulljson "{}" | eval network=network("192.0.2.128", 24) => 192.0.2.0json "{}" | eval network=network("192.0.2.128", 28) => 192.0.2.128json "{}" | eval network=network(ip(-1073741184), 28) => 192.0.2.128json "{}" | eval network=network("21DA:00D3:0000:2F3B:02AA:00FF:FE28:9C5A", 96) => 21da:d3:0:2f3b:2aa:ff:0:0

## 암호화 함수

### decode()

바이너리 값을 지정된 인코딩으로 해석하여 문자열을 반환합니다.

#### 문법

decode(BLOB\_EXPR[, CHARSET])

필수 매개변수

**BLOB\_EXPR**

바이너리로 평가되는 표현식. 바이너리가 아닌 값을 받으면 null을 반환합니다.

선택 매개변수

**CHARSET**

파일의 인코딩(기본값: utf-8). 인코딩 형식 이름은 IANA Charset Registry에 등록된 Preferred MIME Name 또는 Aliases에 등록된 이름을 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

#### 사용 예

json "{}" | eval encoded=encode("hello, world!"), decoded=decode(encoded) => encoded: 68656c6c6f2c20776f726c6421 # 바이너리 => decoded: "hello, world!" # 문자열

### decrypt()

Java가 제공하는 Cipher 클래스를 이용해 암호화된 바이너리 값을 복호화하여 반환합니다.

#### 문법

decrypt(CIPHER, KEY, DATA[, IV])

필수 매개변수

**CIPHER**

알고리즘/모드/패딩 형식으로 구성된 문자열을 반환하는 표현식. 모드와 패딩을 생략하고 알고리즘만 입력하면 기본 암호 알고리즘을 적용합니다.

* AES만 입력하면 AES/CBC/NoPadding를 사용합니다.
* RSA만 입력하면 RSA/ECB/PKCS1Padding을 사용합니다.

사용할 수 있는 알고리즘, 모드, 패딩은 Java Security Standard Algorithm Names 문서를 참조하십시오: <https://docs.oracle.com/en/java/javase/11/docs/specs/security/standard-names.html>.

* 알고리즘은 **Cipher Algorithm Names** 섹션에서 확인할 수 있습니다.
* 모드는 **Cipher Algorithm Modes** 섹션에서 확인할 수 있습니다.
* 패딩은 **Cipher Algorithm Paddings** 섹션에서 확인할 수 있습니다.

로그프레소는 호환성을 위해 Java가 제공하는 다양한 암호 알고리즘을 지원합니다. 그러나 DES 계열이나 ECB 모드와 같이 안전하지 않은 암호 알고리즘이나 모드는 사용하지 않는 것이 좋습니다. Java에서 사용하는 암호화 클래스(Cipher Class)는 다음 주소에 있는 문서를 참조하십시오: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/Cipher.html

다음은 모든 Java 구현체가 반드시 지원해야 하는 형식입니다. DES, DESede 알고리즘, ECB 모드는 안전하지 않으므로 외부 시스템과 호환성 문제 등이 있을 때에만 사용하기 바랍니다. 괄호 안은 암호화 비트를 나타냅니다. - AES/CBC/NoPadding (128) - AES/CBC/PKCS5Padding (128) - AES/ECB/NoPadding (128) - AES/ECB/PKCS5Padding (128) - AES/GCM/NoPadding (128) - DES/CBC/NoPadding (56) - DES/CBC/PKCS5Padding (56) - DES/ECB/NoPadding (56) - DES/ECB/PKCS5Padding (56) - DESede/CBC/NoPadding (168) - DESede/CBC/PKCS5Padding (168) - DESede/ECB/NoPadding (168) - DESede/ECB/PKCS5Padding (168) - RSA/ECB/PKCS1Padding (1024, 2048) - RSA/ECB/OAEPWithSHA-1AndMGF1Padding (1024, 2048) - RSA/ECB/OAEPWithSHA-256AndMGF1Padding (1024, 2048)

**KEY**

지정한 암호 알고리즘에 일치하는 크기의 바이너리 키를 입력합니다. 알고리즘에 따른 키 길이는 다음과 같습니다.

* AES: 16자(128비트), 24자(192비트), 32자(256비트) 중 선택
* RSA: 128자(1024비트), 256자(2048비트) 중 선택

암호화 키 길이는 암호 알고리즘이 요구하는 키의 길이를 8로 나눠 바이트로 환산한 값입니다.

**DATA**

복호화할 바이너리 데이터

선택 매개변수

**IV**

CBC와 같이 초기화 벡터(IV, Initial Vector)가 필요한 운영 모드를 사용할 때 바이너리 값

#### 사용 예

다음 쿼리문을 [encrypt()](https://docs.logpresso.com/ko/query/encrypt-function) 사용 예와 비교해 보십시오.

json "{}" | eval decrypted= decode( decrypt("AES",frombase64("mRcOlK9V47rjVL/RBYQYRw=="), frombase64("y7+NQQ9/9xGtbBq5pgBvCA==") ))

### encode()

문자열을 지정된 인코딩을 이용하여 바이너리 개체로 반환합니다.

#### 문법

encode(STR, [CHARSET])

필수 매개변수

**STR**

문자열로 평가되는 표현식. 바이너리가 아닌 값을 받으면 null을 반환합니다.

선택 매개변수

**CHARSET**

문자열 인코딩 형식(기본값: utf-8). 다음 문서에 등록된 Preferred MIME Name이나 Aliases를 사용합니다: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

#### 사용 예

json "{}" | eval encoded=encode("hello, world!"), decoded=decode(encoded)=> encoded: 68656c6c6f2c20776f726c6421 # 바이너리=> decoded: "hello, world!" # 문자열

### encrypt()

바이너리 값을 지정된 알고리즘과 키로 암호화하여 반환합니다.

#### 문법

encrypt(CIPHER, KEY, DATA[, IV])

필수 매개변수

**CIPHER**

알고리즘/모드/패딩 형식으로 구성된 문자열을 반환하는 표현식. 모드와 패딩을 생략하고 알고리즘만 입력하면 기본 암호 알고리즘을 적용합니다.

* AES만 입력하면 AES/CBC/NoPadding를 사용합니다.
* RSA만 입력하면 RSA/ECB/PKCS1Padding을 사용합니다.

로그프레소는 호환성을 위해 Java가 제공하는 다양한 암호 알고리즘을 지원합니다. 그러나 DES 계열이나 ECB 모드와 같이 안전하지 않은 암호 알고리즘이나 모드는 사용하지 않는 것이 좋습니다. Java에서 사용하는 암호화 클래스(Cipher Class)는 다음 주소에 있는 문서를 참조하십시오: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/Cipher.html

사용할 수 있는 알고리즘, 모드, 패딩은 Java Security Standard Algorithm Names 문서를 참조하십시오: https://docs.oracle.com/en/java/javase/11/docs/specs/security/standard-names.html. \* 알고리즘은 Cipher Algorithm Names 섹션에서 확인할 수 있습니다. \* 모드는 Cipher Algorithm Modes 섹션에서 확인할 수 있습니다. \* 패딩은 Cipher Algorithm Paddings 섹션에서 확인할 수 있습니다.

다음은 모든 Java 구현체가 반드시 지원해야 하는 형식입니다. DES, DESede 알고리즘, ECB 모드는 안전하지 않으므로 외부 시스템과 호환성 문제 등이 있을 때에만 사용하기 바랍니다. 괄호 안은 암호화 비트를 나타냅니다. - AES/CBC/NoPadding (128) - AES/CBC/PKCS5Padding (128) - AES/ECB/NoPadding (128) - AES/ECB/PKCS5Padding (128) - AES/GCM/NoPadding (128) - DES/CBC/NoPadding (56) - DES/CBC/PKCS5Padding (56) - DES/ECB/NoPadding (56) - DES/ECB/PKCS5Padding (56) - DESede/CBC/NoPadding (168) - DESede/CBC/PKCS5Padding (168) - DESede/ECB/NoPadding (168) - DESede/ECB/PKCS5Padding (168) - RSA/ECB/PKCS1Padding (1024, 2048) - RSA/ECB/OAEPWithSHA-1AndMGF1Padding (1024, 2048) - RSA/ECB/OAEPWithSHA-256AndMGF1Padding (1024, 2048)

**KEY**

지정한 암호 알고리즘에 일치하는 크기의 바이너리 키를 입력합니다. 알고리즘에 따른 키 길이는 다음과 같습니다.

* AES: 16자(128비트), 24자(192비트), 32자(256비트) 중 선택
* RSA: 128자(1024비트), 256자(2048비트) 중 선택

암호화 키 길이는 암호 알고리즘이 요구하는 키의 길이를 8로 나눠 바이트로 환산한 값입니다.

**DATA**

암호화할 바이너리 데이터

선택 매개변수

**IV**

CBC와 같이 초기화 벡터(IV, Initial Vector)가 필요한 운영 모드를 사용할 때 바이너리 값

#### 사용 예

다음 쿼리문을 [decrypt()](https://docs.logpresso.com/ko/query/decrypt-function) 사용 예와 비교해 보십시오.

json "{}" | eval encrypted=tobase64( encrypt("AES", frombase64("mRcOlK9V47rjVL/RBYQYRw=="), binary("hello, world!") ))```

### hash()

단방향 해시 알고리즘을 수행한 결과를 바이너리 값으로 반환합니다.

#### 문법

hash(HASH\_ALGO, BIN\_DATA)

필수 매개변수

**HASH\_ALGO**

해시 알고리즘을 md5, sha1, sha256, sha384, sha512 중에서 선택해서 입력합니다.

**BIN\_DATA**

해시를 적용할 데이터를 지정합니다. 데이터는 바이너리 형태이어야 합니다. 바이너리가 아닌 값을 받으면 null을 반환합니다.

#### 사용 예

json "{}" | eval hash=hash("md5", binary("hello, world!")) => 3adbbad1791fbae3ec908894c4963870json "{}" | eval hash=hash("sha1", binary("hello, world!")) => 1f09d30c707d53f3d16c530dd73d70a6ce7596a9json "{}" | eval hash=hash("sha256", binary("hello, world!")) => 68e656b251e67e8358bef8483ab0d51c6619f3e7a1a9f0e75838d41ff368f728json "{}" | eval hash=hash("sha384", binary("hello, world!")) => fdbd8e75a67f29f701a4e040385e2e23986303ea10239211af907fcbb83578b3e417cb71ce646efd0819dd8c088de1bdjson "{}" | eval hash=hash("sha512", binary("hello, world!")) =>6c2618358da07c830b88c5af8c3535080e8e603c88b891028a259ccdb9ac802d0fc0170c99d58affcf00786ce188fc5d753e8c6628af2071c3270d50445c4b1cjson "{}" | eval hash=hash("md5", "hello world") => nulljson "{}" | eval hash=hash("sha1", null) => nulljson "{}" | eval hash=hash("sha1", 1234) => null

### rand()

0보다 크고 지정한 경계값보다 작은 임의의 정수를 반환합니다.

#### 문법

rand(NUM[, SEED])

**NUM**

0보다 큰 정수를 입력합니다. 0보다 크고 NUM보다 작은 임의의 숫자를 반환합니다.

**SEED**

이 값을 입력하면 무작위 정수 값을 반환하지 않고 항상 고정된 값을 반환합니다. 이 함수를 이용하는 쿼리문을 디버깅하거나 쿼리문의 동작을 검증하기 위한 시험 등의 목적으로 사용합니다.

SEED 매개변수는 쿼리문 기능 검증이나 시험과 같은 목적으로 쿼리문을 실행할 때 항상 값은 값을 반환하게 할 목적으로 쓰입니다. 실제 운영 환경에서는 사용에 주의하십시오.

#### 사용 예

json "{}" | eval rand=rand(1000) => 0~999의 임의의 값

### randbytes()

지정된 길이만큼 임의의 값으로 채워진 바이너리를 반환합니다.

#### 문법

randbytes(NUM)

**NUM**

바이너리 길이 상수를 입력합니다. 1에서 10,240까지의 길이만 허용됩니다.

#### 사용 예

json "{}" | eval rand\_blob=randbytes(32) => 바이너리 값 예시 6765eab83af980a266461427739cc37a8b4ee60dab09b091282c1070a3191e2d

## 배열 함수

### flatten()

재귀적으로 중첩된 배열의 요소를 모두 꺼내어 단일 배열로 변환합니다. 그 외의 경우는 입력 값을 그대로 반환합니다. [strjoin()](https://docs.logpresso.com/ko/query/strjoin-function)을 사용하여 배열을 하나의 문자열로 병합하기 전에, 중첩된 배열 요소들을 단일 배열로 변환하는 용도로 사용합니다.

#### 문법

flatten(ARRAY\_EXPR)

필수 매개변수

**ARRAY\_EXPR**

단일 배열로 변환할 값을 반환하는 표현식

#### 사용 예

json "{}" | eval array=flatten(array(1, array(2, 3), 4))| # 반환값: [ 1, 2, 3, 4 ]

### foreach()

배열 내의 인자를 빼내지 않고 배열을 대상으로 혹은 복수의 배열끼리 연산을 수행합니다.

#### 문법

foreach(OP\_EXPR, LIST\_EXPR\_1, LIST\_EXPR\_2, ...)

필수 매개변수

**OP\_EXPR**

배열 요소 사이에 수행할 연산식. 첫번째 배열의 요소는 \_1, 두 번째 배열의 요소는 \_2, N 번째 배열 요소는 \_N을 매개변수로 사용

**LIST\_EXPR\_1, LIST\_EXPR\_2, ...**

배열을 반환하는 표현식을 쉼표(,)로 구분하여 지정

#### 설명

매개변수로 전달되는 배열들의 길이가 같지 않으면 긴 배열을 구성하는 요소의 개수에 맞추어 짧은 배열에 null이 할당된 요소를 채운 후에 연산을 수행합니다. 예를 들어, 첫번째 배열이 5개 요소, 두 번째 배열이 3개 요소로 구성되어 있으면 두 번째 배열에 값이 null인 요소를 2개 더 추가한 뒤에 연산을 수행합니다.

인자로 목록 대신 스칼라 값이 전달되면 리스트로 복제하여 확장 첫번째 리스트를 \_1, 두번째 리스트를 \_2의 방식으로 치환하여 OP\_EXPR에 따라 각각 연산합니다.

#### 사용 예

json "{}" | eval arr1= array(-1, -2, -3, -4, -5), arr2= array(1,2,3,4,5) | eval \_output = foreach(\_1 \* \_2, arr1, arr2) | order arr1, arr2, \_output=> [-1,-4,-9,-16,-25]

### subarray()

인자로 지정한 배열의 부분 배열을 반환합니다.

#### 문법

subarray(ARRAY\_EXPR, INT\_START, [INT\_END])

**ARRAY\_EXPR**

배열을 반환하는 표현식

**INT\_START**

배열에서 잘라낼 구간의 시작 인덱스 번호. 번호는 0부터 시작합니다.

**INT\_END**

배열에서 잘라낼 구간의 끝 인덱스 번호. 끝 인덱스 번호의 배열 요소는 부분 배열에 포함되지 않습니다.

#### 설명

배열 요소의 인덱스 번호는 0부터 시작합니다. 5 개의 요소를 갖는 배열에서 인덱스 번호는 처음부터 0, 1, 2, 3, 4가 됩니다. INT\_START, INT\_END에 음수를 지정할 수 있습니다. 5 개의 요소를 갖는 배열에서 음수로 부여하는 인덱스 번호는 처음부터 -5, -4, -3, -2, -1입니다.

#### 사용 예

일반적인 사용 예

json "{}" | eval parent=array(1, 2, 3, 4, 5) | eval child=subarray(parent, 2) | # 반환 값: parent: [1, 2, 3, 4, 5] child: [3, 4, 5]

json "{}" | eval arr=subarray(array(1, 2, 3, 4, 5), 2, 4) | # 반환 값: arr: [3, 4]

json "{}" | eval arr=subarray(array(1, 2, 3, 4, 5), 1, -1) | # 반환 값: arr: [2, 3, 4]

유효 범위를 벗어나는 INT\_START 또는 INT\_END를 지정한 예

json "{}" | eval arr=subarray(array(1, 2, 3, 4, 5), 5) | # 반환 값: null

json "{}" | eval arr=subarray(array(1, 2, 3, 4, 5), 0, 5) | # 반환 값: arr: [1, 2, 3, 4, 5]

### sumarray()

인자로 지정한 배열의 요소들을 모두 더한 결과를 반환합니다. 숫자가 아니거나, null인 요소들은 연산에서 제외됩니다.

#### 문법

subarray(ARRAY\_EXPR)

**ARRAY\_EXPR**

배열을 반환하는 표현식

#### 사용 예

json "{}"| eval sum=sumarray(array(1, 2, 3, 4, 5))| # 반환 값: 15

json "{}"| eval arr=sumarray(array(1, 2, 3, null, "a", 4, 5))| # 반환 값: 15

json "{}"| eval arr=sumarray(array(null, null, null, "a", "b"))| # 반환 값: 0

### unique()

표현식 값이 배열인 경우 중복된 원소를 제거한 배열을 반환하고, 단일 값을 인자로 받으면 하나의 원소만 포함한 배열을 반환합니다.

#### 문법

unique(EXPR)

**EXPR**

중복된 원소를 제거할 배열을 반환하는 표현식. 이 때 반환되는 배열의 순서는 보장하지 않습니다. 표현식이 단일 값인 경우, 하나의 원소만 포함한 배열을 반환합니다. 표현식이 null인 경우 null을 반환합니다.

#### 사용 예

1, 1, 2, "2" 배열에서 중복된 원소를 제거

json "{}"| eval array=unique(array(1, 1, 2, "2"))| # 반환하는 값: ["2", 1, 2]

### valueof()

매개변수로 주어진 배열이나 복합 객체에서 특정 키, 인덱스에 해당하는 위치의 값을 반환합니다.

#### 문법

valueof(COMPOUND\_OBJ\_EXPR, KEY\_EXPR)

**COMPOUND\_OBJ\_EXPR**

맵이나 배열과 같은 복합 객체를 반환하는 표현식

**KEY\_EXPR**

맵의 키, 배열의 인덱스와 같이 요소의 특정한 위치의 값을 가리키는 표현식

#### 설명

맵이나 배열에서 특정 키에 해당하는 값을 반환합니다. 다음과 같은 예외 상황에서, null을 반환합니다.

* 복합 객체 표현식에 맵이나 배열이 아닌 객체를 입력했을 때
* 맵의 키와 키 표현식의 타입이 일치하지 않을 때
* 배열의 인덱스 숫자와 키 표현식의 타입이 일치하지 않을 때

#### 사용 예

원소 개수가 3개인 배열에서 2번 항목을 추출 (배열에서 인덱스는 0부터 시작)

json "{}" | eval foods=array("Apple","Banana","Cucumber") | eval food=valueof(foods,2) | # 반환값: "Cucumber"

맵 객체에서 키가 "b"인 항목을 추출

json "{}" | eval foods=dict("a","Apple","b","Banana","c","Cucumber" ) | eval food = valueof(foods,"b") | # 반환값: "Banana"

## 이벤트 컨텍스트 함수

### evtctxget()

키와 연관된 이벤트 컨텍스트를 선택하여 속성 정보를 조회합니다. 지정된 키를 가진 이벤트 컨텍스트가 존재하지 않으면 null을 반환합니다.

#### 문법

evtctxget(TOPIC, KEY, ATTR)

필수 매개변수

**TOPIC**

조회할 이벤트 컨텍스트의 주제

**KEY**

조회할 이벤트 키

**ATTR**

속성 문자열. 아래의 문자열 중 하나를 지정할 수 있습니다:

* counter: 이벤트 카운터, 동일한 키에 대하여 이벤트가 발생한 횟수를 정수로 반환합니다. 항상 1 이상의 값을 반환합니다.
* created: 이벤트가 최초로 발생한 시각을 날짜 타입으로 반환합니다.
* expire: 만료 시각이 지정된 경우, 이벤트 컨텍스트가 삭제될 시각을 날짜 타입으로 반환합니다.
* timeout: 타임아웃이 지정된 경우, 이벤트 컨텍스트가 타임아웃으로 삭제될 시각을 날짜 타입으로 반환합니다.
* rows: 현재 이벤트 컨텍스트에 저장된 모든 레코드를 배열로 반환합니다.

#### 사용 예

주제가 txmatch, txkey가 001122, 타임아웃 10초의 이벤트 컨텍스트가 존재할 때:

evtctxget("txmatch", "001122", "counter") => 1evtctxget("txmatch", "001122", "created") => "Fri May 02 15:21:50 KST 2014"evtctxget("txmatch", "001122", "expire") => nullevtctxget("txmatch", "001122", "timeout") => "Fri May 02 15:22:00 KST 2014"evtctxget("txmatch", "001122", "rows") => [{txkey=001122, type=send}]

### evtctxgetvar()

키와 연관된 이벤트 컨텍스트에서 사용자 변수를 조회합니다. 지정된 키를 가진 이벤트 컨텍스트가 존재하지 않거나, 변수가 존재하지 않으면 null을 반환합니다.

#### 문법

evtctxgetvar(TOPIC, KEY, VARIABLE)

필수 매개변수

**TOPIC**

조회할 이벤트 컨텍스트 주제

**KEY**

조회할 이벤트 키

**VARIABLE**

조회할 사용자 변수

#### 사용 예

web\_session 컨텍스트의 식별자로 sessionkey 필드 값을 사용하고, client\_ip 변수를 조회하는 예:

evtctxgetvar("web\_session", sessionkey, "client\_ip")

### evtctxsetvar()

키와 연관된 이벤트 컨텍스트에 사용자 변수를 설정합니다. 이벤트 컨텍스트가 존재하지 않거나, 변수 이름 표현식을 평가한 결과가 null인 경우 false를 반환하고, 변수 설정에 성공한 경우 true를 반환합니다.

#### 문법

evtctxsetvar(TOPIC, KEY, VARIABLE, VALUE)

필수 매개변수

**TOPIC**

이벤트 컨텍스트 주제

**KEY**

조회할 이벤트 키

**VARIABLE**

조회할 사용자 변수

**VALUE**

변수에 할당할 값을 반환하는 표현식

#### 사용 예

web\_session 컨텍스트의 식별자로 sessionkey 필드 값을 사용하고, client\_ip 변수를 ip 필드 값으로 설정하는 예:

evtctxsetvar("web\_session", sessionkey, "client\_ip", ip)

## 객체 그룹 함수

### matchbehavior()

행위 프로파일을 검색하여 프로파일 된 데이터가 있는 경우 true, 없으면 false를 반환합니다.

#### 문법

matchbehavior(STR\_GUID, KEY\_EXPR,...)

필수 매개변수

**STR\_GUID**

행위 프로파일 GUID. GUID 문자열은 유효한 행위 프로파일 식별자이어야 합니다. 유효하지 않은 행위 프로파일 GUID를 지정한 경우 쿼리가 실패합니다.

**KEY\_EXPR,...**

쉼표(,)를 구분자로 하는 키 표현식 목록. 키 매개변수의 순서는 행위 프로파일에 지정된 키 필드의 순서와 동일해야 합니다. 키 매개변수의 수는 행위 프로파일 설정에 지정된 키 필드의 수와 동일해야 합니다. 키 표현식의 평가 값은 문자열 혹은 IP 주소 타입만 허용됩니다. 허용되지 않는 타입이 인자로 전달된 경우, false 값을 반환합니다.

### matchblackip()

주소 그룹에 대상 표현식의 평가 값이 포함되어 있으면 true, 없으면 false를 반환합니다.

#### 문법

matchblackip(STR\_GUID, IP\_ADDR\_EXPR)

필수 매개변수

**STR\_GUID**

주소 그룹 GUID. GUID 문자열은 유효한 주소 그룹 식별자이어야 합니다. 유효하지 않은 GUID를 지정한 경우 쿼리가 실패합니다.

**IP\_ADDR\_EXPR**

IP 주소 표현식. 대상 표현식은 IPv4 주소 혹은 IPv4 문자열만 허용됩니다. 대상 표현식에서 허용되는 타입이 아닌 경우 false 값을 반환합니다.

### matchfeed()

위협 인텔리전스 피드에서 대상 표현식의 평가 값이 검색되면 true, 검색되지 않으면 false를 반환합니다.

#### 문법

matchfeed(STR\_FEED, STR\_EXPR)

필수 매개변수

**STR\_FEED**

위협 인텔리전스 피드 문자열. 피드 문자열은 아래의 유형을 사용할 수 있습니다. 유효하지 않은 피드 문자열 상수를 사용한 경우 쿼리가 실패합니다.

|  |  |  |  |
| --- | --- | --- | --- |
| 이름 | 유형 | 이름 | 설명 |
| otx | IP 주소 | OTX 피드 | ISAC이 제공하는 OTX (Open Threat Exchange) IP 주소 피드 |
| tor | IP 주소 | TOR 프록시 | ISAC이 제공하는 Tor 네트워크의 EXIT 라우터 IP 주소 피드 |
| mdl\_domain | 도메인 | 악성 도메인 주소 목록 | ISAC이 제공하는 랜섬웨어 등 악성코드 도메인 주소 피드 |
| mdl\_ip | IP 주소 | 악성 IP 주소 목록 | ISAC이 제공하는 악성 IP 주소 피드 |
| abusech | 도메인 | abuse.ch | ISAC이 제공하는 abuse.ch 악성 도메인 주소 피드 |
| malc0de | MD5 | malc0de | ISAC이 제공하는 malc0de.com 악성코드 MD5 해시 피드 |
| logpresso\_cti\_ip | IP 주소 | 악성 IP 주소 목록 | 로그프레소 CTI 서비스에서 제공하는 악성 IP 주소 피드 |
| logpresso\_cti\_domain | 도메인 | 악성 도메인 주소 목록 | 로그프레소 CTI 서비스에서 제공하는 도메인 주소 피드 |
| logpresso\_cti\_url | URL | 악성 URL 목록 | 로그프레소 CTI 서비스에서 제공하는 URL 피드 |
| logpresso\_cti\_md5 | MD5 | 악성코드 MD5 해시 목록 | 로그프레소 CTI 서비스에서 제공하는 MD5 해시 피드 |
| logpresso\_cti\_sha1 | SHA1 | 악성코드 SHA1 해시 목록 | 로그프레소 CTI 서비스에서 제공하는 SHA1 해시 피드 |
| logpresso\_cti\_sha256 | SHA256 | 악성코드 SHA256 해시 목록 | 로그프레소 CTI 서비스에서 제공하는 SHA256 해시 피드 |

**STR\_EXPR**

위협 인텔리전스 피드에서 검색할 문자열

### matchnet()

네트워크 대역에 IP 주소 표현식의 평가 값이 포함되어 있으면 true, 없으면 false를 반환합니다.

#### 문법

matchnet(STR\_GUID, IP\_ADDR\_EXPR)

필수 매개변수

**STR\_GUID**

네트워크 대역 GUID. GUID 문자열은 유효한 네트워크 대역 식별자이어야 합니다. 유효하지 않은 네트워크 대역 GUID를 지정한 경우 쿼리가 실패합니다.

**IP\_ADDR\_EXPR**

IP 주소 표현식. 대상 표현식은 IPv4 주소 혹은 IPv4 문자열만 허용됩니다. 대상 표현식에서 허용되는 타입이 아닌 경우 false 값을 반환합니다.

### matchport()

지정된 포트 그룹에 포트 및 프로토콜 조합이 포함되어 있으면 true, 없으면 false를 반환합니다.

#### 문법

matchport(PORT\_GUID, PORT\_EXPR[, PROTO\_EXPR])

필수 매개변수

**PORT\_GUID**

포트 그룹 GUID. GUID 문자열은 유효한 포트 그룹 식별자이어야 합니다. 유효하지 않은 포트 그룹 GUID를 지정한 경우 쿼리가 실패합니다.

**PORT\_EXPR**

포트 번호 표현식. 평가 값은 0 ~ 65535 사이의 정수이어야 합니다. 표현식을 평가할 수 없거나 유효하지 않은 값인 경우 false 를 반환합니다.

선택 매개변수

**PROTO\_EXPR**

프로토콜 표현식. TCP 혹은 UDP 문자열만 허용됩니다. 표현식을 평가할 수 없거나 유효하지 않은 값인 경우 false를 반환합니다. 표현식의 평가 값이 null일 때 포트 그룹이 TCP, UDP 중 하나라도 포함하면 true를 반환합니다.

### matchsig()

대상 표현식을 평가한 문자열에서 지정된 패턴 그룹 중 하나 이상의 패턴이 매칭되면 true, 매치되는 패턴이 없으면 false를 반환합니다.

#### 문법

matchsig(STR\_GUID, STR\_EXPR)

**STR\_GUID**

패턴 그룹 GUID. GUID 문자열은 유효한 패턴 그룹 식별자이어야 합니다. 유효하지 않은 패턴 그룹 GUID를 지정한 경우 쿼리가 실패합니다.

**STR\_EXPR**

검색 대상 문자열 표현식. 평가 값은 문자열이어야 합니다. 표현식을 평가할 수 없거나 유효하지 않은 값인 경우 false 를 반환합니다.

#### 설명

예를 들어, 아래 표와 같이 패턴 그룹에 패턴 "REMOTE\_ADDR" and ("fputs" or "fwrite") 검증식 path == "lib.php" 를 "zb now connect"라는 패턴 이름으로 설정한 경우, 대상 표현식 평가 값에서 REMOTE\_ADDR 문자열과 fputs 혹은 fwirte 문자열이 동시에 검색되는지 확인하고, 그 후에 path 필드 값이 lib.php 문자열과 일치하는지 확인합니다.

**패턴 예시**

|  |  |  |
| --- | --- | --- |
| expr (필수) | expr2 (선택) | rule (필수) |
| 키워드 패턴: 1차 고속 탐지 | 불린 표현식: 2차 필터링 | 패턴 이름 |
| "addextendedproc" and "xp\_cmdshell" |  | xp\_cmdshell |
| "REMOTE\_ADDR" and ("fputs" or "fwrite") | path == "lib.php" | zb now\_connect |

# 집계 함수

### array()

그룹에 속한 값들의 배열을 생성합니다. 그룹 당 최대 100개의 항목을 수집하여 중복된 값도 포함한 집합으로 배열을 생성합니다.

그룹에 속한 모든 유일한 값의 집합을 추출하고 싶다면 [values()](https://docs.logpresso.com/ko/query/values-aggregate-function) 함수를 사용하십시오.

#### 문법

array(EXPR)

필수 매개변수

**EXPR**

배열로 변환할 필드를 반환하는 표현식

### avg()

표현식이 반환하는 숫자값들의 평균을 계산합니다. 반환하는 값이 null이거나, 숫자가 아닌 값을 반환하는 표현식은 무시합니다.

#### 문법

avg(NUM\_EXPR)

필수 매개변수

**NUM\_EXPR**

숫자값들을 반환하는 표현식

#### 사용 예

# 1~100 범위의 숫자로 무작위 100개의 레코드를 갖는 필드를 2개 만들고 평균 계산 | json "{}" | repeat count=100 | eval n1=rand(101), n2=rand(101) | stats avg(n1) as n1\_avg, avg(n2) as n2\_avg

### corr()

그룹별 피어슨 상관계수를 계산합니다. EXPR\_X, EXPR\_Y 두 표현식 중 하나라도 null을 반환하거나, 숫자가 아닌 값을 반환하면 무시합니다.

#### 문법

corr(EXPR\_X, EXPR\_Y)

필수 매개변수

**EXPR\_X**

피어슨 상관계수를 계산할 필드 목록 표현식

**EXPR\_Y**

피어슨 상관계수를 계산할 필드 목록 표현식

#### 사용 예

json "{}" | repeat count=100| eval n1=seq(), n2=sqrt(seq()) | stats corr(n1, n2)

### count()

그룹별로 행 수를 계산합니다. 표현식을 입력하지 않으면 전체 행 수를 반환합니다. 표현식이 지정된 경우에는 null이 아닌 숫자값을 반환합니다.

#### 문법

countcount(EXPR)

선택 매개변수

**EXPR**

그룹값을 반환하는 표현식

### cov()

그룹별 공분산을 계산합니다. EXPR\_X, EXPR\_Y 두 표현식 중 하나라도 null을 반환하거나, 숫자가 아닌 값을 반환하면 무시합니다.

#### 문법

cov(EXPR\_X, EXPR\_Y)

필수 매개변수

EXPR\_X:그룹별 공분산을 계산할 필드 목록 표현식

**EXPR\_Y**

그룹별 공분산을 계산할 두 번째 필드 목록 표현식

### dc()

그룹에 속한 유일한 값들의 개수를 추출합니다.

#### 문법

dc(EXPR)

필수 매개변수

**EXPR**

그룹값을 반환하는 표현식

### estdc()

그룹에 속한 유일한 값들의 근사치를 추출합니다.

#### 문법

estdc(EXPR[, NUMBER])

필수 매개변수

**EXPR**

그룹값을 반환하는 표현식

선택 매개변수

**NUM**

4 ~ 24 사이의 비트수(기본값: 16). 비트수가 높을수록 정확도와 메모리 사용량이 증가합니다.

### first()

그룹에 속한 표현식 중에서 첫번째 표현식의 값을 반환합니다.

#### 문법

first(EXPR)

필수 매개변수

**EXPR**

그룹값을 반환하는 표현식

### last()

그룹에 속한 표현식 중에서 마지막 표현식의 값을 반환합니다.

#### 문법

last(EXPR)

필수 매개변수

**EXPR**

그룹값을 반환하는 표현식

### max()

그룹에 속한 표현식 중에서 최대값을 계산합니다. null인 표현식은 무시됩니다. 서로 다른 타입 간의 비교는 정의되지 않은 동작을 수행합니다.

#### 문법

max(EXPR)

필수 매개변수

**EXPR**

그룹값을 반환하는 표현식

### min()

그룹에 속한 표현식 중에서 최소값을 계산합니다. null인 표현식은 무시됩니다. 서로 다른 타입 간의 비교는 정의되지 않은 동작을 수행합니다.

#### 문법

min(EXPR)

필수 매개변수

**EXPR**

그룹값을 반환하는 표현식

### slope()

그룹별 X/Y 선형 회귀선의 기울기를 계산합니다. EXPR\_X, EXPR\_Y 두 표현식 중 하나라도 null을 반환하거나, 숫자가 아닌 값을 반환하면 무시합니다.

#### 문법

slope(EXPR\_X, EXPR\_Y)

필수 매개변수

**EXPR\_X**

선형 회귀선의 기울기를 계산할 숫자값들을 반환하는 표현식

**EXPR\_Y**

선형 회귀선의 기울기를 계산할 숫자값들을 반환하는 표현식

### stddev()

그룹에 속한 모든 표현식의 표준편차를 계산합니다. null을 반환하거나, 숫자가 아닌 값을 반환하는 표현식은 무시합니다.

#### 문법

stddev(NUM\_EXPR)

필수 매개변수

**NUM\_EXPR**

숫자값들을 반환하는 표현식

### sum()

그룹에 속한 모든 표현식의 합을 계산합니다. null을 반환하거나, 숫자가 아닌 값을 반환하는 표현식은 무시합니다.

#### 문법

sum(NUM\_EXPR)

필수 매개변수

**NUM\_EXPR**

숫자값들을 반환하는 표현식

### values()

그룹에 속한 모든 유일한 값의 집합을 추출합니다. 그룹 당 최대 100개의 항목을 수집하여 유일한 값의 집합으로 배열을 생성합니다.

#### 문법

values(EXPR)

필수 매개변수

**EXPR**

그룹값을 반환하는 표현식

### var()

그룹에 속한 모든 표현식의 분산을 계산합니다. null을 반환하거나, 숫자가 아닌 값을 반환하는 표현식은 무시합니다.

#### 문법

var(NUM\_EXPR)

필수 매개변수

**NUM\_EXPR**

숫자값들을 반환하는 표현식