Logpresso Query

# Overview

# Before You Start

## User Interface Notation

Graphic user interface (GUI) elements are expressed as follows.

**GUI Conventions**

|  |  |
| --- | --- |
| Notation Format | Description |
| **Menu 1 > Menu 2** | Displays the multi-level menu path using ">" in bold. |
| **[Tab]** | Displays the name of the tab |
| **Button** | Displays button names in bold. |

## Command Notation

The following table lists the notational conventions used for commands, options and input values in this manual.

**Conventions for Commands**

|  |  |
| --- | --- |
| Notation Format | Description |
| table araqne\_query\_logs | Monospaced lowercase to input what you type as is. |
| VALUE, TABLE, TABLE.INDEX, FIELD | Monospaced uppercase to express what you need to enter according to the user environment. |
| opt=value, [opt=value] | Parameters of the commands in the form of "option=value". A pair of square brackets ([ ]) means this is an option you can omit. |
| opt={true|false}, opt=INT{s|m|d|w|mon} | Parameters that require you to select and specify one of the multiple values are enclosed in a pair of curly braces ({ }). |

This manual uses the conventions listed above when describing the syntax of commands in this manual. For example, the syntax of the [stream](https://docs.logpresso.com/en/query/stream-command) command is as follows:

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

## Acronyms, Abbreviations and Terms

This manual uses the following terms:

**ENT**

Abbreviation for Logpresso Enterprise

**FRS**

Abbreviation for Logpresso Forensic

**GUID**

Abbreviation for Globally Unique Identifier. A hexadecimal value generated from the unique identifier

**MAE**

Abbreviation for Logpresso Maestro

**SNR**

Abbreviation for Logpresso Sonar

**STD**

Abbreviation for Logpresso Standard

**Table**

Database object that contains all the data in a database

**Web console**

Web-based graphical user interface provided within Logpresso's suite

### Running Queries in the Web Console

The standard SQL is a declarative language that allows you to obtain the data you want without having to specify the detailed process of data processing. However, the standard SQL has many constraints on processing unstructured data, as well as the drawback that the description of the streaming process is not natural.

Logpresso has inherited the design philosophy of Unix-based operating systems that maximize simplicity, cohesiveness, reusability, and flexibility. One command performs the smallest and the simplest task, but combining multiple commands helps effectively process complex and unstructured data.

The following sections describe how to use queries in the Logpresso web console and the basic structure of Logpresso query.

#### Query Menu Path by Product

You can use queries in the Logpresso web console. There are many interfaces where you can input query string, but a separate interface for running queries is also provided. The query menu by product suite is as follows:

* ENT, STD: **Query** or **Query > Query**
* MAE, SNR: **Analysis > Query**

#### Executing a Query

To execute a query, input the query string in the input box and click **Run**. The query string may be a short statement using a single command or may consist of multiple commands where data is passed from one to the next using pipes (|).

#### Query Keyboard Shortcuts

Query input box supports following keyboards shortcuts.

**Query Keyboard Shortcuts**

|  |  |
| --- | --- |
| Function | Keyboard shortcut |
| Executing a query | **Ctrl+Enter** or **Shift+Enter** |
| Viewing the command list and help | **Ctrl+Space** |
| Automatic alignment and indentation of query | **Ctrl+Shift+F** |

For viewing the command list or help (**Ctrl+Space**), the operation varies depending on whether you input a command.

* If you press the shortcut keys without providing a command, you can see the command list appears.
* If you press the shortcut keys after providing a command, you can see a list of options available appears.

Automatic alignment and indentation of query commands help the user understand long query commands written across multiple lines by automatically applying indentations and line breaks. This shortcut is only supported by MAE and SNR.

### Query Types

Query types of Logpresso are roughly classified into four types depending on how they are executed.

#### Ad-hoc Query

An ad-hoc query refers to arbitrary queries that you can create and execute at any given time. You can programmatically execute queries at any given time through the log query menu of Logpresso's web console, the Logpresso terminal accessed via SSH, or the Logpresso client SDK.

If you switch a query that runs for a long time to the background, it will continue to run even when the current session is logged out or disconnected. You can then switch that query back to the foreground to see the result of the query.

#### Real-time Query

A real-time query refers to a query that processes data received in real time while running for a specified time range from the point of execution. It can use the logs in real time as soon as logs are collected by the log collector, the results of the stream query are output, or the data is provided in a table. This is useful for diagnosis because it immediately samples data without storing all the data on the disk.

Real-time query commands include [logger](https://docs.logpresso.com/en/query/logger-command), [stream](https://docs.logpresso.com/en/query/stream-command), and [table](https://docs.logpresso.com/en/query/table-command).

#### Stream Query

A stream query refers to a query that runs indefinitely in the background for real-time data sources until the system shuts down. A stream query runs queries while continuously guaranteeing the input sequence.

For ENT and STD, you can find stream queries in **Query > Stream**. Stream queries support following three stream types as input:

Log Collector

All logs collected by the logger are provided into the stream query. For ENT and STD, you can configure the logger in **LOG SOURCE**, or for MAE and SNR, in **Collect > Connector**.

Table

Every time a new row is written to the table, it is provided into the stream query. It can be thought of as an evolved use case of the triggers used in a relational database management system (RDBMS). For ENT and STD, you can configure the table in **Table**, or for MAE and SNR, in **System > Table**.

Stream Query

You can use the output of other stream queries as input. Take, for example, a scenario in which a stream query that parses for unstructured logs is placed at the frontend, and multiple analysis stream queries that use that stream query as input are placed. A stream query is divided into streaming mode and refresh mode.

**Streaming mode**

If the query is configured only with commands—streamable commands—that do not rely on data input completion, you can set the stream query in streaming mode.

**Refresh mode**

For example, in the case of statistics and sorting, you can run operations only after the input of the entire data is completed, so the input completion signal is passed at regular intervals.

If you use stream queries to calculate statistics for specific time units and to store the result in an intermediate statistics table, and you design this table to query for the final statistical results, you can get statistical results in real time for large data streams while using very little disk space. You can use Groovy scripting for highly complex real-time analysis and processing.

#### Scheduled Query

Scheduled queries are executed according to the schedule specified by the user. You can selectively save query results and send the results that match the alarm conditions by email.

For ENT and STD, you can load the query results using the **Saved query results** in **Query > Load**.

### Query Syntax

#### Format of Commands

A query consists of one or more commands. The basic units that make up a command are name of command, options, and objects.

Commands with Target Objects

The object may be a log collector, stream query, log parser, table where data is stored, or full-text index. Expressions or subquery commands are supported depending on the command. The composition of the command statements executed on such objects is as follows.

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

As an example of the simplest command statement, a command that queries the data in the system table araqne\_query\_logs is as follows:

table araqne\_query\_logs

Commands with No Target Objects

Commands with no objects are primarily used in commands that receive and process data from other commands. An example is a [decodedns](https://docs.logpresso.com/en/query/decodedns-command) command. The composition is as follows.

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

These commands receive and process the output returned by the forwarding command (FORWARDING\_STATEMENT) as an input through a pipe (|).

#### Input Processing Using Pipes

In Logpresso, the output of one command can be passed to another command as an input using a pipe (|). For example, the following query only searches the logs whose **login\_name** field is "root" in the **araqne\_query\_logs** table.

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

This query shows queries executed by the root account. Use the following query to calculate the statistics in 10-minute increments of how many rows containing the string "root".

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

In this way, the output of the first command is transferred to the input of the second command, the output of the second to the input of the third command, and the output of the third to the result of the query. The results of the query can be temporarily written to the disk or streamed instantly over the network, depending on the client's request.

#### Subquery

Some query commands execute nested commands in the command statement, receive the results, and then execute them. A nested command is called a subquery.

To express a subquery, enclose it within a pair of square brackets ([ ]). Subqueries are executed ahead of main query command. The records returned by a subquery are processed by the main query.

When there is a subquery, the structure of the command is as follows:

command [ SUBCOMMAND\_STATEMENT ]

#### Comments

You can use the comment out command '#' to insert a description in the command line or comment out a single command line or consecutive command lines. In the query input box, the commented-out command line is grayed out.

Single-Line Comments

You can insert a '#' at the beginning of the command line to treat it as a comment.

A whitespace character is required after the '#'. If there is no whitespace character, commenting out is not applied.

# Querying the CPU usage recorded in sys\_cpu\_log for the last hour | table duration=1h sys\_cpu\_logs | # eval total = kernel + user

In the above example, Querying the CPU usage recorded in sys\_cpu\_log for the last hour and eval total = kernel + user are commented out, and only table duration=1h sys\_cpu\_logs is executed.

Multi-line Comments

To create a multi-line comment, put # [ at the start of your comment and ] at the end.

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

In the above example, the command inside the brackets is commented out, so the query that is executed is table duration=1h sys\_cpu\_logs | sort \_time.

The comment out command # ignores subsequent strings and pipes (|) within the pair of square brackets ([ ]) that encloses the subquery. It ignores any line breaks in the subquery. In other words, it ignores the entire subquery and comments it out until you see the pipe outside the pair of square brackets.

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

In the above example, the subquery union [ table sys\_cpu\_logs | limit 30 ] is all commented out, so the query that is executed is table sys\_cpu\_logs | eval total = kernel + user.

#### Query Parameter

You can assign values to query parameters and use them as needed. This is useful when dynamically assigning values and executing queries because it uses expressions with functions instead of constants. You can use query parameters when executing a scheduled query, if you want to look up and process data for a week-range based on the current date, or if you want to execute a query using the parameter value you provide when running the procedure.

Declaration of Parameters

You can declare parameters using [set](https://docs.logpresso.com/en/query/set-command) or [setq](https://docs.logpresso.com/en/query/setq-command).

References to Parameters

You can refer to the value assigned to a parameter using the parameter reference function [$()](https://docs.logpresso.com/en/query/$-function).

#### Function

You can use functions in your query command. You can use a function wherever you can use an expression. However, the value returned by the function must be processable by the expression.

#### Procedure

Logpresso provides procedure that allows you to call the predefined query command like a function. This is similar to a DBMS procedure, and provides the following benefits:

**Improve reusability and maintenance**

You can improve reusability by modularizing queries that provide specific functionality through procedures. You just need to tell the user the name of the procedure and the parameters to use. You can maintain procedure with ease as you can redefine them without affecting other code.

**Improve Security**

Commands that connect to external systems such as [dbquery](https://docs.logpresso.com/en/query/dbquery-command), [ftp](https://docs.logpresso.com/en/query/ftp-command), and [sftp](https://docs.logpresso.com/en/query/sftp-command) require profile privileges. Granting profile privileges directly to the user is insecure because the user can run arbitrary operations from the external system. However, Logpresso allows users to execute commands that require specific administrative privileges or configure commands that can run arbitrary tasks on local/remote hosts as procedures, and then manage user privileges so that users can use the desired tasks without a granted system-wide administrative privileges. For example, you can apply the administrative privileges in a way that limits user privileges to view a part of the source data, or masks and displays the source data.

**Access to Logpresso’s system table**

To access the Logpresso's system table, you must have administrative privileges. When the user needs to access the Logpresso's system setting information, Logpresso allows the user to access it through the procedure.

Defining the Procedure

You can define and manage procedures in the web console. Procedure management is available in the following paths:

* (ENT, STD) **Query > Procedure**
* (MAE, SNR) **Analysis > Procedure**

A query command to be used as a procedure may contain parameters or custom fields to use when calling the procedure.

The query defined in the procedure can use the [$()](https://docs.logpresso.com/en/query/$-function) function to refer to parameters passed by the user when calling the procedure. The following is an example:

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

In the example query command, threshold is the parameter.

The most common mistake in writing a procedure is to write a query thinking that the [$()](https://docs.logpresso.com/en/query/$-function) function reference is replaced like a macro. The [$()](https://docs.logpresso.com/en/query/$-function) function can only be specified where an expression can be assigned in a query command. For example, the following procedure is not the correct query because [dbquery](https://docs.logpresso.com/en/query/dbquery-command) does not support an arbitrary SQL statement input as an expression.

dbquery USERDB $("sql")

Calling Procedure

[proc](https://docs.logpresso.com/en/query/proc-command) command calls and executes the procedure. See the description of the command on how to call it.

# Enterprise Commands

## Parameters

### set

Evaluates expressions using functions or the like and then assigns that value to the parameter.

#### Syntax

set VAR\_NAME=EXPR

Required Parameter

**VAR\_NAME = EXPR**

Value obtained by evaluating the expression to the query parameter. On the right side of the command, you can use any expression that can be evaluated without a record at the time the query is executed.

* It works regardless of whether whitespaces are inserted before or after the assignment operator (=).
* You can use any expression that can be evaluated without a record at the beginning of the query.
* The query parameter is valid for one query instance while it is alive.
* If there are multiple set commands, they are evaluated in order from the left.

#### Description

The query parameter is valid for one query instance while it is alive, and you can use the set command to evaluate the parameter value at the time the query is parsed. The query string below is an example of using the [table](https://docs.logpresso.com/en/query/table-command) command to dynamically query data from 00:00 hour three days before to before 00:00 hour on the current day.

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

You can use the set command to set the query parameter as described above, and you can use the [$()](https://docs.logpresso.com/en/query/$-function) function to refer the value of the query parameter.

The options of each query command can be replaced with query parameters. For example, you can use query parameters when executing a scheduled query, if you want to look up and process data for a week range based on the current date, or if you want to execute a query using the parameter value you provide when running the procedure.

When you call the procedure, the values passed to the procedure's parameters are set as query parameters. Therefore, when you create or edit a procedure, you can use it with a [$()](https://docs.logpresso.com/en/query/$-function) function by assuming that the query string already has a value corresponding to the procedure parameter.

### setq

Executes a subquery and sets the key-value pair in the first record as the query parameter.

#### Syntax

setq [ SUBQUERY ]

Required Parameter

**[ SUBQUERY ]**

Subquery command enclosed in a pair of square brackets ([ ]).

#### Description

This sets the field-value pair present in the first record of the result of executing the subquery as a query parameter. If the subquery returns one or more results, the command ignores them from the second record.

Subqueries consisting of setq commands are executed before all other commands in the entire query command. If there are multiple setq commands, they are executed sequentially.

## Data Source

### csvfile

Loads data in a comma-separated values (CSV) or tab-separated values (TSV) file. This loads the header information in the first line of the CSV or TSV file and uses it as a field name.

#### Syntax

csvfile [OPTIONS] PATH

Required Parameter

**PATH**

Path to the file from which you want to load the data. Using a wildcard (\*) in the file name, you can load all files containing a specific string pattern in the file name at once. For example, if you enter allow-\*.csv in PATH, you can load all files, such as "allow-ip.csv", "allow-user.csv", "allow-url.csv" at once. To load a file, the Logpresso daemon must have the read permission to the file.

Optional Parameter

**cs=CHARSET**

Character set (default: utf-8). This option is case-insensitive. Use the preferred MIME name or aliases registered in the following document as CHARSET: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**limit=INT**

Maximum number of records to load (default: unlimited).

**maxcol=INT**

Maximum number of columns to load (default: 10,000). If the maximum number of columns is exceeded, use the rest option to define the processing method.

**offset=INT**

Number of records to skip (default: 0).

**rest=BOOL**

Boolean option to process the column data exceeding the maximum number specified by the maxcol option (default: f). See usage #3 and #4.

* t: Puts data beyond the maximum number of columns specified by the maxcol option in the **\_rest** field.
* f: Discards the rest of the columns beyond the maximum number of columns specified by the maxcol option.

**strict=BOOL**

Compliance with RFC4180 (<https://tools.ietf.org/html/rfc4180>) (default: f). See Usage #5 to #8.

* t: Strictly parses to conform RFC 4180 as the same as when you open the CSV file with Microsoft Excel. This option cannot be used when tab=t.
* f: Flexibly parses the CSV file.

**tab=BOOL**

Option to use tab character as a separator (default: f).

* t: Uses tab character as a separator. This is useful for processing tab-separated values (TSV) files.
* f: Uses comma (,) as a separator.

#### Usage

Read the /opt/logpresso/wp-nginx.csv file.

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

Read 20 records after skipping the header line of /opt/logpresso/wp-nginx.csv file.

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

Read only 4 columns in the /opt/logpresso/wp-nginx.csv file.

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

Read only 4 columns from the /opt/logpresso/wp-nginx.csv and assign the rest to the **\_rest** field.

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

Data with a white space between the separator and the column. Compare the results of each query example.

When strict=t, if there is a whitespace between the separator and the column, the double quotes (") are recognized as a character and are not parsed as intended.

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

When strict=f, if the pair of double quotes ("") is matched, only the strings inside the pair of quotes are recognized as columns, so it is parsed as intended.

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

Data without a white space between the separator and the column.

Regardless of the strict value, there is no whitespace between the separator and the column, so it is parsed as intended.

# Download: 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

Data in which double quote characters (") are escaped with a backslash (\).

When strict=t, the command recognizes the escape character (\) as a general character, so if you use " when writing double quotes (") in a column enclosed in a pair of double quotes (" "), it is not parsed as intended.

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

When strict=f, two consecutive double quotes ("") and an escaped double quote (") are parsed as a double quote within the column as intended.

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

### fulltext

Searches for data stored in a table using a full-text index.

#### Syntax

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], ...]

Required Parameter

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

String literal or expression to match records in the table and/or index. If you specify only TABLE, all indexes in the table are retrieved. If you specify the same table or index multiple times, the same search results are duplicated and returned as many time as specified.

EXPR is an expression that expresses the data to be searched and must satisfy the following rules.

* You can use comparison operators. The comparison operators you can use are: ==, !=, >=, >, <, <=
* The string to be searched must be enclosed in a pair of double quotes (" ") and is case insensitive.
* You can provide a combination of the logical operators and, or, not, and a pair of parentheses (( )).
* This searches all tables unless you specify a table.
* If there is a table or index specified more than once, it is returned as many times as it is duplicated.

EXPR recognizes the subquery enclosed in a pair of square brackets([ ]) in the expression. This runs the subquery first before running an index search and then searches all terms to be returned in the results of the subquery. The more search targets returned by the subquery, the slower the index search speed. We recommend that you use the ['fields'](https://docs.logpresso.com/en/query/fields-command) command in a subquery to retrieve only the fields that you really need.

range() and iprange() functions are exclusive to EXPR.

range() function returns the values that fall within a specified range in the table index.

range(MIN\_INT, MAX\_INT)

**MIN\_INT**

Starting index value. This value is include in the range.

**MAX\_INT**

Ending index value. This value is include in the range.

iprange() function returns IP addresses that fall within a specified IPv4 or IPv6 address range in the table index.

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

**START\_IP\_EXPR**

Expression to return the starting IP address of the IP address range. This ip address is included in the range.

**END\_IP\_EXPR**

Expression to return the ending IP address of the IP address range. This ip address is included in the range.

Optional Parameter

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

Time range to search the previous data based on the current time. You can specify time in units of mon (month), w (week), d (day), h (hour), m (minute), and s (second). For example, 10s refers to "the last 10 seconds" based on the current time. This option cannot be used with from and to.

**from=yyyyMMddHHmmss**

Start date and time of the period to search in the form of yyyyMMddHHmmss. The time period for the search includes the specified time point. If you provide only the first part, the remaining digits are recognized as 0. For example, if you provide 20130605, it is recognized as 20130605000000 (June 5, 2013, 00:00:00). This option cannot be used with duration.

**to=yyyyMMddHHmmss**

Specify the date and time to end the search in the form of yyyyMMddHHmmss. The time period for the search does NOT include the specified time point. The input format is the same as from. This option cannot be used with duration.

**limit=INT**

Maximum number of search results (default: unlimited).

**offset=INT**

Number of search results to skip (default: 0).

**order={desc|asc}**

Search order of the index (default: desc)

* desc: Searches from the most recent data to the oldest ones.
* asc: Searches from the oldest data to the most recent ones.

**tt=BOOL**

Boolean option to use the search term tokenizer (default: f).

* t: Searches the string by splitting it into tokenizers for each index.
* f: NOT use the search tokenizer

When you use the tt option, the string wildcard (\*) in EXPR can only be placed at the beginning or end of the string. For example, you can enter "\*asp", "asp\*", and "\*asp\*" in the EXPR, but not "a\*sp". Tokens are separated by the and logical operator to reconstruct the query statement. For example, the query statement fulltext tt=t dst == "10.10.130.235" is reconstructed as fulltext dst == "10" and dst == "10" and dst == "130" and dst == "235".

If you do not use 'duration', 'from' or 'to', all logs are searched.

#### Usage

Search for logs dated June 5, 2013 including "1.2.3.4" from the table.

fulltext from=20130605 to=20130606 "1.2.3.4"

Search for all web logs including "cmdshell" from the iis table.

fulltext "cmdshell" from iis

Search for all web logs including "MSIE" or "Firefox" strings while including the word "game" from the "iis" table.

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

Search for web logs including numbers in the range of 400 to 500 from the iis table.

fulltext range(400, 500) from iis

Search for IP addresses that fall within the range from 192.0.0.1 to 192.0.0.255 in the tables with the name pattern \*.\*SSLVPN.

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

Search for the IP set of the blacklist DB from the iis table.

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

Search for full text for a set of tables where the parser of the table is openssh.

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

Search for full text for the next 20 fidx index data after skipping the first 5 index data.

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

Split the "1.2.3.4" string using the tokenizer for the fidx index and match them against the fidx index from the iis table

fulltext tt=t "1.2.3.4" from iis.fidx

### json

Generates the data source using the JSON string. Typically, this command is used to test the operation of the query command that follows.

#### Syntax

json JSON\_DATA

Required Parameter

**JSON\_DATA**

JSON string enclosed in a pair of double quotes (" "), or an expression that returns a JSON formatted string. If the JSON string has double quotes ("), it must be escaped with a backslash like this: (").

#### Usage

Generate a log with “a” => 8, “b” => "miles” key-value pairs.

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

Generate the “a” => 8, “b” => "miles" log and “a” => 2, “b” => "cats" log.

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

### jsonfile

Loads data in a new line delimited JSON file. The keys are used as field names, and the values are used as field values.

#### Syntax

jsonfile [OPTIONS] PATH

Required Parameter

**PATH**

Path to the file from which you want to load the data. If you use a wildcard (\*) in the file name, you can load all files containing a specific string pattern in the file name at once. To load the file, the Logpresso daemon must have the read permission to the file.

Optional Parameter

**limit=INT**

Maximum number of JSON entries to load (default: unlimited). The command distinguishes records using newline characters (CRLF or LF).

**offset=INT**

Number of records to skip (default: 0).

**overlay=BOOL**

Output option for JSON source data (default: f).

* t: Outputs the parsed data to the fields, and the JSON original data to the **line** field
* f: outputs only the parsed data to the fields

#### Usage

Read the /opt/logpresso/wp-nginx.json file.

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

Read 20 records after skipping the first line of /opt/logpresso/wp-nginx.json file.

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

Read the /opt/logpresso/wp-nginx.json and import the JSON original data to the **line** field.

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

### load

Loads the saved query results.

#### Syntax

load GUID

Required Parameter

**GUID**

GUID assigned to the saved query results

#### Description

There is no command to load GUID information, so there is no case where the user directly executes load command (ENT, STD). When you click an item name stored in **Query > Load > Saved Results**, this command is executed in **Query > Query** and returns the saved query result.

### logger

Outputs the logs collected by the collector in real time for a specified amount of time. Administrative privileges are required to execute this command.

#### Syntax

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

Required Parameter

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

Period of time for which to receive data in real time from the time the query is executed. You can specify the time in units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second). For example, 10s refers to "the next 10 seconds" from the time the query is executed.

**NAMESPACE\LOGGER[, ...]**

Logical path to the log collector that receives data in real time. Use comma (,) as a separator to specify multiple log collectors. Using a wildcard (\*) in LOGGER, you can receive logs from all log collectors containing specific string patterns in the name at once.

You can see the NAMESPACE in the web interface.

* (ENT, STD) You can see it on the **Device** column under **LOG SOURCE > All**.
* (SNR) You can see it in the **local** or on the **Identifier** under **System > Sentry management > Sentry list**.

You can see the name of the LOGGER in the web interface.

* (ENT, STD) You can see it in the **Name** column under **LOG SOURCE > All**.

#### Usage

Receive logs for 10 seconds from local\sample1 and local\sample2 log collectors.

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

### pcapfile

Loads packets from a PCAP file.

#### Syntax

pcapfile FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the file from which you want to load the data. If you use a wildcard (\*) in the file name, you can load all files containing a specific string pattern in the file name at once. To load the file, the Logpresso daemon must have the read permission to the file.

#### Description

This returns the packet binary to the **payload** field. This command is used to load the PCAP file and then pass it to [decodedhcp](https://docs.logpresso.com/en/query/decodedhcp-command), [decodedns](https://docs.logpresso.com/en/query/decodedns-command), [decodehttp](https://docs.logpresso.com/en/query/decodehttp-command), [decodesflow](https://docs.logpresso.com/en/query/decodesflow-command), [pcapdecode](https://docs.logpresso.com/en/query/pcapdecode-command) and the like, which are commands for processing packets.

### remote

Executes queries on the remote federation node. If the node fails to connect, the query fails.

#### Syntax

remote NODE [ SUBQUERY ]

Required Parameter

**NODE**

The name of the target node on which to execute the query. Provide a federation node identifier after confirming its name from the query results of [system nodes](https://docs.logpresso.com/en/query/system-nodes-command).

**[ SUBQUERY ]**

Subquery command enclosed in a pair of square brackets ([ ]).

#### Usage

Execute the system tables query on the n1 node

remote n1 [ system tables ]

### result

Loads the query result being executed in the current session.

#### Syntax

result [offset=INT] QUERY\_ID

Required Parameter

**QUERY\_ID**

ID of the query to load the result. You can see the query ID using [system queries](https://docs.logpresso.com/en/query/system-queries-command).

Optional Parameter

**offset=INT**

Number of records to skip (default: 0).

#### Usage

Load query result of the query #616.

result 616

Load query results of the query #616 after skipping 10 of them.

result offset=10 616

### stream

Either receives output from the specified stream or forwards input data to the specified stream. Administrative privileges are required to execute this command.

#### Syntax

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

Required Parameter

**STREAM, ...**

Name of the stream. Use comma (,) as a separator to specify multiple log streams. You can use a wildcard (\*) in the stream name. If you use a wild card, it receives the output from all streams matching a specific string pattern in stream name at once.

Optional Parameter

**forward=BOOL**

Option to receive output from or forward input to the specified stream specified by STREAM, ... (default: f).

* t: Forwards the input to the stream specified by STREAM, .... Use this option only when you explicitly enable the streaming feature. This option cannot be used when the window option is specified.
* f: Receives the input from the stream specified by STREAM, ....

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

Amount of time to receive data in real time from the time the query is executed. You can specify time in units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second). When the unit is y, only 1y is allowed. For example, 10s refers to "the next 10 seconds" from the time the query was executed. This option cannot be used when forward=t.

#### Usage

Receive the outputs of sample1 and sample2 stream queries in real time for 10 seconds

stream window=10s sample1, sample2

Pass 100 records from the test table to stream sample1 and sample2 as input.

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

### table

Retrieves the data stored in the table.

#### Syntax

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

Required Parameter

**TABLE, ...**

List of tables, separated by a comma (,).

If you put a question mark (?) after the table name, no error occurs even if the table does not exist. For example, the table test query fails if test table does not exist. But if you execute the query table test?, the query returns an empty result without an error.

You can use a wildcard (\*) in the table name. For example, if you execute the query command table sys\_\*, all tables starting with sys\_ are retrieved. Tables that you do not have read permissions for are excluded from the search. After executing a query, you can see the table name in the **\_table** field.

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

Metadata of the table to look up. The meta() function returns tables that match attributes specified by the KEY\_VALUE\_EXPR option.

**KEY\_VALUE\_EXPR**

Comparison expression in the form of "KEY == VALUE" or "KEY != VALUE". You can use it with logical operators such as and or or, or with the not unary negation operator.

**KEY**

Key name of the table metadata

**VALUE**

Value of the table metadata (a wildcard pattern can be used)

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

Table name of the table with names containing specific string patterns. If you don't specify the table, the command attempts to execute a metadata conditional expression on every table.

For ENT and STD, you can specify table metadata in **TABLE > (Select a table) [General]**. The metadata key for parser settings is logparser.

Optional Parameter

If you do not use duration, from or to, all logs are searched.

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

Time range to search the previous data based on the current time. You can specify the time in units of mon (month), w (week), d (day), h (hour), m (minute), and s (second). For example, 10s refers to "the last 10 seconds" based on the time the query is executed. This option cannot be used with from, to or window.

**from=yyyyMMddHHmmss**

Start date and time of the search period in the form of yyyyMMddHHmmss. The time period for the search includes the specified time point. If you provide only the first part, the command recognizes the remaining digits as 0. For example, if you provide 20130605, the command recognizes it as 20130605000000 (June 5, 2013, 00:00:00). This option can be used with to, but cannot be used with duration and window.

**to=yyyyMMddHHmmss**

End date and time of the search period in the form of yyyyMMddHHmmss. The time period for the search does not include the specified time point. The input format is the same as from. This option can be used with from, but cannot be used with duration and window.

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

Period of time for which to receive data in real time from the time the query is executed. You can specify the time in units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second). When the unit is y, only 1y is allowed. For example, 10s refers to "the next 10 seconds" from the time the query was executed. This option cannot be used with duration, from or to.

**limit=INT**

Maximum number of records to load (default: unlimited).

**offset=INT**

Number of records to skip (default: 0).

**order=STR**

Sorting order of the records (default: desc).

* asc: Sorts in ascending order, the oldest at the top.
* desc: Sortes in descending order, the latest records at the top.

#### Usage

Read the last 100 logs from the sys\_cpu\_logs table

table limit=100 sys\_cpu\_logs

Read logs for the last 10 minutes from the sys\_cpu\_logs table.

table duration=10m sys\_cpu\_logs

Read all logs corresponding to the date of June 5, 2013, from the sys\_cpu\_logs table.

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

Read all logs from the sys\_cpu\_logs and sys\_mem\_logs tables in sequence.

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

Read data from tables which have parser metadata with the value of openssh.

table meta("parser==openssh")

### textfile

Loads the data from the text file and returns to the **line** field.

#### Syntax

textfile [OPTIONS] PATH

Required Parameter

**PATH**

Path to the file from which to load the data. If you use a wildcard (\*) in the file name, you can load all files containing a specific string pattern in the file name at once. To load a file, the Logpresso daemon must have the read permission to the file.

If the extension of the file name to be loaded is .gz, Logpresso decompresses and loads the file automatically.

Optional Parameter

You can omit options when creating a query. The omitted option is set to the default value.

**brex="REGEX"**

Regular expression to match the starting row of a record when the record consists of multiple lines. The lines are merged into a single record until a line matching the regular expression appears. If you omit this option, the starting line is recognized based on the newline character (CRLF or LF).

To match the last line of the record, use the erex option.

**cs=CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**df="TIME\_FMT"**

Timestamp format. For example, you may provide yyyy-MM-dd HH:mm:ss.SSS. This option is used with the dp option.

**dp="REGEX"**

Regular expression for date extraction. This extracts the timestamp string by a specified regular expression, then parses it into the timestamp format of the df option to assign the time type value on the **\_time** field. You can use this with the df option.

If you omit this option, the time at which the data is loaded is recorded in the **\_time** field.

**erex="REGEX"**

Regular expression to be used to match the last line of the record if a record consists of multiple lines. The lines are merged into a single record until a line matching the regular expression appears. If you omit this option, the last line is recognized based on the newline character (CRLF or LF).

To match the starting line of the record, use the brex option.

**limit=INT**

Maximum number of records to load (default: unlimited).

**offset=INT**

Number of records to skip (default: 0).

#### Usage

Read the /var/log/secure log file.

textfile /var/log/secure

Read the iis.txt file encoded with euc-kr.

textfile cs=euc-kr iis.txt

Read all the /var/log/syslog.\*.gz file.

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

### xmlfile

Loads data from the xml file. This command can detect XML file encoding by analyzing the byte order mark (BOM). If you use a wildcard (\*) in the file name, you can load all files containing a specific string pattern in the file name at once.

#### Syntax

xmlfile [OPTIONS] PATH

Required Parameter

**FILE\_PATH**

Path to the file from which to load the data. Using a wildcard (\*) in the file name, you can load all files containing a specific string pattern in the file name at once. For example, if you enter report-\*.xml in PATH, you can load all files, such as "report-2022-01-01.xml", "report-2022-01-02.xml" at once. To load a file, the Logpresso daemon must have the read permission to the file.

Optional Parameter

You can omit options when creating a query. The value for the unspecified option is set to the default value.

**cs=CHARSET**

The encoding of the file (default: utf-8). Nothing specified, this command detects XML file encoding by analyzing BOM. For the encoding name, use the preferred MIME name or aliases registered in the following document: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**xpath=EXPR**

The XPath (XML path language) expression to select the XML nodes. For XPath, refer to <https://www.w3.org/TR/xpath-31/>.

#### Usage

Read the report\_kr.xml file encoded in euc-kr.

xmlfile cs=euc-kr report\_kr.xml

Retrieve the title node information from the first book subnode of the bookstore node in the books.xml file.

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

### zipfile

Loads data from a text file compressed with ZIP. You can use wildcards(\*) in the ZIP file path and ZIP entry path.

#### Syntax

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

Required Parameter

**ZIP\_PATH**

Path of the zip file. If you use a wildcard (\*) in the file name, you can retrieve all the files containing a specific string pattern in the file name at once. The Logpresso daemon must have the read permission to the file.

**FILE\_IN\_ZIP**

Zip entry path of the text file from which to load the data. If you use a wildcard (\*) in the file name, you can load all files containing a specific string pattern in the file name at once.

Optional Parameter

**limit=INT**

Maximum number of records to load (default: unlimited).

**offset=INT**

Number of records to skip (default: 0).

#### Usage

Read the iis.txt file among text files compressed in the /opt/logpresso/imported.zip file.

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

Read all txt files compressed in the /opt/logpresso/testdata.zip file.

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

Read all txt files compressed in all ZIP files in the /opt/logpresso path.

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

## Data Processing

### alertmsg

Converts the alert log to the message in the locale (language) specified by the user.

#### Syntax

alertmsg [locale=LOCALE\_CODE]

Optional Parameter

**locale=LOCALE\_CODE**

The locale (language) to be applied to the alert message in the user session (default: en). The languages currently supported are English (en) and Korean (ko).

#### Description

After running the alretmsg, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| category | String | Category of the alert |
| description | String | Description of the alert |
| message | String | Alert message |
| name | String | Alert title |

#### Usage

Convert locale (language) of the alert messages to English.

table sys\_alerts | alertmsg

Convert locale (language) of the alert messages to Korean.

table sys\_alerts | alertmsg locale=ko

### auditmsg

Converts the audit log to the message in the locale (language) specified by the user.

#### Syntax

auditmsg [locale=LOCALE\_CODE]

Optional Parameter

**locale=LOCALE\_CODE**

The locale (language) to be applied to the audit log message in the user session (default: en). The languages currently supported are en and ko.

#### Usage

Convert language of the audit log messages to Korean.

table sys\_audit\_logs | auditmsg locale=ko

### boxplot

Calculates the minimum, maximum, and quartile required to draw a box plot.

#### Syntax

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

Required Parameter

**EXPR**

An expression that is subject to the statistics.

Optional Parameter

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

Grouping fields in the aggregation, separated by a comma (,). This option MUST follow after the EXPR option.

#### Description

The output fields are as follows:

* **count**: Total number of records belonging to GRP\_FIELD. Without GRP\_FIELD, count the total number of records.
* **GRP\_FIELD**: Value of group field entered in the by clause
* **iqr1**: First quartile for each group. The median value of the lowest 50% based on the median value, the lowest 25% of the total data
* **iqr2**: Second quartile for each group (median value). The most central value when the data is sorted in order
* **iqr3**: Third quartile for each group. The median value of the highest 50% based on the median value, the highest 25% of the total data
* **max**: Maximum value for each group
* **min**: Minimum value for each group

#### Usage

Five-number summary for overall CPU load.

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

Five-number summary for CPU load by date.

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

### bypass

Bypasses all input values as they are. This command is used to create a field index for all input fields or to pass all results in stream queries as they are.

#### Syntax

bypass

### cube

Automatically performs all possible aggregations of the specified fields. The command returns an aggregated result containing all the possible combinations for the selected fields if multiple fields are specified in the by clause.

#### Syntax

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

Required Parameter

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

Pair of a [aggretgational function](https://docs.logpresso.com/en/query/section-aggregate-functions) (AGGR\_FUNC) and optional alias (ALIAS) to be displayed as a field name. If no alias specified, the command uses the function name, such as avg(), as the field name. It is recommended that you specify an ALIAS.

Optional Parameter

**label=FIELD**

Label to be given to the aggregate value (default: null).

**parallel=BOOL**

Option to enable parallel processing (default: f).

* t: Enables processing the query in parallel. The processing speed increases but the order of data is not guaranteed. Avoid using this option in query commands where the order of the data matters.
* f: Disables processing the query in parallel.

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

Grouping fields with by directive, separated by a comma(,). This option MUST follow after the AGGR\_FUNC [as ALIAS] option.

#### Usage

Retrieve records from web server log table web\_access to calculate subtotals and grand totals of the count for all permutations of the **date** field and the **status** field.

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

Calculate the aggregation of the count and size for all combinations generated by the values of the action and status fields.

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

### curvefit

Runs a linear regression analysis for the input using the Least Mean Square (LMS).

#### Syntax

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

Required Parameter

**INDEPENDENT\_FIELD**

Field as the independent variable. The value of the independent variable must be numeric.

**DEPENDENT\_FIELD**

Field as the dependent variable. The value of the dependent variable must be numeric.

Optional Parameter

**degree=INT**

Degree of the polynomial function that approximates the input value (default: 3).

#### Description

This runs a linear regression analysis using the least mean square for up to 10,000 input record values. This assigns the independent variable field value to the **\_x** field and the predicted value to the **\_p** field. It ignores records after 10,000.

#### Usage

Approximate the CPU usage rate for the past 1 hour with a tenth polynomial function.

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

Decodes the DHCP packets.

#### Syntax

decodedhcp

#### Description

The output fields are as follows:

* **client\_ip**: IP address of the DHCP client. The address of any client which an IP address is not assigned is 0.0.0.0.
* **client\_mac**: MAC address of the DHCP client.
* **fingerprint**: DHCP fingerprint if present. See the list of DHCP and BOOTP parameters managed by IANA for the meaning of each number: <https://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml>
* **gateway\_ip**: Gateway IP address specified by the DHCP relay agent when communicating with the DHCP server on behalf of the client. Typically, the IP address at which the DHCP server can communicate with the DHCP agent.
* **next\_server\_ip**: IP address of the secondary DHCP server
* **options**: Field that displays the DHCP option numbers as an array
* **tx\_id**: DHCP transaction identifier
* **your\_ip**: IP address assigned by the DHCP server to the client

#### Usage

Decode DHCP packets from a pcap file.

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

### decodedns

Decodes the DNS request and response packets.

#### Syntax

decodedns

#### Description

The output fields are as follows:

* **additionals**: Other additional records (array)
* **answers**: Answer resource records from the DNS server (array)
* **authorities**: Information of authoritative DNS servers (array)
* **bytes**: Size (integer) of the DNS payload
* **direction**: Direction of the transaction (string)
* c->s: Request from the client
* s->c: Response from the server
* **domain**: Domain address to be queried (string)
* **dst\_ip**: Destination IP address of the DNS transaction (IP address)
* **dst\_port**: Destination port of the DNS transaction (integer)
* **flags**: DNS header flag. Refer to: [https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-12](https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml)
* **ip**: IP address connected to the domain address (IP address)
* **src\_ip**: Source IP address of the DNS transaction (IP address)
* **src\_port**: Source port of the DNS transaction (integer)
* **status**: Query result. If there is an error, an error message is displayed.
* FORMAT\_ERROR: The DNS server could not interpret the request.
* NAME\_ERROR: The domain name in the request does not exist.
* NO\_ERROR: No error.
* NOT\_IMPLEMENTED: The DNS server does not support the type of request.
* REFUSED: The DNS server refused the request.
* SERVER\_FAILURE: The DNS server was unable to process the request due to an internal server error.
* **txid**: DNS transaction ID (hexadecimal string)
* **type**: DNS record type (strings such as A, AAAA, CNAME, MX, NS, PTR, SOA, SRV, TXT). Refer to: [https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4](https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml)

#### Usage

Decode DNS packets from a pcap file.

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

### decodehttp

Decodes the HTTP header in the packets.

#### Syntax

decodehttp

#### Description

The output fields are as follows:

* **dst\_ip**: Destination IP address (IP address)
* **dst\_port**: Destination port (integer)
* **host**: Name of the web server in the FQDN (Fully Qualified Domain Name) format (string)
* **method**: HTTP method (string)
* **path**: Resource path (string). Typically, the URI (Uniform Resource Identifier). Refer to: <https://tools.ietf.org/html/rfc3986>
* **rcvd**: Data received by the client from the server (bytes)
* **req\_time1**: Time the HTTP request is initiated (Epoch timestamp)
* **req\_time2**: Time the HTTP request is completed (Epoch timestamp)
* **res\_time1**: Time the HTTP response is initiated (Epoch timestamp)
* **res\_time2**: Time the HTTP response is completed (Epoch timestamp)
* **sent**: Data sent by the client to the server (bytes)
* **src\_ip**: Source IP address (IP address type)
* **src\_port**: Source port (integer)
* **status**: HTTP response code of the server (ingeter). Refer to: <https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml>

#### Usage

Decode HTTP packets from a pcap file.

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

### decodesflow

Decodes the sFlow packets.

#### Syntax

decodesflow

#### Description

The output fields are as follows:

* **agent\_addr**: IP address of sFlow agent
* **agent\_id**: Agent identifier
* **counters**: When **sample\_type** is counters, the following map-type information is output.
* **admin\_status**: Whether the administrator port is enabled (true, false)
* **if\_direction**(0: unknown, 1: Full-duplex, 2: Half-duplex, 3: Receive, 4: Send)
* **if\_index**: Interface identifier
* **if\_speed**: Connection link speed in bps
* **if\_type**: Ethernet is 6, Refer to the interface type standard number assigned by IANA: "IANAifType ::= TEXTUAL-CONVENTION" and "SYNTAX INTEGER" sections in <https://ietf.org/assignments/ianaiftype-mib/ianaiftype-mib>
* **in\_bcast\_pkts**: Number of broadcast packets received
* **in\_discards**: Number of discarded packets received
* **in\_errors**: Number of packets with errors received
* **in\_mcast\_pkts**: Number of multicast packets received
* **in\_octets**: Total bytes received (bytes)
* **in\_ucast\_pkts**: Number of unicast packets received
* **in\_unknown\_protos**: Number of packets whose protocols are unknown among the received packets
* **oper\_status**: Whether the actual link is enabled (true, false)
* **out\_bcast\_pkts**: Number of broadcast packets sent
* **out\_discards**: Number of discarded packets to be sent
* **out\_errors**: Number of packets with errors among the packets to be sent
* **out\_mcast\_pkts**: Number of multicast packets sent
* **out\_octets**: Bytes sent
* **out\_ucast\_pkts**: Number of unicast packets sent
* **promisc\_mode**: Whether promiscuous mode is enabled (true, false)
* **drops**: Number of packets lost due to lack of performance. This outputs information when **sample\_type** is flow.
* **dst\_ip**: Destination IP address. Typically, the address of the sFlow collenction server.
* **dst\_port**: Destination port number
* **flow**: A packet randomly sampled according to **sampling\_rate**
* **flow\_seq**: Increases by 1 each time a sample created for each **src\_id**
* **flows**: Outputs the flow data information sampled when **sample\_type** is flow.
* **counters**: Inerface counters that transmit at regular intervals
* **frame\_length**: Length in byte of the packet before sampling
* **header**: Ethernet header octet byte stream
* **protocol**: Connection layer protocol (e.g., ethernet)
* **stripped**: Number of octets removed from the packet before extracting the data link layer header octets
* **input\_if\_index**: Outputs the interface identifier information received when **sample\_type** is flow
* **output\_if\_index**: Outputs the interface identifier information sent when **sample\_type** is flow
* **protocol**: Transport layer protocol. udp only.
* **sample\_pool**: Number of sources to be sampled. This outputs information when **sample\_type** is flow.
* **sample\_type**: Type of sampling
* **sampling\_rate**: Sampling rate. One from the specified number of packets is extracted. This outputs information when **sample\_type** is flow.
* **src\_id**: Interface identification number
* **src\_id\_type**: Interface type defined in RFC 2613 (0: ifIndex, 1: smonVlanDataSource, 2: entPhysicalEntry). Refer to 3.1.1 DataSource Objects in RFC 2613: <https://tools.ietf.org/html/rfc2613>
* **src\_ip**: Source IP address. Typically, the address of the sFlow agent.
* **src\_port**: Source port number
* **uptime**: Uptime after the agetn starts
* **ver**: sflow version. Only version 5 supported.

#### Usage

Decode sFlow packets from sflow.pcap file.

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

### eval

Evaluates the expression on the right and then assigns a new field or replaces the old field value. You can enter any combination of expressions that can be evaluated to value on the right side.

#### Syntax

eval FIELD=EXPR, ...

Required Parameter

**FIELD=EXPR, ...**

Statement to assign the value obtained by evaluating the expression to a new or old field. You can enter any combination of expressions that can be evaluated to value on the right side. Use commas (,) as a separators to specify multiple pairs of assignments. If there are multiple pairs of FIELD=EXPR, the command evaluates them in order from the left.

#### Usage

Evaluate the [int("100")](https://docs.logpresso.com/en/query/int-function) function to convert the string to an integer and then assign it to the **num** field.

json "{}" | eval num = int("100")

Call the [typeof()](https://docs.logpresso.com/en/query/typeof-function) to assign the type of a specific value to the **type1**, **type2** field.

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

Sum up the field values

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

Assign the results of executing the [`concat("hello", ", world")`](https://docs.logpresso.com/en/query/concat-function) function to the **msg** field

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

### evalc

Evaluates the expression on the right, and then assigns a new query parameter or replaces the value of an old query parameter.

#### Syntax

evalc VAR\_NAME=EXPR

Required Parameter

**VAR=EXPR**

Query parameter. Assign the parameter name on the left side and any combination of expressions that can be evaluated as a value to the right. The value obtained by evaluating the expression is assigned to the query parameter. Unlike the [set](https://docs.logpresso.com/en/query/set-command) command, it evaluates all the data at the time the query is executed.

#### Usage

Set the alert query parameter to true if count exceeds the threshold

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

All input data are assigned after being evaluated with the right side expression, so if its count does not exceed the threshold, use the [$()](https://docs.logpresso.com/en/query/$-function) function to re-assign the existing variable value as it is.

### explode

Expands the values of an array field into separated rows. This command is generally used to convert an array (horizontal) into a column (vertical). It preserves the input row if the specified field does not exist, is not an array, or is null.

#### Syntax

explode FIELD

Required Parameter

**FIELD**

Name of the field containing the array.

#### Usage

Calculate IP statistics

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/en/query/json-command): Generates the data source using "{line: '10.0.0.1 10.0.0.2'},{line:'10.0.0.2 10.0.0.3'}".

[eval](https://docs.logpresso.com/en/query/eval-command): Assigns the value in form of array that evaluated the [`split(line, " ")`](https://docs.logpresso.com/en/query/split-function) function to the new **ip** field.

[explode](https://docs.logpresso.com/en/query/explode-command): Explodes the array in the **ip** field in unit of array component, expanding the two rows into four.

[stats](https://docs.logpresso.com/en/query/stats-command): Calculates the [count](https://docs.logpresso.com/en/query/count-aggregate-function) of the number of rows in **ip** field.

### fields

Either selects only specific fields or selectively excludes only specific fields.

#### Syntax

fields [-] FIELD, ...

Required Parameter

**FIELD, ...**

List of fields. Use comma (,) as a separator.

Optional Parameter

**-**

Exclude the fields after the minus symbol (-), these fields are excluded from the output. If you omit this sign, only those fields are selected.

#### Usage

Select only the **src\_ip** and **action** fields.

fields src\_ip, action

Remove only the **line** field

fields – line

### flowsearch

Loads a flow rule consisting of the subnet of the IP network, ports, and protocol conditions defined by a subquery, compares them with input records, and assigns all searched flow identifiers as arrays in the **\_flow** field.

#### Syntax

flowsearch [ SUBQUERY ]

Required Parameter

**[ SUBQUERY ]**

Subquery to define the flow rules, enclosed in a pair of square brackets ([ ]).

#### Description

You can load the flow rule from any location, including a file, table, remote RDBMS, and the like, and the field configuration and type must match to be recognized as a valid rule. The number of flow rules to be applied to a subquery cannot exceed 10,000. From the 10,001st rule, they are ignored.

If the subquery fails, the cause of the error is output in the \_flowsearch\_error field. If you add an exception handling command that checks for the existence of the \_flowsearch\_error field value after the flowsearch command, unintended errors or malfunctions can be prevented.

**Definition of input fields**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required | Description |
| src\_ip | IP address | Yes | Source IP address |
| src\_port | Integer | No (null allowed) | Source port number |
| dst\_ip | IP address | Yes | Destination IP address |
| dst\_port | Integer | No (null allowed) | Destination port number |
| protocol | String | No (null allowed) | Protocol |

If the field type in the input record does not match, or if required fields are missing, the command outputs the record as it is without checking the flow rule.

**Definition of flow rule fields**

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required | Description |
| src\_ip | IP address | Yes | Source IP address |
| src\_cidr | Integer | Yes | Source netmask (0-32) |
| src\_port | Integer | No (null allowed) | Source port number (0-65535) |
| dst\_ip | IP address | Yes | Destination IP address |
| dst\_cidr | Integer | Yes | Destination netmask (0-32) |
| dst\_port | Integer | No (null allowed) | Destination port number (0-65535) |
| protocol | String | No (null allowed) | Protocol (TCP, UDP, ICMP, ...) |
| flow | Arbitrary | Yes | Flow identifier |

Each time a record is provided into the flowsearch command, it compares the 5-tuple value input with the flow rule and assigns the matching flow identifier as a list in the **\_flow** field.

Both the **ip** and **cidr** of the flow rule are required fields, but if the **src\_ip** of the rule is 0.0.0.0 and the **src\_cidr** is 0, it is true for all source IP addresses. So set 0.0.0.0/0 for the rule to allow all values for the source or destination.

For example, for the flow rule below, if the input record is src\_ip=106.75.11.63, src\_port=57776, dst\_ip=106.246.20.67, dst\_port=80, and protocol=TCP, then it matches with the flow 2, so the **\_flow=["flow2"]** field is added to the output record.

**Examples of flow rules**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

#### Usage

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

Runs a script written in Groovy.

#### Syntax

groovy CLASS\_NAME

Required Parameter

**CLASS\_NAME**

Name of the class you want to execute.

#### Description

Groovy is a dynamic object-oriented language developed with influence from languages such as Python and Ruby and runs on the JVM. The script file must meet the following constraints to be executed:

* You need to specify the script file name in the following format: CLASS\_NAME.groovy
* The script files SHOULD be in **data/araqne-logdb-groovy/query\_scripts** in the directory where Logpresso is installed.
* You need to import and use the package provided by Logpresso. Use the following packages as needed.
* groovy.transform.CompileStatic
* org.araqne.logdb.groovy.GroovyQueryScript (required)
* org.araqne.logdb.QueryStopReason
* org.araqne.logdb.Row (required)
* org.araqne.logdb.RowBatch
* org.araqne.logdb.RowPipe

To improve the performance of Groovy scripts, refer to the following:

* Avoid using string processing methods as much as possible. As the number of string objects increases, garbage collection occurs more frequently in the JVM.
* Avoid using the split() and tokenize() methods as much as possible.
* split() is very slow because it uses a regular expression internally.
* Use indexOf() or substring() instead. Although the code is longer, it provides better processing performance.
* Avoid using Pattern.compile(). Reusing the Matcher instance by calling Matcher.reset() provides better performance.
* DO NOT use exceptions for the normal flow of control
* If exceptions occur frequently, processing performance becomes significantly slower.
* If possible, handle possible error cases through conditional testing.

#### Usage

Save the following script as **ToAscii.groovy** in **data/araqne-logdb-groovy/query\_scripts** in the directory where Logpresso is installed.

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) }}

This script encodes the 32nd to 127th characters in ASCII format among the binary values decoded in the PCAP file and assigned on the **payload** field.

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

### limit

Constrain the number of rows returned and then cancels the query.

#### Syntax

limit [INT\_OFFSET] INT\_MAX

Required Parameter

**INT\_MAX**

Maximum number of records from the query results. The query is canceled when the specified number of query inputs is reached. Note that some commands may not work as intended if the query is canceled.

Optional Parameter

**INT\_OFFSET**

Number of query results to skip (default: 0).

#### Usage

Return the first 5 rows and then cancel the query.

table sys\_cpu\_logs | limit 5

The query command above has the same results as the following:

table limit=5 sys\_cpu\_logs

Ignore the first row and return the following 2 rows and then cancel the query.

table sys\_cpu\_logs | limit 1 2

The query command above has the same results as the following:

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

### mpsearch

Matches thousands or more keyword patterns simultaneously. When the pattern specified in the subquery is detected in the field subject to be matched, it assigns a list of all detected patterns to the **\_mp\_result** field.

#### Syntax

mpsearch FIELD [ SUBQUERY ]

Required Parameter

**FIELD**

Name of field to match multi-patterns.

**[ SUBQUERY ]**

Subquery to load the list of keyword patterns to be searched, enclosed in a pair of square brackets ([ ]).

#### Description

The output of the subquery must contain the **expr**, **expr2**, and **rule** string fields.

* **expr** (required): Write it by combining strings into a boolean expression. The command quickly detects the corresponding string from the values of the strings of the field to be scanned and then checks whether it matches the expression.
* **expr2** (optional): If the boolean expression in the **expr** field is true, this selectively provides an opportunity to match further using the values in other fields.
* **rule** (required): Provide the pattern identifier or name.

Examples of patterns are as follows:

**Examples of pattern**

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

If only the xp\_cmdshell pattern is detected, the value of the **\_mp\_result** field is as follows:

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

#### Usage

Match multi-patterns to the signature field by loading the pattern list from the external DB.

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

### order

Sorts the specific fields to be printed out in the specified order, and displays the remaining fields in lexicographical order.

#### Syntax

order FIELD, ...

Required Parameter

**FIELD, ...**

Names of fields to be ordered in order, separated by a comma (,). The command sorts the fields that are not listed here in lexicographical order.

#### Usage

Define the field output order of the sys\_cpu\_logs table as **kernel**, **idle**, **user**, **\_time**, **\_table**, and **\_id**.

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

Define the field output order of the sys\_cpu\_logs table as **idle**, **kernel** and then the rest in lexicographical order

table sys\_cpu\_logs | order idle, kernel

### parallel

Processes input data in parallel using a subquery, and then combines the results of subquery and passes them along.

#### Syntax

parallel core=INT [ SUBQUERY ]

Required Parameter

**[ SUBQUERY ]**

Subquery to be processed in parallel, enclosed in a pair of square brackets ([ ]).

**core=INT**

Number of logical cores in the CPU to be used for parallel processing of a subquery

#### Usage

First, create a data table creation for testing.

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

Execute the subquery in parallel.

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

### parse

Parses data using a predefined parser or user-defined rule.

#### Syntax

parse [overlay=BOOL] PARSERparse [OPTIONS] PARSING\_RULE, ...

Required Parameter

**PARSER**

Predefined parser name. You can see the names of the parsers in the web console.

* (STD, ENT) Check the **Name** in **SYSTEM CONFIG > PARSER/TRANSFORMER > Parser**
* (MAE, SNR) Check the **Parser code** in **Logger > Parser**, **Normalizer**.

This option cannot be used with the field=TARGET\_FIELD option.

**PARSING\_RULE, ...**

User-defined rules, separated by a comma (,). The syntax of a parsing rule is "START\_ANCHOR\*STOP\_ANCHOR" as FIELD\_NAME:

* START\_ANCHOR\*STOP\_ANCHOR: Parse anchors
* as FIELD\_NAME: Label given as the name of the field

The parse command parses strings according to specified start and stop anchors, and then labels them with FIELD\_NAME.

Optional Parameter

**overlay=BOOL**

Option to control data-overwrite (default: f).

* t: Outputs the parsed data and the original data.
* f: Outputs the parsed data only.

**field=TARGET\_FIELD**

Field to be parsed in the input data stream (default: line). This option cannot be used with the PARSER argument.

#### Usage

Parse the logs stored in the ssh\_log table using the openssh parser.

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

Extract the field by specifying the start and end text from the log (using the following contents by saving it as "sample.txt").

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

You can extract the **session**, **proto**, **policy**, **rule**, and **end\_time** fields from the above source using the command below.

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

Parses the comma-separated values (CSV) string or tab-separated values (TSV) string.

#### Syntax

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

Optional Parameter

**field=TARGET\_FIELD**

Field to be parsed in the input data stream (default: line).

**overlay=BOOL**

Option to control data-overwrite (default: f).

* t: Overwrites the input data with the parsed data.
* f: Outputs the parsed data only.

**strict=BOOL**

Option to conform RFC 4180 (default: f).

* t: Parses the CSV file by strictly applying the RFC 4180: <https://tools.ietf.org/html/rfc4180>. It is the same as when you open the CSV file with Excel. This option cannot be used when tab=t.
* f: Flexibly parses the CSV file.

**tab=BOOL**

Option to use a tab character as a separator (default: f).

* t: Uses a tab character as a separator. This is useful for processing tab-separated values (TSV) files.
* f: Uses a comma (,) as a separator.

Target Object

Target Objects

**FIELD, ...**

Field names for the parsed fields, separated by a comma (,). This option must be placed last in the expression. If names are not provided, default names are used: column0, column1, ..., columnN in order.

#### Usage

Parse a comma-separated text

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

Parse a comma-separated text to give names name1 and name2 to fields in order from the front.

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

### parsejson

Parses a JSON string.

#### Syntax

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

Optional Parameter

**field=TARGET\_FIELD**

Field to be parsed in the input data stream (default: line).

**overlay=BOOL**

Option to control data-overwrite (default: f).

* t: Outputs both the parsed data and the original data.
* f: Outputs the parsed data only.

#### Usage

Parse the JSON text of the **line** field.

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

### parsekv

Parses a string consisting of key-value pairs.

#### Syntax

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

Optional Parameter

**field=TARGET\_FIELD**

Field to be parsed in the input data stream (default: line).

**kvdelim="CHAR"**

Character to use as the key-value delimiter (default: =).

**overlay=BOOL**

Option to control data-overwrite (default: f).

* t: Outputs both the parsed data and the original data.
* f: Outputs the parsed data only.

**pairdelim="CHAR"**

Delimiter that separates the key-value pairs(default: space).

#### Usage

Convert the JSON string in the **line** field into a key-value pair.

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

### parsemap

Extracts all the keys from the map to the field.

#### Syntax

parsemap [overlay=BOOL] field=TARGET\_FIELD

Required Parameter

**field=TARGET\_FIELD**

Field to be parsed in the input data stream. The field's value must be a map type. If the value of the target field is null or is not a map type, the original data is passed as is.

Optional Parameter

**overlay=BOOL**

Option to control data-overwrite (default: f).

* t: Outputs both the parsed data and the original data.
* f: Outputs the parsed data only.

#### Usage

Extract all key-value pairs as fields from map data in **complex** fields

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

### parsexml

Parses the XML document into a set of complex objects.

#### Syntax

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

Optional Parameter

**field=TARGET\_FIELD**

Field to be parsed in the input data stream (default: line).

**overlay=BOOL**

Option to control data-overwrite (default: f).

* t: Outputs both the parsed data and the original data.
* f: Outputs the parsed data only.

#### Usage

Extract XML nodes that belong to the root XML element into the field.

* If the XML node contains only a text node, the command uses the element tag as the name of the field and assigns the text value of the text node to the value of the field.
* If the XML node has attributes, the command converts the name-value pair of each XML attribute to the key-value pair of the map, and converts the text value of the text node of the XML element to the value of the **\_text** field.

For example, if you parse the XML in the form of <doc><id>sample</id></doc>, the value of the sample string is assigned to the **id** field.

If the form of XML is , two key-value pairs of locale=en and \_text=Logpresso are assigned to the **name** field like {"locale":"en","\_text":"Logpresso"}. You can easily extract fields from the map inside a complex object by combining the [parsemap](https://docs.logpresso.com/en/query/parsemap-command) command.

json| parsexml| parsemap field=name overlay=t

### pcapdecode

Decodes the packets and outputs layer 4 metadata fields.

#### Syntax

pcapdecode

#### Description

After running the pcapdecode command, the output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| src\_mac | String | Source MAC address |
| dst\_mac | String | Destination MAC address |
| vlan\_id | Integer | VLAN ID |
| protocol | String | esp, icmp, tcp, or udp |
| src\_ip | IP address | Source IP address |
| src\_port | Integer | Source port |
| dst\_ip | IP address | Destination IP address |
| dst\_port | Integer | Destination port |
| payload | Binary | Packet payload |

#### Usage

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

### pcapreplay

Replays previously captured network traffic. Administrative privileges are required to execute this command.

#### Syntax

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

Required Parameter

**device="DEVICE\_NAME"**

Name of the network device to replay packets among the devices identified by the [system pcapdevices](https://docs.logpresso.com/en/query/system-pcapdevices-command) command. To specify the interface, specify the name of the device identified as **name**.

Optional Parameter

**pps=INT**

Packet replay speed in packets/sec.

#### Description

For this command to work, a driver such as libpcap or winpcap must be installed, and the Logpresso process must be able to use RAW I/O for the network interface with administrative privileges.

You can use this command by applying it in a way that transmits incoming traffic to the monitor port of the IPS or network traffic analysis device.

To replay the packet data stored in the table in chronological order, you need to apply the order=asc option to the [table](https://docs.logpresso.com/en/query/table-command) command to sort the packets in the original chronological order.

#### Usage

Transmit traffic from the PCAP device enp0s3 at a speed of 1,302,083 pps (about 1 Gbps) after reading the **payload** field from the record stored in the tapped\_traffic table for the last 5 minutes.

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

### pivot

Executes one or more aggregation on each row or column group.

#### Syntax

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

Required Parameter

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

Pairs of a [group function](https://docs.logpresso.com/en/query/section-aggregate-functions) (AGGR\_FUNC) and optional alias (ALIAS) as a field name, separated by a comma (,). It is recommended that you specify an ALIAS. If no alias provided, the function name is labeled as the field name, such as count() and sum(sent\_pkts).

Optional Requirements

**parallel=BOOL**

Option to enable parallel processing (default: f).

* t: Enables processing the query in parallel. The processing speed increases but the order of data is not guaranteed. Avoid using this option in query commands where the order of the data matters.
* f: Disables processing the query in parallel.

**by|rows GRP\_FIELD, ...**

Grouping fields with by or rows directive, separated by a comma(,).

**for|cols GRP\_FIELD, ...**

Grouping fields with for or cols directive, separated by a comma(,).

If the 'by|rows' or 'for|cols' clause is not specified, the entire data that comes from the previous query command is aggregated into one group. There is a side effect that is sorted out based on the group field.

#### Usage

Return the number of rows.

pivot count

Return the number of rows by each src\_ip value.

pivot count by src\_ip

Count the number of rows for each **protocol** field value (e.g., TCP, UDP, ICMP and the like) for the **src\_ip** and **dst\_ip** fields.

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

Count the number of rows ([count()](https://docs.logpresso.com/en/query/count-aggregate-function)) and traffic ([sum(bytes)](https://docs.logpresso.com/en/query/sum-aggregate-function)) for each **protocol** field value (e.g., TCP, UDP, and ICMP) for the **src\_ip** and **dst\_ip** fields.

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

### prev

Adds the field value of the previous record to the field (e.g., **prev\_count**) with the prefix **prev\_** to the next input record. This command is often used to capture the change in a data series.

#### Syntax

prev INPUT\_FIELD, ...

Required Parameter

**INPUT\_FIELD, ...**

Fields to track its previous value, separated by a comma (,). The command saves the value of the previous record of the specified fields to the field with the prefix **prev\_**.

#### Usage

Calculate the amount of change in GC count every minute.

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

Retrieve GC logs with a GC occurrence interval of 10 seconds or less.

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

### rename

Changes the source field name to the specified field name.

#### Syntax

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

Required Parameter

**FIELD**

Name of the source field

**as NEW\_NAME**

New name to the field with as directive

#### Usage

Rename src\_ip field to Source

rename src\_ip as Source

### repeat

Repeats the result as many times as specified. It does not guarantee the iteration order. It may be repeated for each row, or it may be repeated for a set of records.

#### Syntax

repeat count=INT

Required Parameter

**count=INT**

Number of times to repeatedly return the result

#### Usage

Return the latest 10 CPU usage rates 3 times

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

Generate 100 arbitrary data points.

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

### rex

Extracts the fields from the specified field using a regular expression.

#### Syntax

rex field=FIELD "REGEX"

Required Parameter

**field=FIELD**

Target field from which to extract the string using a regular expression.

**"REGEX"**

Extended regular expression to give the field name. If you specify the group in the form of (?<field>) in the regular expression, the command extracts the string matched to the group to the **field** field.

#### Usage

Look up a file path starting with GET /game/flash/ or POST /game/flash from the **line** field and then return the matched ones to the **filename** field.

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

Extract the string in the timestamp pattern from the **line** field and assign it to the **timestamp** field.

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

Extract the strings from the **line** field and assign them to the **url** and **querystring** fields.

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

### rollup

Calculates multiple levels of subtotals across a group of fields along with the grand total. This command creates subtotals that roll up from the most detailed level to a grand total, following a grouping field specified in the by clause.

#### Syntax

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

Required Parameter

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

Pairs of a [aggregate function](https://docs.logpresso.com/en/query/section-aggregate-functions) (AGGR\_FUNC) and optional alias (ALIAS) as a field name, separated by a comma (,). If no alias provided, the function name is labeled as the field name, such as count() and sum(sent\_pkts). It is recommended that you specify an ALIAS.

Optional Parameter

**label=VALUE**

Label given to the aggregate value (default: null).

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

Grouping fields with by directive, separated by a comma(,).

#### Usage

Calculate the subtotal for each **action** field, and grand total.

rollup count by action

Calculate sub-totals and grand totals (the label is displayed as "TOTAL") of the count and size for the action and status fields.

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

### search

Filters only the input data that match the specified expression.

#### Syntax

search [limit=INT] EXPR

Required Parameter

**EXPR**

Filter conditions in the form of an expression. For example, you can enter a comparison expression in the form "KEY == VALUE" or "KEY != VALUE" or a boolean expression. You can concatenate conditional expressions using logical operators such as and and or.

Only if the EXPR is true, the data can be passed to the next query command.

Optional Parameter

**limit=INT**

Maximum number of records to return (default: unlimited).

#### Usage

Filter a log containing the game string literal in the **line** field (supports wildcards).

search line == "\*game\*"

Filter a log where the status code is not 200.

search status != 200

Search for the case where src\_ip is 1.2.3.4 and dst\_port is 22.

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

### serial

Serializes the input in tuple units to run commands whose order is important and passes the subquery results by concatenating them.

#### Syntax

serial [ SUBQUERY ]

Required Parameter

**SUBQUERY**

Subquery that can process the stream, enclosed in a pair of square brackets ([ ]).

#### Usage

Apply the CEP function ([evtctxgetvar()](https://docs.logpresso.com/en/query/evtctxgetvar-function), [evtctxsetvar()](https://docs.logpresso.com/en/query/evtctxsetvar-function)) by row

table iis | # Serializing CEP operations | 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

Extracts a signature consisting of a set of special characters from the **line** field. This command is typically used to extract log samples by pattern type before developing a parser.

#### Syntax

signature

#### Usage

Extract the first sample log for each signature.

signature | stats first(line) by signature

### sort

Sorts input data based on the specified fields.

#### Syntax

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

Required Parameter

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

Sort fields and order in which to sort the input data set, separated by a comma (,). The default order of the field is ascending order. To sort in descending order, prefix the name with a minus symbol(–).

Optional Parameter

**limit=INT**

Number of records to return from the sorted results (default: unlimited).

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

After partitioning based on the value of the partition field, you can sort records by partition. If you use the limit option with the by clause, the command returns top n records from each partition.

#### Usage

Return the top 10 records in descending order based on the **count** field.

sort limit=10 -count

Return the top 10 records in descending order based on the **bytes** and **pkts** fields.

sort limt=10 -bytes, -pkts

Return the top 10 records in descending order based on the **bytes** and **pkts** fields for each **src** and **dst**.

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

### stats

Executes one or more aggregation on each row group.

#### Syntax

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

Required Parameter

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

Name of a [group function](https://docs.logpresso.com/en/query/section-aggregate-functions) (AGGR\_FUNC) and optional alias (ALIAS) as a field name. If no alias is provided, the command uses the function name as the field name, such as **count()** and **sum(sent\_pkts)**. It is recommended that you specify an ALIAS.

Optional Parameter

**parallel=BOOL**

Option to enable parallel processing (default: f).

* t: Enables processing the query in parallel. The processing speed increases but the order of data is not guaranteed. Avoid using this option in query commands where the order of the data matters.
* f: Disables processing the query in parallel.

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

Grouping fields with by directive, separated by a comma(,).

#### Usage

Return the number of rows.

stats count

Return the number of rows for each value of **src\_ip** field.

stats count by src\_ip

Return the number of rows by grouping it into pairs of **src\_ip** and **dst\_ip** fields.

stats count by src\_ip, dst\_ip

Calculate [sum(bytes)](https://docs.logpresso.com/en/query/sum-aggregate-function) and [count()](https://docs.logpresso.com/en/query/count-aggregate-function) by grouping them into pairs of **src\_ip** and **dst\_ip** fields.

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

### timechart

Calculates the result of the aggregate function for every specified time span. If you specify a group field using the by clause, the field is created as a group field value and the statistics for each field are calculated.

#### Syntax

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

Required Parameter

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

Time span to create a row group based on the \_time field. You can specify time in units of s (second), m (minute), h (hour), d (day), w (week), mon (month), and y (year). For example, 10m is a unit of 10 minutes. If you use a unit of month mon, you can specify only 1mon, 2mon, 3mon, 4mon, and 6mon among the divisors of 12 to enable aggregation. That is, 3mon is allowed, but 5mon is NOT. You need to use 1y instead of 12mon. When the unit is y, only 1y is allowed.

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

Name of a [aggregate function](https://docs.logpresso.com/en/query/section-aggregate-functions) (AGGR\_FUNC) and optional alias (ALIAS) with the as directive as a field name. If no alias is provided, the command uses the function name as the field name, such as count() and sum(sent\_pkts). It is recommended that you specify an ALIAS.

Optional Parameter

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

Grouping fields with by directive, separated by a comma(,).

#### Usage

Count total log occurrences for every 10 minutes.

timechart span=10m count

Show the trend of changes in bytes for every 1 minute.

timechart span=1m sum(bytes)

Count the number of log occurrences for every 1 hour for each destination port.

timechart span=1h count by dst\_port

### tojson

Converts the given field values to the JSON string.

#### Syntax

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

Optional Parameter

**output=TARGET\_FIELD**

Field to store the converted JSON string (default: \_json).

**FIELD, ...**

Fields to be converted into the JSON string, separated by a comma (,) (default: all fields).

#### Usage

Convert the all fields into the json string and assign it to the **result** field.

tojson output=result

Convert the **\_time** and **line** fields into the json format and assign it to the **jsonlog** field.

tojson output=jsonlog \_time, line

## Data Mapping

### lookup

Looks up values in the specified lookup table, and assign them to fields. First, you need to load the lookup table in advance, or use the [memlookup](https://docs.logpresso.com/en/query/memlookup-command) command to configure the in-memory lookup table.

#### Syntax

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

Required Parameter

**LOOKUP\_TABLE**

Lookup table to be used for field value conversion. Logpresso has a built-in geoip lookup table, which contains the following fields: **country** (ISO 2-digit country code), **region**, **city**, **latitude**, and **longitude**. You can use this table to convert input field values, which are either IP address types or strings, to values in the mapping fields.

**KEY\_FIELD**

Field name that operates as the key in the lookup table.

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

MAP\_FIELD refers to the name of the field to be mapped based on the key field value in the lookup table. After retrieving a record that matches the key value in the lookup table, it takes the specified field value from the lookup record and assigns it to the output field. You can use the as clause to specify the output field name (ALIAS) of the corresponding lookup mapping field. If you omit the clause, the mapping field name is used as it is.

#### Usage

Lookup the geolocation of the IP address using geoip

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

Enumerates the contents of a lookup table. The contents of the lookup table created based on the database in the web console, geoip lookup table, and lookup using the [memlookup](https://docs.logpresso.com/en/query/memlookup-command) command cannot be enumerated.

You can add a file-based lookup table in "**QUERY > Lookup**" (ENT, STD).

#### Syntax

lookuptable LOOKUP\_TABLE [OPTIONS]

Required Parameter

**LOOKUP\_TABLE**

Lookup table name to enumerate.

Optional Parameter

**limit=INT**

Maximum number of records to load (default: unlimited).

**offset=INT**

Number of records to skip (default: 0).

**FIELD, ...**

Field names, separated by a comma(,).

#### Usage

Enumerate all fields of the lookup table country\_code

lookuptable country\_code

Enumerate only 30 records of **code** fields in the lookup table country\_code

lookuptable country\_code limit=30 code

Enumerate **country** and **population** fields in the lookup table country\_code

lookuptable country\_code country, population

### memlookup

Creates or drops an in-memory lookup table that can be called by the [lookup](https://docs.logpresso.com/en/query/lookup-command) command, or enumerates all the records created in the table.

#### Syntax

To enumerate in-memory lookup tables, or to enumerate records of the specific in-memory lookup table

memlookup [op=list] [name=TABLE]

To create an in-memory lookup table (using the data received by the pipe)

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

To drop an in-memory lookup table

memlookup op=drop name=TABLE

Required Parameter

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

Operation to be performed (default: list).

* build: Builds a lookup table using the data received as input until the query command is complete.
* drop: Drops the mapping table specified by the name option.
* list: Enumerates in-memory lookup table, or records of the lookup table specified by the name option. If the lookup table is not created by memlookup, the query fails. If you execute the memlookup command without any option, it is the same as executing it by specifying only the op=list option.

**name=TABLE**

Target table on which to run the operation specified by the op={build|drop|list} option. When op=list, if you do not specify any lookup table, the result displays the metadata of all in-memory lookup tables. The information shown here is as follows: **name** (lookup name), **key** (key field name), and **size** (number of records of the lookup table). This option can be omitted when op=list.

**key=KEY\_FIELD FIELD, ...**

Key field name when op=build.

**FIELD, ...**

Field names, separated by a comma (,), to be created when op=build.

#### Usage

Create a lookup table from a query.

Create a lookup table named http\_status with the status field as the key and desc1 and desc2 as the data field in a CSV file that has status, desc1 and desc2 columns.

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

Enumerate in-memory lookup tables.

You can see the lookup table information created by memlookup. The information returned is the lookup table name, key field, and the total number of records.

memlookup

The above command has the same result as the following command.

memlookup op=list

Enumerate all records in the specified lookup table.

If you enumerate the list by specifying the name of the lookup, you can see all the information for the lookup.

memlookup name=http\_status

The above command has the same result as the following command.

memlookup op=list name=http\_status

Drop an in-memory lookup table.

You can drop the specified lookup table by assigning drop as the value of the operator (op) option. If you do not specify a lookup name, an error occurs.

memlookup op=drop name=http\_status

### nslookup

Loads the value specified as the domain field, queries the DNS to resolve names, and outputs the result.

#### Syntax

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

Required Parameter

**ns=IP\_ADDR**

IPv4 or IPv6 address of the DNS server.

**DOMAIN\_FIELD**

Field with domain name.

**output FIELD, ...**

Fields to retrieve from the DNS response, separated by a comma (,). Specify one or more of the following:

* ip: IPv4 or IPv6 address
* status: Transaction message status. If there is an error, an error message is displayed.
* flags: Control flags in request/response message
* AA: Authoritative answer
* TC: Truncated
* RD: Recursion desired
* RA: Recursion available
* answers: Response result of DNS server
* authorities: Information of authoritative DNS servers
* additionals: Other additional information

Optional Parameter

**cache=INT**

Cache size of DNS response in bytes (default: 1,048,576. approx. 1 MB)

**timeout=INT**

DNS response timeout in seconds (default: 5).

**type=TYPE**

The type of DNS record to query the DNS server (default: A). Specify one of the following:

* A: IPv4 address record, mapping hostnames to an IP address of the host.
* AAAA: IPv6 address record, mapping hostnames to an IP address of the host.
* CNAME: Canonical name. An alias from one domain name to another domain name.
* MX: Mail exchange record. The mail server responsible for accepting email messages on behalf of a domain name.
* NS: Name server record, delegating a DNS zone to use the given authoritative name servers.
* PTR: PTR resource record, a pointer to canonical name. Commonly used for reverse DNS lookups.
* TXT: Text record, arbitrary human-readable or machine-readable text.

#### Usage

Perform DNS lookup for domain by asking name server 1.1.1.1.

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

## Data Loading

### drop

Discards all incoming data. This is used when executing a query command when executing only batches where no query results are needed, or when trying to measure only the query execution time of the previous command.

#### Syntax

drop

### import

Inserts new records into the specified table. Administrative privileges are required to execute this command.

#### Syntax

import TABLE

Required Object

**TABLE**

Name of table in which to store the data.

#### Usage

Import the entire sys\_cpu\_logs data into old\_sys\_cpu\_logs.

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

### insert

Inserts records by selecting a table based on the field values entered. Administrative privileges are required to execute this command.

#### Syntax

insert table=TABLE

Required Parameter

**table=FIELD**

Name of field to record data. When data is logged, the field specified in the table option is excluded and logs without that field are not logged.

### outputcsv

Exports the specified fields of all input data to the CSV/TSV file.

#### Syntax

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

Required Parameter

**FILE\_PATH**

Path to the CSV/TSV file.

**FIELD, ...**

Fields to be output in a CSV or TSV file, separated by a comma(,).

Both the default CSV and TSV files have fixed columns, orders, and numbers on each of their first lines, but Logpresso data may have different fields existing in each row. So please be sure to define the output fields. If you want to record data in a file without specifying an output field, refer to the 'outputjson' command.

Optional Parameter

**append=BOOL**

Option to enable appending data to the end of the file specified in the FILE\_PATH (default: f).

* t: Appends the field records to the end of the file specified as FILE\_PATH. If the file does not exist, the file is created. You cannot set this option to t when overwrite=t.
* f: NOT append the field records to the end of the file specified as FILE\_PATH. The query fails if the file exists.

**bom=BOOL**

Option to enable the addition of BOM(byte order mark) to the file header (default: f).

* t: Adds BOM to the file header
* f: NOT add BOM to the file header.

**encoding=CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

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

Cycle to flush the output buffer to the file specified as FILE\_PATH. You can use one of the cycle units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second). For example, to flush the buffer every 5 seconds, specify 5s.

**overwrite=BOOL**

Option to enable overwriting the file specified as FILE\_PATH, if it exists (default:f).

* t: Overwrites the file specified as FILE\_PATH, if it exists. You cannot set this option to t when the append option is t.
* f: NOT overwrite the file specified as FILE\_PATH, if it exists. The query fails if the file exists.

**partition=BOOL**

Option to enable macro in the FILE\_PATH (default: f).

* t: Enables macro
* f: Disables macro

You can specify FILE\_PATH to change the directory and file path over time using a macro with the partition=t option. The available macros are {logtime:FMT} and {now:FMT}. For input examples, refer to Usage #2.

* {logtime:FMT}: Names the directory or file based on the log occurrence time.
* {now:FMT}: Names the directory or file based on the current time.

If you set 'partition=t' and do not use a macro on the path, the query fails.

**tab=BOOL**

Option to use tab character as a separator (default: f).

* t: Uses tab character as a separator. This is useful for processing tab-separated values (TSV) files.
* f: Uses comma (,) as a separator.

**tmp=TMP\_FILE\_PATH**

Path to a temporary file. Once you set this option, the command creates a temporary file and outputs the result, and moves the file to the path specified by FILE\_PATH when the query finishes successfully.

#### Usage

Record **src\_ip** and **dst\_ip** field values in the ippair.csv file.

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

Partition the directory according to the log occurrence date using macro, create the file name based on the current time, and then record the **src\_ip** and **dst\_ip** field values.

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

### outputjson

Exports the value of a specific field in JSON format. Each JSON record is separated by a newline.

#### Syntax

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

Required Parameter

**FILE\_PATH**

Path to JSON file.

Optional Parameter

**append=BOOL**

Enables or disables appending data to the end of the file specified in the FILE\_PATH (default: f).

* t: Appends the field records to the end of the file specified as FILE\_PATH. You cannot set this option to t when overwrite=t.
* f: NOT append the field records to the end of the file specified as FILE\_PATH. The query fails if the file exists.

**encoding=CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

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

Cycle to flush the output buffer to the file specified as FILE\_PATH and to flush buffer. You can use one of the cycle units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second). For example, to flush the buffer every 5 seconds, specify 5s.

**overwrite=BOOL**

Option to enable overwriting the file specified as FILE\_PATH, if it exists (default:f).

* t: Overwrites the file specified as FILE\_PATH, if it exists. You cannot set this option to t when the append=t.
* f: NOT overwrite the file specified as FILE\_PATH, if it exists. The query fails if the file exists.

**partition=BOOL**

Option to enable macro in the FILE\_PATH (default: f).

* t: Enables macro
* f: Disables macro

You can specify FILE\_PATH to change the directory and file path over time using a macro when partition=t. The available macros are {logtime:FMT} and {now:FMT}. For input examples, refer to Usage #3.

* {logtime:FMT}: Names the directory or file based on the log occurrence time.
* {now:FMT}: Names the directory or file based on the current time.

If you set 'partition=t' and do not use a macro on the path, the query fails.

**tmp=TMP\_FILE\_PATH**

Path to a temporary file. Once you set this option, the command creates a temporary file and outputs the result, and moves the file to the path specified by FILE\_PATH when the query finishes successfully.

**FIELD, ...**

Fields to be output in the JSON file, separated by a comma(,) (default: all fields).

#### Usage

Record all fields in the output.json file

outputjson /opt/logpresso/files/output.json

Record src\_ip and dst\_ip in the ippair.json file

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

Partition the directory according to the log occurrence date using macro, create the file name based on the current time, and then record the src\_ip and dst\_ip field values.

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

### outputpcap

Records the **payload** field, which is received as an input, as a PCAP file in the specified file system path.

#### Syntax

outputpcap FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to save the PCAP file

#### Usage

Store only packets with a destination or source port of 80 while monitoring the stream created by the pcap\_stream log collector for 5 minutes.

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

### outputtxt

Records the given field values to the specified file system path as a text file.

#### Syntax

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, ...

Required Parameter

**FILE\_PATH**

Path to save the txt file.

**FIELD, ...**

Fields to be output in the TXT file, separated by a comma(,).

Optional Parameter

**append=BOOL**

Enables or disables appending data to the end of the file specified in the FILE\_PATH (default: f).

* t: Appends the field records to the end of the file specified as FILE\_PATH. You cannot set this option to t when overwrite=t.
* f: NOT append the field records to the end of the file specified as FILE\_PATH. The query fails if the file exists.

**delimiter="CHAR"**

Character to use as the field delimiter (default: space).

**encoding=CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

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

Cycle to flush the output buffer to the file specified as FILE\_PATH and flush buffer. You can use one of the cycle units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second). For example, to flush the buffer every 5 seconds, specify 5s.

**gz=BOOL**

Option to enable compressing text files into a gz archive file (default: f).

* t: Enables GZIP compression.
* f: Disables GZIP compression.

**overwrite=BOOL**

Option to enable overwriting the file specified as FILE\_PATH, if it exists (default:f).

* t: Overwrites the file specified as FILE\_PATH, if it exists. You cannot set this option to t when the append=t.
* f: NOT overwrite the file specified as FILE\_PATH, if it exists. The query fails if the file exists.

**partition=BOOL**

Option to enable macro in the FILE\_PATH (default: f).

* t: Enables macro.
* f: Disables macro.

You can specify FILE\_PATH to change the directory and file path over time using a macro when partition=t. The available macros are {logtime:FMT} and {now:FMT}. For input examples, refer to Usage #2.

* {logtime:FMT}: Names the directory or file based on the log occurrence time.
* {now:FMT}: Names the directory or file based on the current time.

If you set 'partition=t' and do not use a macro on the path, the query fails.

**tmp=TMP\_FILE\_PATH**

Path to a temporary file (default: None). Once you set this option, the command creates a temporary file and outputs the result, and moves the file to the path specified by FILE\_PATH when the query finishes successfully.

#### Usage

Record src\_ip and dst\_ip in the ippair.txt file.

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

Partition the directory according to the log occurrence date using macro, creates the file name based on the current time, and then records the src\_ip and dst\_ip field values.

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

### sendmail

Passes the incoming record as email. The sendmail command loads the input into the mail queue as soon as it comes in and sends the mail asynchronously. Administrative privileges are required to execute this command.

To send email, you need to configure the mail server in the System Settings menu.

#### Syntax

sendmail [html=BOOL]

Optional Parameter

**html=BOOL**

Option to regard the **message** field value received as input as html format and parses the body of the mail (default: f).

* t: Parses the **message** field as html.
* f: Parses the **message** field as plain text.

#### Description

The fields as required input are as follow:

**to**

Email addresses separated by a comma(,).

**subject**

Email subject.

**message**

Email body text.

If the input field is missing or the address list is incorrect, the command records an error message in the **\_sendmail\_fail** field. However, errors caused by missing SMTP settings are not displayed.

You can use the 'logpresso.mailQueue' command to see a list of outgoing mail currently pending and delete it at once with the logpresso.clearMailQueue command in the logpresso shell.

#### Usage

Create to, subject, and message fields and pass them to the sendmail command to send an email.

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

### sendsyslog

Sends syslog message to the specified IP address.

#### Syntax

sendsyslog [OPTIONS] dst=IP\_ADDR

Required Parameter

**dst=IP\_ADDR**

IP address of syslog server

Optional Parameter

**format=json|txt**

Format of the log to be sent: either json or txt (default: txt).

* json: Converts all data received as input into JSON format and transmits it
* txt: Transmits the string value of the **line** field as it is.

**port=INT**

Port number of syslog server (default: 514). This designates a value from 1 to 65535 for the port number.

**pri=INT**

PRI constant value defined in RFC 5424: <https://tools.ietf.org/html/rfc5424> (default: 134, meaning Facility: local0, Severity: Info).

The PRI constant is calculated as a value that adds SEVERITY to a value multiplied by 8 in FACILITY. The following table is a table that is configured as a value calculated according to the calculation formula.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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** (default) | 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**

Replaces the source IP address with an arbitrary IP instead of the Logpresso IP address

To replace the source IP address and transmit it, the 'libpcap' library must be installed on your operating system. You may also need to recompile the 'araqne-pcap' library for your operating system. Use PCAP to create and transmit readdressed packets only when specifying the source IP address other than Logpresso IP address. Note that if the packet size exceeds the MTU, the transmission fails.

## Data Merging

### join

Compares the fields of data received as input to the subquery result field and joins them.

'join' does not support stream queries. To apply join to stream queries, use the 'streamjoin' command.

#### Syntax

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

Required Parameter

**KEY\_FIELD, ...**

Key fields as the criteria for join, separated by a comma (,).

**[ SUBQUERY ]**

Subquery that returns the data to be joined with the input data, enclosed in a pair of square brackets ([ ]).

Optional Parameter

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

Join type (default: inner).

* cross: Also known as "Cartesian product", it returns a set consisting of M x N records by combining a set of input data (M records) with a set of subquery results (N records). This type of JOIN does not require a joining condition.
* full: For records with matching keys, it returns by combining them, and for records with no matching keys, it returns them as they are. This is similar to the union of data.
* If the keys match, it combines the subquery field with the input data record and then returns.
* If the keys do not match, it returns the input data and the subquery data respectively as they are.
* inner: In general, join refers to "inner join". It combines and returns only the records with matching keys. It does not return records that do not contain keys. This is similar to the intersection of data.
* left: For records with matching keys, it returns by combining them, and for records with no matching keys, it returns only records of input data.
* leftonly: It returns only records with keys that do not match the set of subquery results. It does not return records with matching keys.
* right: For records with matching keys, it returns by combining them, and for records with no matching keys, it returns only the result of the subquery.
* rightonly: returns only records with keys that do not match the set of subquery results. It does not return records with matching keys.

#### Usage

Execute inner join with the **code** field as the key.

# Pass json with the code field as input data | json "[ {'code':1}, {'code':2}, {'code':3} ]" | # A subquery command that returns json with code and name fields Executes the inner join query using input data and subquery command result data with the "code" field as the key. | join code [ json "[ {'code':1, 'name':'foo'}, {'code':2, 'name':'bar'} ]" ]

**Result of inner join:**

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

Output the result except those retrieved in the subquery (leftonly join).

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

**Result of leftonly join:**

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

The query command executed above has the same execution result as the following query command (application of the result of left join).

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)

Output statistics for each department. It returns all departments even if statistical values do not exist (right join).

json "[ {'id': 1, 'cases': 1000}, {'id':2, 'cases': 2000} ]" | join type=right id [ json "[ {'id':1, 'dept':'sales'}, {'id':2, 'dept':'operation'}, {'id':3, 'dept':'technical'} ]" ]

**Result of right join:**

|  |  |  |
| --- | --- | --- |
| id | dept | cases |
| 1 | sales | 1000 |
| 2 | operation | 2000 |
| 3 | technical |  |

Combine document security violation logs and media control violation logs based on accounts, and output log if they do not match (full join).

json "[ {'acct':'bob', 'document security violation': 1}, {'acct':'alice', 'document security violation': 5} ]" | join type=full acct [ json "[ {'acct':'alice', 'media control violation': 8}, {'acct':'clark', 'media control violation': 3} ]" ]

**Result of full join:**

|  |  |  |
| --- | --- | --- |
| acct | document security violation | media control violation |
| bob | 1 |  |
| alice | 5 | 8 |
| clark |  | 3 |

### streamjoin

Compares the fields of stream data received as input to the subquery result field and joins them.

#### Syntax

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

Required Parameter

**KEY\_FIELD, ...**

Key fields as the criteria for join, separated by a comma (,).

**[ SUBQUERY ]**

Subquery that returns the data to be joined with the input data, enclosed in a pair of square brackets ([ ]).

Optional Parameter

**timeout=INT{s}**

Time for waiting until the subquery is completed (default: no timemout).

**type={inner|left|leftonly}**

Join type (default: inner).

* inner: In general, join refers to "inner join". It combines and returns only the records with matching keys. It does not return records that do not contain keys. This is similar to the intersection of data.
* left: For records with matching keys, it returns by combining them, and for records with no matching keys, it returns only records of input data.
* leftonly: It returns only records with keys that do not match the set of subquery results. It does not return records with matching keys.

#### Description

The streamjoin command loads the results of the subquery into off-heap memory and performs a hash join, so it is faster than the [join](https://docs.logpresso.com/en/query/join-command) command and can also be used in stream queries. However, only inner, left and leftonly are available, and the size of data that can be processed is limited by the capacity of the memory pool. If the subquery fails, the command adds an exception message to the **\_streamjoin\_fail** field.

You can adjust the size of the memory pool by specifying the following options when running Logpresso (default: 500M): -Dlogpresso.streamjoin.max\_buffer\_size=1G

You can check the status of memory usage with the following query:

* Status of memory pool usage: system memory pools
* Status of memory usage by query: system memory objects

#### Usage

Join the data imported from the database with the **code** field as a key (See [dbquery](https://docs.logpresso.com/en/query/dbquery-command)).

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

Join the data imported from the database with the **code** field as a key. However, limit SQL queries to 10 seconds.

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

### union

Merges the results of the subquery. It does not guarantee the output order, because union is executed in parallel with other queries. As with running statistical processing, this is primarily used when high performance is required and the order is not important.

#### Syntax

union [ SUBQUERY ]

Required Parameter

**[ SUBQUERY ]**

Subquery that returns the data to be combined with the input data, enclosed in a pair of square brackets ([ ]).

#### Usage

Merge SQL query results of 2 DBs.

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

## Complex Event Processing

### evtctxadd

Creates the event context with the specified key, if the input data matches the conditional expression.

#### Syntax

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

or

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

Required Parameter

**dynamic=BOOL**

Option to enable dynamic options (default: f).

* t: Gets topic, expire, timeout, maxrows options from input records. You cannot use the topic, expire, timeout, and maxrows options when dynamic=t.
* f: Disables dynamic option.

**topic=STR**

Name of event context. the topic acts like the table name in an in-memory database. You cannot use this option when dynamic=t.

**key=KEY\_FIELD**

Name of the field to record the unique key that distinguishes the event context.

**CONDITIONAL\_EXPR**

Conditional expression for creating an event context.

Optional Parameter

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

Expiration period in units of mon (month), d (day), h (hour), m (minute), and s (second). The event context is deleted once a specified expiration period has passed from when the event context is created. Once the expiration period is set, it is not extended, even if there is input data that matches the conditional expression CONDITIONAL\_EXPR. You cannot use this option when dynamic=t.

**maxrows=INT**

Maximum number of rows to store in the event context (default: 10). You cannot use this option when dynamic=t.

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

Length of time after the last event received until timeout. You can specify in units of mon (month), d (day), h (hour), m (minute), and s (second). You cannot use this option when dynamic=t.

#### Usage

**Generate a timeout if it takes more than 10 seconds to receive a response after sending a message.**

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

The example query command consists of the following event context creation/deletion commands:

* If the **type** value is send, use the evtctxadd command to create the event context.
* If the **type** value is recv, use the [evtctxdel](https://docs.logpresso.com/en/query/evtctxdel-command) command to delete the event context.

When the event context condition occurs, both commands distinguish the event context by associating the topic txmatch and the event context key field **txkey**.

Now, if the following event data is passed as an input,

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

The event context is created when the first data is provided. Different events occur depending on the time the second data is provided.

* If the second data is entered within 10 seconds, an event context deletion (EventCause.REMOVAL) event occurs.
* If the second data is entered after 10 seconds, or if it is not provided, a timeout (EventCause.TIMEOUT) event occurs.

Depending on the cause of the deletion of the event context, you can perform different processing with subsequent commands.

### evtctxdel

Removes the event context with the given key, if the input data matches the conditional expression.

#### Syntax

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

Required Parameter

**dynamic=BOOL**

Option to enable dynamic options (default: f).

* t: Gets topic option from input records. You cannot use the topic option when dynamic=t.
* f: Disables dynamic option.

**topic=STR**

Name of event context. The topic is like the table name in an in-memory database.

**key=KEY\_FIELD**

Field to extract the key value that distinguishes the event context.

**CONDITIONAL\_EXPR**

Conditional expression for removing an event context.

### evtctxdrop

Deletes all event contexts corresponding to the specified topic at once.

#### Syntax

evtctxdrop all=BOOL

or

evtctxdrop topic="STR"

Required Parameter

You must specify one of the following options:

**all=BOOL**

Option to delete all event contexts at once (default: f). This option cannot be used with topic=STR.

**topic="STR"**

Name of event context. The command deletes all event contexts with names that match the topic. This option cannot be used with all=t.

### evtctxlist

Loads the list of event contexts.

#### Syntax

evtctxlist [topic=STR]

Optional Parameter

**topic=STR**

Name of event context. The command loads the event context with a name that matches the topic (default: the entire list of event topics).

## Machine Learning

### anomalies

Calculates the anomaly score using the Isolation Forest modeling (a way of creating a decision tree model by sampling some data).

#### Syntax

Calculate the anomaly score using a stored training model.

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

Calculate the anomaly score using a model trained based on subquery results.

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

Required Parameter

**FIELD, ...**

Fields to be used for the Isolation Forest modeling. Use a comma(,) as a separator.

**model=MODEL**

Name of the Isolation Forest model. You can generate and train the Isolation Forest model by connecting to the Logpresso engine via CLI.

**[ SUBQUERY ]**

Subquery that returns the data set for model training.

Optional Parameter

**sample=INT**

Number of samples to draw when training the Isolation Forest model (default: the square root of the number of samples).

**size=INT**

Number of trees within the Isolation Forest (default: 100).

#### Description

The anomaly score, ranging from 0 to 1, is assigned to the **\_score** field.

* The higher the score, the more likely it is an anomaly.
* A score much smaller than 0.5 indicates normal observations.
* If all scores are close to 0.5, the entire sample does not seem to have clearly distinct anomalies.

#### Usages

Calculate the anomaly score using the anomal\_stock model.

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

Calculate using a model trained based on the training data set returned from a subquery.

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

Predicts data for given time series data.

#### Syntax

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

Required Parameter

**TIME\_SERIES\_FIELD**

Field that contains time-series data.

Optional Parameter

**count=INT**

Number of rows of data to be predicted(default: 5)

**period=INT**

Time series cycle. It is automatically calculated using a Fast Fourier Transform (FFT) if not specified.

**seed=INT**

Fixed seed value. Specify this option when you want to keep the same result for the same input.

**time=FIELD**

Field to be used as the time record (default: **\_time** field)

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

Grouping fields in the aggregation with by directive, separated by a comma (,). This option MUST follow after the TIME\_SERIES\_FIELD option.

#### Description

It is recommended that you set a constant interval between the data in the time field by using the [timechart](https://docs.logpresso.com/en/query/timechart-command) command, and then use the forecast command. There must be at least 4 data points for each grouping field, and the value of the period must be less than one-half of the number of data points.

#### Usage

Predict values of the **count** field.

forecast count

Set the time series cycle of the **traffic** field to 5, and assign the time series prediction data according to the time written in the **set\_time** field. Assign 1234 as a fixed seed value to maintain the same result.

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

Predict 10 data points of the series from the **sent\_bytes** field aggregated by the **region** group.

forecast count=10 sent\_bytes by region

### kmeans

Classifies the input record into k clusters based on Euclidean distance.

#### Syntax

kmeans [OPTIONS] FIELD, ...

Required Parameter

**FIELD, ...**

Name of the fields to be calculated, separated by a comma (,).The field value must be numeric, and any input record whose specified field value is not numeric is ignored. Up to 100,000 input records are allowed. The command classifies the records into the N number of clusters (N starting from 1) and assigns them to the **\_cluster** field. If there are more than 100,000 valid input records, it ignores records after 100,000.

Optional Parameter

**k=INT**

Number of clusters (default: 3)

**iter=INT**

Number of times to repeat kmeans (default: 100,000)

#### Usage

You can test the operation method of the kmeans command with iris data, which is often quoted in machine learning. Run the classification using length and width and compare it to the name of the actual species (download: <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

Calculates Local Outlier Factor (LOF) by calculating the Local Reachability Density (LRD) of each point based on the k-nearest neighbors and calculating the ratio of the local reachability density relative to the adjacent neighbors.

#### Syntax

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

Required Parameter

**FIELD, ...**

Fields that contain numeric data such as integers, real numbers, and dates. Use comma (,) as a separator.

Optional Parameter

**k=INT**

Number of adjacent nodes to be used for calculation (default: 10)

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

Grouping fields in the aggregation with by directive, separated by a comma (,). This option MUST follow after FIELD, ....

If you want to calculate the scoring for each group by using the by clause, the number of records in each group must be greater than the number of adjacent nodes (the value specified by k=INT). If the number of records in the group is less than the number of adjacent nodes, the LOF in the **\_lof** field is not calculated as intended.

#### Description

This calculates the LOF score on the **\_lof** field for each record, and this value can be classified as follows:

* If the value is greater than 1 (LOF(k) > 1): It is located outside the cluster. The greater it is than 1, the more likely it is to be an anomaly.
* If the value is an approximation of 1 (LOF(k) ≈ 1): It is located at the boundary of the cluster.
* If the value is less than 1 (LOF(k) < 1): It is located inside the cluster.

#### Usage

Calculate the anomaly based on the field values of **sepal\_length** and **sepal\_width** (download: <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

Returns the predicted target using the Random Forest modeling (a way of training an ensemble of decision trees).

#### Syntax

Predict using a stored training model.

rforest [size=INT] model=MODEL

Predict using a model trained based on subquery results.

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

Required Parameter

**FIELD, ...**

Fields as predictor variables for the Random Forest modeling.

**model=MODEL**

Name of the Random Forest model. You can generate and train the Random Forest model by connecting to the Logpresso engine via CLI.

**target=TARGET\_FIELD**

Field as a target variable for the Random Forest modeling.

**[ SUBQUERY ]**

Subquery that returns the data set for model training.

Optional Parameter

**size=INT**

Number of trees within the random forest (default: 100)

#### Description

This command returns the predicted value of the target field into the **\_guess** field.

#### Usages

Predict using the rforest\_titanic model.

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

Predict using a model trained based on the training data set returned from a subquery.

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

Decomposes time series data into trends, seasonality, and errors. This command returns up to 1,000 for each grouping field (the field specified by the by clause), and if you do not specify the grouping field, the number of output rows is limited to 1,000.

To increase the limit of the number of stl outputs, add the -Dlogpresso.stl.limit=N booting option and provide the desired value.

#### Syntax

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

Required Parameter

**NUMERIC\_FIELD**

Time series data to be calculated. The field value must be numbers, such as integers, real numbers, or dates.

Optional Parameter

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

Time series cycle. You can specify time in units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second). The command performs the analysis assuming that the time series data has repeatability according to the cycle you specified. If you do not specify a time series cycle, it automatically calculates the seasonality cycle through spectral analysis.

**by GRP\_FIELD**

Grouping fields with by directive, separated by a comma(,).

#### Description

STL is an abbreviation for seasonal-trend decomposition procedure based on loess. If the period is not specified, the command automatically calculates the seasonality cycle through spectral analysis.

The stl command analyzes the time series data and outputs the **\_trend**, **\_seasonal**, and **\_error** fields. If the time series data has no cycle (e.g., period=0m), the **\_seasonal** field does not output.

## Procedure

### proc

Executes the user-defined procedure.

#### Syntax

proc PROC\_NAME(PARAMETER, ...)

Required Parameter

**PROC\_NAME(PARAMETER\_1, PARAMETER\_2, ... PARAMETER\_N)**

Procedure to execute and parameters in the pre-defined format. If you pass an argument to the procedure according to the parameter format defined, the argument is set to the query parameter and the query that is defined in advance is executed after. You can pass an expression that can be evaluated as a constant according to the parameter type defined by the procedure. The owner of the procedure or an authorized user executes the query with the owner permission for the procedure.

#### Usage

Save the query command that extracts the record of more than N% overload over the last 24 hours in the web console as a procedure. The name of the procedure here is cpu\_overload. You can create a procedure query to refer to the query parameter using the [$()](https://docs.logpresso.com/en/query/$-function) function.

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

Now, you can call the procedure as follows:

proc cpu\_overload(90)

## External System Integration

### dbcall

Calls the SQL stored procedure and returns the result of execution (result set and/or output parameters).

#### Syntax

dbcall PROFILE {SQL\_STATEMENT}

Required Parameter

**PROFILE**

JDBC connect profile. You can configure the connect profile in the web console.

**{SQL\_STATEMENT}**

SQL statement that calls the SQL stored procedure enclosed in a pair of curly braces({ }).

You can define input and output parameters that start with a colon (:) in the SQL query command.

* The input parameter is defined in the form of :name, and a query parameter defined by the [set](https://docs.logpresso.com/en/query/set-command) command is inserted.
* The output parameter is defined in the form of :name(type). The available output parameter types are varchar, int and datetime.

The output method is as follows:

* If the SQL query command returns only the output parameter, the dbcall command outputs 1 tuple consisting of the output parameter.
* If the SQL query command returns the result set and output parameter, the dbcall command adds the output parameter field to all tuples of the result set and returns them.
* If the SQL query command returns multiple result sets, the dbcall command loads all result sets and then outputs them.

#### Usage

Load the column configuration of a specific table on a Microsoft SQL server.

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

Load the line value with an id of 1000 as a user-defined procedure in Microsoft SQL Server.

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

### dbload

Converts the query result into an SQL query command and provides it into an external SQL server. It has the same functionality as the [dboutput](https://docs.logpresso.com/en/query/dboutput-command) command, and only the default value of the rowretry option is different.

#### Syntax

dbload PROFILE [batchsize=INT] [database=SCHEMA] [rowretry=t] [stoponfail=t] [type=update] table=TABLE FIELD, ...

Required Parameter

**PROFILE**

JDBC connect profile. You can configure the connect profile in the web console.

**table=TABLE**

Name of table in which to input the data.

**FIELD, ...**

Field names to be provided in the database, separated by a comma(,). If you prefix the plus sign (+) with the field name, it is recognized as a key column.

The field name must match the column name of the target table. If the field and column names do not match, use the [rename](https://docs.logpresso.com/en/query/rename-command) command before the dboutput command to match the name of the column in the SQL database.

Optional Parameter

**batchsize=INT**

Size of processing unit to be applied to the database batch transaction. Larger units are more efficient because they are committed at once, but if the transaction fails, the number of rollback records also increases. 2000 is recommended. If you do not specify a processing unit size, the processing speed may be slow because the command commits the transaction one by one.

**database=SCHEMA**

Schema or database to switch to when connecting to the server.

**rowretry=BOOL**

Option to retry transaction row by row if any query fails (default: t). When enabled, performance may be degraded, but data loss can be minimized.

* t: Retries the transaction row by row after the batch transaction fails.
* f: NOT retry the transaction row by row after the batch transaction fails.

**stoponfail=BOOL**

Option to stop the query when transaction fails (default: f).

* t: Stops the query when the transaction fails.
* f: Skips the failed transaction and executes the next transaction.

**type=update**

Type of SQL query: either insert or update (default: insert). If you set it to update, you need to specify one or more fields as key columns. This command checks whether the SQL database has a key column by executing an SQL SELECT statement, and runs an INSERT command if there is no key column or an UPDATE command if there is a key column.

### dblookup

Assigns the input record to the placeholder of the SQL query and execute the query. The column values of the first loaded record is assigned to the fields.

#### Syntax

dblookup PROFILE [bypass=BOOL\_EXPR] SQL\_STATEMENT

Required Parameter

**PROFILE**

JDBC connect profile. You can configure the connect profile in the web console.

**SQL\_STATEMENT**

SQL query statement to run.

You can define the input parameter that starts with a colon (:) in the SQL query command. The input parameter is in the form of :name, and the field value of the input record is assigned.

Optional Parameter

**bypass=BOOL\_EXPR**

Boolean expression as condition for not executing the SQL query command. If the condition is true, the command exports the output without executing the SQL query statement.

bypass=BOOLEAN\_EXPR generally configures the conditional expression so that the command does not execute SQL unless there is a field value that falls into the conditional clause.

#### Usage

Expand the field by importing the user name (name) and gender (sex) with the login value.

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

### dboutput

Converts the query result into an SQL query command and provides it into an external SQL server. It has the same functionality as the [dbload](https://docs.logpresso.com/en/query/dbload-command) command, and only the default value of the rowretry option is different.

#### Syntax

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

Required Parameter

**PROFILE**

JDBC connect profile. You can configure the connect profile in the web console.

**table=TABLE**

Name of table where you want to input the data.

**FIELD, ...**

Field names to be provided in the database separated by a comma(,). If you prefix the plus sign (+) with the field name, it is recognized as a key column.

The field name must match the column name of the target table. If the field and column names do not match, use the [rename](https://docs.logpresso.com/en/query/rename-command) command before the dboutput command to match the name of the column in the SQL database.

Optional Parameter

**batchsize=INT**

Size of processing unit to be applied to the database batch transaction. Larger units are more efficient because they are committed at once, but if the transaction fails, the number of rollback records also increases. 2000 is recommended. If you do not specify a processing unit size, the processing speed may be slow because the command commits the transaction one by one.

**database=SCHEMA**

Schema or database to switch after connecting.

**rowretry=BOOL**

Option to retry transaction row by row when the batch transaction fails (default: f). When enabled, performance may be degraded, but data loss can be minimized.

* t: Retries the transaction row by row after the batch transaction fails.
* f: NOT retry the transaction row by row after the batch transaction fails.

**stoponfail=BOOL**

Option to stop the query when transaction fails (default: f).

* t: Stops the query when the transaction fails.
* f: Skips the failed transaction and executes the next transaction.

**type=update**

Type of SQL query: either insert or update (default: insert). If you set it to update, you need to specify one or more fields as key columns. This command checks whether the SQL database has a key column by executing an SQL SELECT statement, and runs an INSERT command if there is no key column or an UPDATE command if there is a key column.

### dbquery

Runs an SQL query on an external database server.

#### Syntax

dbquery PROFILE SQL\_STATEMENT

Required Parameter

**PROFILE**

JDBC connect profile. You can configure the connect profile in the web console.

**SQL\_STATEMENT**

SQL query statement to run. This loads all result sets loaded through JDBC as key-value pairs.

You can define input and output parameters that start with a colon (:) in the SQL query command.

* The input parameter is in the form of :name, and a query parameter defined by the [set](https://docs.logpresso.com/en/query/set-command) command can be inserted.
* The output parameter is defined in the form of :name(type). The available output parameter types are varchar, int and datetime.

#### Usage

Load 100 logs from the weblogs table.

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

Load the list of employees for the last week from the employee table using the input parameter.

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

### dbscript

Executes an SQL script to load data. Administrative privileges are required to execute this command.

#### Syntax

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

Required Parameter

**PROFILE**

JDBC connect profile. You can configure the connect profile in the web console.

**SQL\_FILE\_PATH**

Path to the SQL script file to run. The maximum length of an SQL script file cannot exceed 1 MB (1,048,576 bytes).

The SQL script file must meet the following conditions: - Only the 'SELECT' query is available. - You can use a question mark (?) to specify where to insert the parameter.

Optional Parameter

**cs=CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**:parameter ...**

Parameters to be referenced in the script. Use whitespaces as a separator. It replaces the placeholder in the SQL file in the order of the parameters. The name of the parameter must start with a colon (:).You can set the parameters using the [set](https://docs.logpresso.com/en/query/set-command) command or take over the procedure call argument as a parameter. For calling procedures, refer to the [dbcall](https://docs.logpresso.com/en/query/dbcall-command) command.

### ftp

Allows you to browse the file system on the FTP server and transmit the input records to the file.

#### Syntax

ftp PROFILE SUBCOMMAND [OPTIONS] PATH

Required Parameter

**PROFILE**

FTP connect profile. You can configure the profile in the web console.

**SUBCOMMAND**

Command to be executed in the FTP session: ls, cat, put.

**ls**

Lists files and directories in the PATH on the FTP server.

**cat**

Reads files in the PATH and outputs their contents as records in the fields line by line. The available export file formats are CSV, JSON, TSV and plain text files.

**put**

Converts the name and values specified by the fields option into the file in a format specified by the format option, then transmits that file to the PATH on the FTP server.

**PATH**

Path to a directory or file. If you use a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. /var/log/httpd/access.\*).

Optional Parameter

The options for each SUBCOMMAND are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Options | `cat` | `put` | `ls` |
| append | - | O | - |
| encoding | O | O | O |
| fields | - | O | - |
| format | O | O | - |
| limit | O | - | - |
| offset | O | - | - |
| overwrite | - | O | - |

**append=BOOL**

Option to enable appending data to the end of the file specified in the PATH (default: f).

* t: Appends the field records to the end of the file specified by PATH. If the file does not exist, the file is created. You cannot set this to t when overwrite=t.
* f: NOT append the field records to the end of the file specified by PATH. The query fails if the file exists.

When using the 'append=t' option, always keep the list order of the 'fields' option the same so that data can be consistent.

**encoding=CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**fields=FIELD\_LIST**

Fields to be transmitted to the FTP server (default: line). Use comma (,) without any whitespace, as a separator. If there is no **line** field or the specified field is empty, it is replaced with a hyphen symbol (-) in the output to indicate the field is empty.

**format={csv|json|tsv}**

File format (csv, json, tsv, default: plain text).

* csv or tsv:
* When **SUBCOMMAND** is cat, the first line is considered a regular record. Field name (column header) is assigned in the form columnN (N is a number starting from 0).
* , When **SUBCOMMAND** is put, field names (column header) are assigned with the field names specified by the fields option.
* json:
* When **SUBCOMMAND** is cat, it parses the file into the records of key-value pairs line by line. Field names are specified as keys and field values as values.
* When **SUBCOMMAND** is put, it transmits the records consisting of the key-value pairs of the fields specified by the fields option. If the fields option is not specified, records consisting of all field values are transmitted.
* Not specified (plain text):
* When **SUBCOMMAND** is cat, it loades the values to the line field line by line.
* When **SUBCOMMAND** is put, it transmits the file in a text format. Values are separated by tab characters in plain text, and empty values (nulls) are replaced with hyphens (-).

**limit=INT**

Number of records to be output when importing files from the FTP server (default: unlimited).

**offset=INT**

Number of records to skip when importing files from the FTP server (default: 0).

**overwrite=BOOL**

Option to enable overwriting the file specified as PATH, if it exists (default:f).

* t: Overwrites the file specified as PATH, if it exists. You cannot set this to t when the append is t.
* f: NOT overwrite the file specified as PATH, if it exists. The query fails if the file exists.

#### Usage

You first need to configure an FTP connect profile (PROFILE) and an Apache weblog parser (httpd) to run this example. You can specify the Apache weblog parser using the following options:

|  |  |  |
| --- | --- | --- |
| Parser Name | Parser Type | Log Format |
| httpd | Apache Web Log | %h %l %u %t "%r" %>s %O "%{Referer}i" "%{User-Agent}i" |

Parse the wp-nginx.log file and transmit it to the FTP server as a CSV file.

To better understand the format option, try putting (transmit) the file by not specifying the format option or specifying it to json or tsv.

wget url="https://raw.githubusercontent.com/logpresso/dataset/main/wp-nginx.log" | eval line = subarray(split(line, "\n"), 0) | explode line | parse httpd | ftp FTP\_PROFILE put format=csv overwrite=t fields=remote\_host,login,user,date,request,status,sent,referer,user\_agent /opt/logpresso/wp.csv

List directories or files from the FTP server.

ftp PROFILE ls /opt/logpresso ftp PROFILE ls /opt/logpresso/wp.\*

Each query result field has the following meanings:

* type (string): dir when it is a directory, file when it is a file
* name (string): Directory or file name
* file\_size (integer): File size, 0 when it is a directory
* owner (string): Owner
* group (string): Owned group
* modified\_at (date): Last modified time

Read the first 5 records of the wp.csv file.

ftp PROFILE cat limit=5 /opt/logpresso/wp.csv

Read the wp.json file into JSON format

ftp PROFILE cat format=json /opt/logpresso/wp.json

### hdfs

Allows you to browse HDFS or transmit the input records to the file.

#### Syntax

hdfs PROFILE SUBCOMMAND [OPTIONS] PATH

Required Parameter

**PROFILE**

HDFS connect profile. You can configure the profile in the web console.

**SUBCOMMAND**

Command to be executed in the FTP session: ls, cat, put.

* ls: Lists all information about the files in the path specified by PATH.
* lsr: Recursively lists all the files in the directory specified by PATH.
* cat: Loads the contents of text files, CSV files, JSON files, HDFS sequence, and plain text files in the HDFS file system. It parses according to the file format specified by the format option.
* put: Transmits the values of the field specified by the fields option to the HDFS file system.
* rm: Removes the file in the path specified in the input record.

**PATH**

Path to a directory or file. If you use a wildcard (\*) int the file name, you can retrieve all files containing a specific string pattern in the file name(e.g. /var/log/httpd/\*).

* When **SUBCOMMAND** is ls, you can enter either a directory or a file path.
* When **SUBCOMMAND** is cat, you can enter only the file path.
* When **SUBCOMMAND** is put, you can enter only the file path.
* When **SUBCOMMAND** is rm, the PATH is not required.

Optional Parameter

The options for each SUBCOMMAND are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Options | `cat` | `put` | `ls`/`lsr`/`rm` |
| append | - | O | - |
| compression\_type | - | O | - |
| fields | - | O | - |
| flush | - | O | - |
| format | O | O | - |
| key\_field | - | O | - |
| key\_type | - | O | - |
| limit | O | - | - |
| offset | O | - | - |
| partition | - | O | - |
| value\_field | - | O | - |
| value\_type | - | O | - |

**append=BOOL**

Enables or disables appending data to the end of the file specified in the PATH (default: f).

* t: Appends the field records to the end of the file specified as PATH.
* f: NOT append the field records to the end of the file specified as PATH. The query fails if the file exists.

**compression\_type=TYPE**

Compression type: either block or record (default: no compression).

* block: block-by-block compression
* record: record-by-record compression

**fields=FIELD,...**

Fields to be transmitted to the HDFS server (default: line). Use comma (,) without any leading or trailing whitespaces as a separator. If there is no line field or the specified field is empty, it is replaced with a hyphen symbol (-) in the output to indicate the field is empty.

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

Cycle to flush output buffer to the file specified as PATH. You can use one of the cycle units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second). For example, to flush the buffer every 5 seconds, specify 5s.

**format=FORMAT**

File format (csv, json, sequence, tsv).

* csv, tsv
* When **SUBCOMMAND** is cat, the first line is considered a regular record. Field name (column header) is assigned in the form columnN (N is a number starting from 0).
* When **SUBCOMMAND** is put, field names (column header) are assigned with the field names specified by the fields option.
* json
* When **SUBCOMMAND** is cat, it parses the file into the records of key-value pairs line by line. Field names are specified as keys and field values as values.
* When **SUBCOMMAND** is put, it transmits the records consisting of the key-value pairs of the fields specified by the fields option. If the fields option is not specified, records consisting of all field values are transmitted.
* sequence
* When **SUBCOMMAND** is cat, it converts the Writable implementation of HDFS to a Logpresso type (the data type of Java) and reads the file record by record.
* The key field is named **key**. The key is converted to a string regardless of its original type.
* When the value field is of type MapWritable, the internal key-value mapping is returned to the field of the returned row. The Hadoop's Writable implementation is converted into a Logpresso type.
* When the value field is not a MapWritable type, it outputs the value to the value field.
* When **SUBCOMMAND** is put, it transmits the file in HDFS sequence format unless it falls under the following conditions:
* When either key or value of the record is empty, that row is not transmitted.
* When the type of value does not match the type specified by the value\_type option, string type is converted to a string, and numeric types such as int, long, float, and double are converted to 0, and boolean type to false.
* When it can convert the type of value without compromising precision, it converts it to the specified type and outputs (for example, when a long type is specified with the value\_type option but an int value comes in, it is converted to a long type and returned).
* Not specified (plain text):
* When **SUBCOMMAND** is cat, values are loaded in the **line** field line by line.
* When **SUBCOMMAND** is put, the file is transmitted in plain text format. Values are separated by tab characters in plain text, and empty values (nulls) are replaced with hyphens (-).

**key\_type=HDFS\_TYPE**

HDFS type in the HDFS data conversion type of Logpresso.

**key\_field=KEY\_FIELD**

Name of the key field. If you do not set this option, the LongWritable counter, which starts from 1, is used.

**limit=INT**

Number of records to be output when importing files (default: unlimited).

**offset=INT**

Number of records to skip when importing files (default: 0)

**partition=BOOL**

Option to enable macro in the PATH (default: f).

* t: Enables macro
* f: Disables macro

You can specify PATH to change the directory and file path over time using a macro when partition=t. The available macros are {logtime:FMT} and {now:FMT}.

* {logtime:FMT}: Names the directory or file based on the log occurrence time.
* {now:FMT}: Names the directory or file based on the current time.

If you set 'partition=t' and do not use a macro on the path, the query fails.

**value\_type=HDFS\_TYPE**

HDFS type in the HDFS data conversion type of Logpresso.

**value\_field=VALUE\_FIELD**

Name of the value field. If you do not set the name of the value field, all fields are transmitted to a single MapWritable.

#### Description

Logpresso uses the data types defined by Logpresso, such as Java standard data types and IP addresses. When importing or transmitting data from HDFS, Logpresso performs the conversion operation according to the HDFS data type. For information on converting data by type, refer to the following table.

**Logpresso and HDFS data conversion type**

|  |  |  |
| --- | --- | --- |
| Logpresso type | HDFS type | Description |
| String | Text | String |
| Null | NullWritable | Null |
| Boolean | BooleanWritable | Boolean |
| Integer | IntWritable, VIntWritable | 4-byte (32 bits) integer |
| Long | LongWritable, VLongWritable | 8-byte (64 bits) integer |
| Float | FloatWritable | Single precision real number |
| Double | DoubleWritable | Double precision real number |

#### Usage

Retrieve the root path file list by accessing the profile of the name vm.

hdfs vm ls /

The output fields are as follows:

* **type** (string): "dir" when it is a directory, "file" when it is a file
* **name** (string): File name
* **path** (string): Absolute path of the file
* **replication** (integer): Number of copies, 0 when it is a directory
* **file\_size** (integer): File size, 0 when it is a directory
* **block\_size** (integer): Block size, 0 when it is a directory
* **modified\_at** (date): Last modified time
* **permission** (string): Permission settings
* **owner** (string): Owner
* **group** (string): Owned group

Read 5 rows after skipping the first line of the /tmp/LICENSE.txt file by accessing the vm profile.

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

Read 3 rows of the /tmp/malware.csv file by accessing the vm profile.

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

Read 1 row of the /tmp/iis.json file by accessing the vm profile.

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

Read 2 records of the /tmp/classloading.seq file by accessing the vm profile.

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

Output only UnloadedClassCount of LoadedClassCount among the JMX class loading logs to the /tmp/class.txt path.

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

Output the sys\_cpu\_logs log to the directory under /tmp by date.

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

Output LoadedClassCount, UnloadedClassCount, and TotalLoadedClassCount among the JMX class loading logs.

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

Output the JMX class loading log as a JSON file.

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

Output the entire JMX class loading log as an HDFS sequence file.

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

Output LoadedClassCount among the JMX class loading logs.

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

### mongo

Allows you to browse MongoDB or transmit the input records to the file.

#### Syntax

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

Required Parameter

**PROFILE**

MongoDB connect profile. You can configure the profile in the web console.

**SUBCOMMAND**

Command to be executed in the MongoDB session: dbs, cols

* cols: Lists collections for the database specified by the database=DB\_NAME option. If no database is specified, all collections from all databases are shown.
* dbs: List all databases in MongoDB. Used alone without any other arguments.

**MONGO\_METHOD**

MongoDB native method. For the collection specified by COL\_NAME, it receives an input record or expression as a parameter argument and executes the method.

Optional Parameter

**database=DB\_NAME**

Database on which to run the cols command.

#### Description

Output fields by SUBCOMMAND

The output fields of the dbs command are:

* **name**: Database name
* **disk\_usage**: Disk usage in bytes
* **empty**: true if the database is empty and false otherwise.

The output fields of the cols command are:

* **name**: Collection name
* **type**: Type (collection)
* **options**: Collection configuration options
* **info**: Additional information such as whether it is read-only.
* **idIndex**: \_id index specification

MONGO\_METHOD

The mongodb command supports the following MongoDB native methods. You can check the usage of each method in [MongoDB Reference Manual](https://www.mongodb.com/docs/manual/reference/method/).

* [db.COL\_NAME.find()](https://www.mongodb.com/docs/manual/reference/method/db.collection.find/)
* [db.COL\_NAME.insert()](https://www.mongodb.com/docs/manual/reference/method/db.collection.insert/)
* [db.COL\_NAME.updateOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateOne/)
* [db.COL\_NAME.updateMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.updateMany/)
* [db.COL\_NAME.deleteOne()](https://www.mongodb.com/docs/manual/reference/method/db.collection.deleteOne/)
* [db.COL\_NAME.deleteMany()](https://www.mongodb.com/docs/manual/reference/method/db.collection.deleteMany/)

find()

db.COL\_NAME.find("FILTER\_EXPR", ["PROJECTION\_EXPR"])

Selects documents specified by COL\_NAME in a collection and returns a cursor to the selected documents.

**"FILTER\_EXPR"**

A JSON expression that specifies search conditions (filters) If you specify it as null, it retrieves all documents.

**["PROJECTION\_EXPR"]**

A JSON expression that specifies the fields to be returned in the documents that match the search condition. If you omit the expression, it returns all fields in the documents that match the search condition. For more information, see ["Projection"](https://www.mongodb.com/docs/manual/reference/method/db.collection.find/) in MongoDB Documentation.

For information on query operators available in FILTER\_EXPR and PROJECTION\_EXPR, refer to the ["Query and Projection Operators"](https://www.mongodb.com/docs/manual/reference/operator/query/) in the MongoDB Documentation.

insert()

db.COL\_NAME.insert()

Inserts a document or documents into a collection specified by COL\_NAME. If the document contains an **\_id** value, it is used as a unique identification key (ObjectId). The **\_id** values must be unique. If the document does not specify an **\_id** field, then MongoDB adds a 12-byte hexadecimal unique identification key for the document before inserting. See usage #4, #5.

updateOne()

db.COL\_NAME.updateOne("KEY", "UPDATE")

Searches the collection specified by COL\_NAME with the value of the KEY field and modifies the value of the UPDATE field in the first document returned in the collection. See usage #6.

When using the **\_id** field as KEY, if the unique identifier is an automatically generated 12-digit binary, the \_id value MUST be set to the binary type.

updateMany()

db.COL\_NAME.updateMany("KEY\_LIST", "UPDATE\_LIST")

Searches the collection specified by COL\_NAME with the value of the KEY\_LIST field and modifies the value of the UPDATE\_LIST field in all the documents returned. Both KEY\_LIST and UPDATE\_LIST use a comma (,) as a separator. Unlike the db.COL\_NAME.updateOne() method, this modifies all searched documents.

deleteOne()

db.COL\_NAME.deleteOne("KEY")

Searches the collection specified by COL\_NAME with the value of the KEY field and deletes the first document returned in the collection.

When using the **\_id** field as KEY, if the unique identifier is an automatically generated 12-digit binary, the \_id value MUST be set to the binary type.

deleteMany()

db.COL\_NAME.deleteMany("KEY\_LIST")

Searches the collection specified by COL\_NAME with the value of the KEY\_LIST field and removes all documents. KEY\_LIST uses a comma (,) as a separator.

#### Usage

Retrieve all documents in the inventory collection.

mongo PROFILE db.inventory.find()

Retrieve documents with 3 or more stars from the restaurants collection.

mongo PROFILE db.restaurants.find("{stars: {$gte: 3}}")

Output only the name and stars fields from the restaurants collection.

mongo PROFILE db.restaurants.find(null, "{name: true, stars: true}")

Provide an arbitrary JSON document into the MongoDB restaurants collection.

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()

Provide the lastest 10 items of the Logpresso inventory table into the MongoDB inventory collection.

table limit=10 inventory | mongo PROFILE db.inventory.insert()

Search the restaurants collection based on the name field and modify the stars value.

json "{}" | eval name="Café Con Leche", stars=4 | mongo PROFILE db.restaurants.updateOne("name", "stars")

Search 1 document with an ObjectId of 5982df1b7098262f64d4ffaf and then delete it.

json "{}" | eval \_id = fromhex("5982df1b7098262f64d4ffaf") | mongo stream2 db.restaurants.deleteOne("\_id")

### rss

Receives and outputs the feeds in the form of RSS1, RSS2, and ATOM through HTTP communication.

#### Syntax

rss [strip=BOOL] url="FEED\_URL"

Required Parameter

**url="FEED\_URL"**

URL of the RSS feed.

Optional Parameter

**strip=BOOL**

Option to strip HTML tags in the RSS feed (default: f).

* t: Strips HTML tags.
* f: NOT strip HTML tags.

#### Description

If you load the RSS feed, the command outputs the following fields for each record.

* guid: Identifier
* author: Author
* title: Title
* content: Body
* link: URL link
* source: Source
* created\_at: Creation time

#### Usage

Load RSS feed.

rss url="http://rss.slashdot.org/Slashdot/slashdotMain" strip=t

### sftp

Allows you to browse the file system on the SFTP server or transmit the input records to the file.

#### Syntax

sftp PROFILE SUBCOMMAND [OPTIONS] PATH

Required Parameter

**PROFILE**

SFTP connect profile. You can configure the profile in the web console.

**SUBCOMMAND**

Command to be executed in the sftp session: ls, cat, put

* ls: Lists the files in the path specified by PATH on the server.
* cat: Loads the content of the file in the path specified by PATH from the server and assigns it in the **line** field.
* put: Transmits the records ​​of the fields specified by the fields option to the SFTP server as a file. The file is created in the path specified by PATH.

**PATH**

Path to a directory or file. If you use a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. /var/log/httpd/\* ).

* When **SUBCOMMAND** is ls, you can enter either a directory or a file path.
* When **SUBCOMMAND** is cat, you can enter only the file path.
* When **SUBCOMMAND** is put, you can enter only the file path.

Optional Parameter

The options for each SUBCOMMAND are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Options | `cat` | `put` | `ls` |
| append | O | - | - |
| encoding | O | O | - |
| fields | - | O | - |
| format | O | O | - |
| limit | O | - | - |
| maxsession | - | O | - |
| multisession | - | O | - |
| offset | O | - | - |
| overwrite | - | O | - |
| partition | - | O | - |

**append=BOOL**

Option to enable appending data to the end of the file specified as PATH (default: f).

* t: Appends the field records to the end of the file specified as PATH. If the file does not exist, the file is created. You cannot set this option to t when the overwrite=t.
* f: NOT append the field records to the end of the file specified as PATH. The query fails if the file exists.

When 'append=t', always keep the list order of the 'fields' option the same so that data can be consistent.

**encoding=CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**fields=FIELD,...**

Fields to be transmitted to the FTP server (default: line). Use comma (,) without any leading or trailing whitespaces as a separator. If there is no line field or the specified field is empty, it is replaced with a hyphen symbol (-) in the output to indicate the field is empty.

**format=FORMAT**

File format (csv, json, tsv, default: plain text).

* csv or tsv
* When **SUBCOMMAND** is cat, the first line is considered a regular record. Field name (column header) is assigned in the form columnN (N is a number starting from 0)
* When **SUBCOMMAND** is put, field names (column header) are assigned with the field names specified by the fields option.
* json
* When **SUBCOMMAND** is cat, it parses the file into the records of key-value pairs line by line. The keys are used as field names, and the values are used as field values.
* When **SUBCOMMAND** is put, it transmits the records consisting of the key-value pairs of the fields specified by the fields option. If the fields option is not specified, records consisting of all field values are transmitted.
* Not specified (plain text)
* When **SUBCOMMAND** is cat, it loades the values to the **line** field line by line.
* When **SUBCOMMAND** is put, it transmits the file in a text format. Values are separated by tab characters in plain text, and empty values (nulls) are replaced with hyphens (-).

**limit=INT**

Number of records to be output when importing files from the SFTP server (default: unlimited).

**maxsession=INT**

Maximum number of sessions when multisession=t (default: 1). If you specify this option without checking whether to use multi-session, the query fails. No matter how large the number is, as many sessions are opened as the number of MaxSessions specified in the sshd\_config file.

**multisession=BOOL**

Option to enable multi-session (default: f). Enable this option after testing because it takes longer to open additional sessions and may result in lower performance than not using it.

* t: Enables multi-session.
* f: Disables multi-session.

**offset=INT**

Number of rows you want to skip when importing files from the SFTP server (default: 0).

**overwrite=BOOL**

Option to enable overwriting the file specified as PATH, if it exists (default: f).

* t: Overwrites the file specified as PATH, if it exists. You cannot set this option to t when the append=t.
* f: NOT overwrite the file specified as PATH, if it exists. The query fails if the file exists.

**partition=BOOL**

Option to enable macro in the PATH (default: f).

* t: Enables macro.
* f: Disables macro.

You can specify PATH to change the directory and file path over time using a macro when partition=t. The available macros are {logtime:FMT} and {now:FMT}. For input examples, refer to Usage #6.

* {logtime:FMT}: Names the directory or file based on the log occurrence time.
* {now:FMT}: Names the directory or file based on the current time.

If you specify a partition option and do not use a macro on the path, the query fails.

#### Usage

Retrieve remote directory files by accessing SSH with an srv profile.

sftp srv ls /

Each query result field has the following meanings:

* **type** (string): dir when it is a directory, file when it is a file
* **is\_link** (boolean): Whether it is a symbolic link
* **name** (string): File name
* **file\_size** (integer): File size, 0 when it is a directory
* **modified\_at** (date): Last modified time
* **uid** (integer): Owner ID
* **gid** (integer): Owned group ID
* **perms** (string): File permission information

Read the first 5 rows of the /logpresso.sh file by accessing the srv profile.

sftp srv cat limit=5 /logpresso.sh

Output only UnloadedClassCount of LoadedClassCount among the JMX class loading logs to the /tmp/class.txt file.

table classloading | sftp srv put fields=UnloadedClassCount,LoadedClassCount /tmp/class.txt

Output the JMX class loading log to the /tmp/class.json file.

table classloading | sftp srv put format=json /tmp/class.json

Output LoadedClassCount, UnloadedClassCount, and TotalLoadedClassCount among the JMX class loading logs to the /tmp/class.csv file.

table classloading | sftp srv put format=csv fields=LoadedClassCount,UnloadedClassCount,TotalLoadedClassCount /tmp/class.csv

Output the LoadedClassCount, UnloadedClassCount, and TotalLoadedClassCount items among the JMX class loading logs to a JSON file. As the file is stored, the year, month and day based on the log time are used as the name of the directory (yyyy/MM/dd) and the hour and minute based on the current time are used as the name of the file (HHmm)

table classloading | sftp srv put format=json partition=t fields=LoadedClassCount,UnloadedClassCount,TotalLoadedClassCount {logtime:/yyyy/MM/dd/}{now:HHmm}.txt

### wget

Receives web resources through HTTP communication, or assigns the received data to the **line** field and the HTTP code of the server to the **\_wget\_code** field.

#### Syntax

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"]

Optional Parameter

**auth="ID:PASSWD"**

Basic HTTP authentication information for HTTP access. For details, refer to the following link: <https://datatracker.ietf.org/doc/html/rfc7617>

**body=FIELD**

Fields to be used as the HTML body. Use with method=post or method=put.

**encoding=CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

**format=FORMAT**

Message format from form, json, and xml (default: form). This can be used for REST API call.

**header=FIELD\_MAP\_TYPE**

Map type field consisting of a key-value as an HTTP header. It transmits only values that key and value are both string types. You can also specify a key-value map using the [dict()](https://docs.logpresso.com/en/query/dict-function) function. See usage #2.

**method=HTTP\_METHOD**

HTTP method from get, post, put, delete (default: get).

The post method has the following characteristics.

* It transmits the key-value of the input record in the form of x-www-form-urlencoded using URL encoding.
* It cannot be used with the url option, so use the header option to provide the HTTP header directly.

**selector="CSS\_SELECTOR"**

Elements to be selected from the HTML DOM tree. The same syntax is used when defining a selector in CSS.

**timeout=INT**

HTTP connection timeout time in seconds (default: 30).

**url="SITE\_URL"**

Web service URL to connect to. The command transmits the HTTP request to the specified web address and receives a HTTP response.

#### Usage

Load the RSS feed title.

wget url="https://logpresso.com/feed/" selector="item title" | explode elements | eval title = valueof(elements, "own\_text") | fields title

Check IP reputations from AbuseIPDB.

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

Report IP addresses to AbuseIPDB.

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

# Sonar Commands

## Events

### alert

Creates a Sonar event using the input record.

#### Syntax

alert

#### Description

This command is available only on the control node and can only be used by the cluster administrator. Basically, you can use it by setting up the alert command in a stream query that receives the event to be transmitted to the control node after real-time rule detection from each data node.

If a duplicate event is received, it may be removed due to the event deduplication setting of the real-time scenario. In addition, a ticket may be created or merged into an existing ticket depending on the real-time scenario settings. You can retrieve the created event in the event menu.

The input record must meet the following specifications:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Required | Type | Description |
| \_logger | Yes | 32-bit integer | Logger ID identifier |
| \_rule | Yes | 32-bit integer | Real-time scenario ID identifier |
| \_time | No | Date/Time | Time at which the original event occurred. If there is no value or the type does not match, it is treated as the time at which the input is made. |
| emp\_key | No | String | Employee number |
| emp\_name | No | String | Employee name |
| host\_ip | No | IP Address | Host IP address |
| src\_ip | No | IP Address | Source IP address |
| src\_country | No | String | Source ISO country code |
| src\_port | No | 32-bit integer | Source port number |
| dst\_ip | No | IP Address | Destination IP address |
| dst\_country | No | String | Destination ISO country code |
| dst\_port | No | 32-bit integer | Destination port number |
| protocol | No | String | Protocol |
| action | No | String | Response method |

### event

Retrieves events based on scenarios provided in Logpresso Sonar.

#### Syntax

event [duration=INT{mon|w|d|h|m|s}] [from=yyyyMMddHHmmss] [to=yyyyMMddHHmmss] [order=STR] [raw=BOOL]

Parameters

If you do not use duration, from or to, all events are searched.

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

Time range to search the previous data based on the current time. You can specify the time in units of mon (month), w (week), d (day), h (hour), m (minute), and s (second). For example, 10s refers to "the last 10 seconds" based on the time the query is executed. This option cannot be used with from or to.

**from=yyyyMMddHHmmss**

Start date and time of the period to search in the form of yyyyMMddHHmmss. The time period for the search includes the specified time point. If you provide only the first part, the remaining digits are recognized as 0. For example, if you provide 20130605, it is recognized as 20130605000000 (June 5, 2013, 00:00:00).to=yyyyMMddHHmmss

**to=yyyyMMddHHmmss**

End date and time of the period to serch in the form of yyyyMMddHHmmss. The time period for the search does NOT include the specified time point.

**order=STR**

Sorting order of the records (default: desc).

* asc: Sorts in ascending order, the oldest at the top.
* desc: Sortes in descending order, the latest records at the top.

**raw=BOOL**

Whether to retrieve the original event log (default: f).

* t: Retrieves event information as the original event log.
* f: Retrieves only normalized event informations.

#### Description

When raw=f, this query command retrieves only normalized event information. When raw=t, it retrieves event information as the original. A single event can be mapped with multiple original event logs so the number of normalized event may differ from the number of the original event log. Event information on the Ticket page is the same of the result of running this query command with raw=t.

The output fields of when running the event query command vary depending on the fields of the original evnet log because each event log has different default fields, or MariaDB database columnm. The **\_time** field of original log is converted to **\_log\_time** field.

#### Usage

Retrieve events occurred from May 23, 2023 00:00:00 to May 23, 2023 23:59:59.

event from=20230523 to=20230524

## Assets

### sonar-set-ip-address

Synchronizes specified field values from an input record to the internal IP asset database.

#### Syntax

sonar-set-ip-address fields=FIELDS [batchsize=INT] [company=GUID]

#### Parameters

**fields=FIELDS**

List of fields to be synchronized, separated by a comma with a following space. See the [Input Fields](https://docs.logpresso.com/en/query/sonar-set-ip-address-command) below for a detailed description.

Input Fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field | Name | Type | Length | Description |
| priority | Priority | 32-bit Integer |  | 3 (High), 2 (Medium), 1 (Low) |
| category\_name | Device type | String | 255 char. | Uncategorized if names are not matched |
| hostname | Host name | String | 255 char. | Computer name or host name |
| workgroup | Work group | String | 255 char. | NT domain or work group |
| emp\_key | Admin(MAIN) employee key | String | 255 char. | Input ignored if keys not matched |
| emp\_key2 | Admin(SUB) employee key | String | 255 char. | Input ignored if keys not matched |
| description | Description | String | 2000 char. |  |
| os\_name | OS | String | 50 char. |  |
| os\_ver | OS version | String | 20 char. |  |
| mac | MAC address | String | 20 char. |  |
| location | Install location | String | 255 char. |  |
| installed | Install date | Date |  |  |
| ext0 | Extended field #0 | String | 255 char. |  |
| ext1 | Extended field #1 | String | 255 char. |  |
| ext2 | Extended field #2 | String | 255 char. |  |
| ext3 | Extended field #3 | String | 255 char. |  |
| ext4 | Extended field #4 | String | 255 char. |  |
| ext5 | Extended field #5 | String | 255 char. |  |
| ext6 | Extended field #6 | String | 255 char. |  |
| ext7 | Extended field #7 | String | 255 char. |  |
| ext8 | Extended field #8 | String | 255 char. |  |
| ext9 | Extended field #9 | String | 255 char. |  |

* category\_name: The device type is labeled as 'Uncategorized' if the device type value from the input record does not match the device type value in the database.
* emp\_key, emp\_key2: The input is ignored if the employee key value in the input record does not match the employee key value in the database.
* You can customize the IP asset database by applying global setting on sonar as follows: logpresso> sonar.setGlobalOption ip\_custom\_fields "0=Manufacturer,1=Model name"

**batchsize=INT**

Number of records to process per commit. The valid range for INT is from 1 to 5000. Specifying this parameter is particulaly useful when processing a large amount of IP address data, as it improves by reducing the number of transactions. However, if an error occurs during synchronization, the process fails for the entire batch.

**company=COMPANY**

Company (tenant) GUID identifier. If not specified, the GUID of the the company assigned to the user account is used as default. When running a query in a system account, you must specify the tenant GUID using this option.

#### Description

This command matches the ip field value from the input record to the IP values in the database. If a match is found, the corresponding object in the database is updated for the field values specified by the 'fields' option. If no match is found, a new entry is added to the database.

Input record must contain a valid IP address value in the ip field. The ip value can be either a string or an IP address. If any value of other type is provided, the synchronization fails and an invalid ip error code is generated to the \_error field. When the ip value is null, an ip is null error code is output to the \_error field.

When the fields option is used, any field that does not exist in the input record will be assigned a null value in the database. On the other hand, When the fields option is not specified, the field values will not be synced, even if the input record contains fields with the same names.

Output Fields

This query command returns all fields from the input record as is. If any error occurs, this query command returns an \_error field in addition to the fields from input record.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_error | String | Error description |

#### Usage

Sync the Genian NAC asset data to the internal IP asset database.

The following example is using the app-extended command [genian-nac-nodes](https://logpresso.store/ko/apps/genian-nac/commands/genian-nac-nodes). Users can synchronize the entire Genian NAC IP asset information to the Logpresso Sonar IP asset database.

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"

## Datasets

### dataset

Loads the dataset corresponding to the specified GUID. Queries can be executed only by the cluster administrator, the account with company administrative privileges, and the account that owns the corresponding dataset.

#### Syntax

dataset guid=DATASET\_GUID

**guid=DATASET\_GUID**

Identifier generated upon dataset creation.

#### Usage

Load a specific dataset.

dataset guid=ac7c717e-86c3-482b-a08f-7b4a572cec79

## Threat Intelligence

### matchfeed

Filters the input records using the threat intelligence feed from the Logpresso CTI.

#### Syntax

matchfeed name=FEED\_ID|type=TYPE fields=FIELD,... [invert=BOOL]

Required Parameter

**name=FEED\_ID**

The identifier of the threat intelligence feed to match against the field record specified by the fields option (default: none). The name=FEED\_ID and type=TYPE options cannot be used at the same time. Use either one.

See the following table for available identifiers. In addition, you can use the feeds provided by apps installed on Lopresso Sonar.

|  |  |  |
| --- | --- | --- |
| FEED\_ID | Type | Description |
| otx | ip | Real-time IP address reputation feed in the format of OTX (Open Threat Exchange) |
| tor | ip | Tor exit node IP address information feed |
| mdl\_domain | domain | Malicious domain name (e.g. C&C domain) feed |
| mdl\_ip | ip | Malicious domain name (e.g. C&C IP address) feed |
| abusech | domain | Malicious domain name (e.g. C&C domain) feed provided by abuse.ch |
| malc0de | md5 | Malware database provided by malc0de.com |

**type=TYPE**

The type of value to match against the threat intelligence feed. Valid values are domain, email, ip, md5, sha256, and url. The type option matches against all threat intelligence feeds with that type information. The name=FEED\_ID and type=TYPE options cannot be used at the same time. Use either one.

* domain: Domain name
* email: Email address
* ip: IP address
* md5: MD5 hash of the binary file
* sha256: SHA256 hash of the binary file
* url: URL

**fields=FIELD,...**

Fields to match values against threat intelligence feeds. Use comma (,) without any leading or trailing whitespaces as a separator.

Optional Parameter

**invert=BOOL**

Option to invert the result of matching the value specified by the fields option against the threat intelligence feed (default: f)

* t: Returns records that do not contain the value specified by fields in the matching result.
* f: Returns records that contain the value specified by fields in the matching result.

#### Description

Refer to the following table for the fields to be returned after executing the command and the feed identifiers.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| feed\_name | String | Threat intelligence feed identifier |
| feed\_field | String | The name of the field where threat information was found |
| feed\_invert | Boolean | The value of the invert option |

### node-feed

Loads the threat intelligence data synchronized with the control node in the data node. This command is available only in the data node.

#### Syntax

node-feed name=FEED\_ID

Required Parameter

**name=FEED\_ID**

Identifier of the threat intelligence feed to query for synchronization. The name=FEED\_ID and type=TYPE options cannot be used at the same time. Use either one.

See the following table for available identifiers.

|  |  |  |
| --- | --- | --- |
| FEED\_ID | Type | Description |
| otx | IP address | Real-time IP address reputation feed in the format of OTX (Open Threat Exchange) |
| tor | IP address | Tor exit node IP address information feed |
| mdl\_domain | Domain | Malicious domain name (e.g. C&C domain) feed |
| mdl\_ip | IP address | Malicious domain name (e.g. C&C IP address) feed |
| abusech | Domain | Malicious domain name (e.g. C&C domain) feed provided by abuse.ch |
| malc0de | MD5 | Malware database provided by malc0de.com |

#### Usage

Look up OTX feeds synchronized in the data node.

node-feed name=otx

Look up malware IP lists synchronized in the data node.

node-feed name=mdl\_ip

## Behavior Profiles

### behavior

Loads the latest data created according to the behavior profile settings. This command is only available for the control node where the behavior profile data is located.

#### Syntax

behavior [OPTIONS] guid=PROFILE\_GUID

Required Parameter

**guid=PROFILE\_GUID**

GUID of behavior profile

Optional Parameter

**from=yyyyMMddHHmmss**

Start date and time of the search period in the form of yyyyMMddHHmmss (default: none). The time period for the search includes the specified time point. If you provide only the first part, the command recognizes the remaining digits as 0. For example, if you provide 20130605, the command recognizes it as 20130605000000 (June 5, 2013, 00:00:00).

**to=yyyyMMddHHmmss**

End date and time of the search period in the form of yyyyMMddHHmmss (default: none). The time period for the search does not include the specified time point. The input format is the same as from.

#### Description

The behavior command loads the latest data created according to the behavior profile settings. You can use it for ad-hoc analysis or for the correlation by joining when detecting batch scenarios. A read lock is set on the behavior profile specified as the GUID while the command is being executed.

### matchbehavior

Matches a behavior profile based on the key fields set in the behavior profile and adds the value field of the searched record to the output record.

#### Syntax

matchbehavior [invert=BOOL] [verify=BOOL] guid=PROFILE\_GUID

Required Parameter

**guid=PROFILE\_GUID**

GUID of behavior profile.

Optional Parameter

**invert=BOOL**

boolean option to output the value field (default: f)

* t: Outputs the records only if the reference key is not included in the behavior profile.
* f: Outputs the records only if the reference key is included in the behavior profile.

**verify=BOOL**

Boolean option to activate the validation for the behavior profile object at the query parsing stage (default: t).

* t: validates the behavior profile object at the query parsing stage.
* f: NOT validate the behavior profile object at the query parsing stage. This option prevents the system from issuing syntax errors at the policy synchronization stage.

#### Description

The matchbehavior command matches a behavior profile based on the key fields set in the behavior profile and then adds the value field of the searched record to the output record. Only string or IP address types are allowed in key fields in the behavior profile. Other types are considered match failures. If the invert option is activated, the command returns output only if the behavior profile match based on the key field fails.

For the output fields, refer to the following table.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| behavior\_guid | String | Behavior profile GUID |
| behavior\_invert | Boolean | invert option value |

### node-behavior

Loads behavior profile data synchronized with the control node in the data node. This command is available only in the data node.

#### Syntax

node-behavior [guid=PROFILE\_GUID]

Optional Parameter

**guid=PROFILE\_GUID**

Behavior profile GUID. If you do not specify an identifier, the command loads a list of synchronized behavior profiles.

If you do not specify the behavior profile GUID, the command returns the following values as shown in the table.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| id | Integer | Integer identifier |
| guid | String | Behavior profile GUID |
| name | String | Behavior profile name |
| description | String | Behavior profile description |
| row | Integer | Number of items in the behavior profile |
| curr\_ver | Integer | Current version number |
| company\_guid | String | Company GUID |
| company\_name | String | Company name |
| schedule | String | Creation cycle of behavior profile data (CRON format) |
| key\_fields | Array | List of behavior profile key fields (name, type) |
| query | String | Behavior profile recreation query |
| created | String | Creation time of the behavior profile |
| updated | String | Last modification time of the behavior profile |

#### Usage

Load the list of behavior profiles synchronized on the data node.

node-behavior

Load the specific behavior profile data synchronized on the data node.

node-behavior guid=c0a8c07f-34e3-48ca-a91c-5bb35684ae79

## Address Groups

### matchblackip

Filters the input records using the given IP blacklist.

#### Syntax

matchblackip [invert=BOOL] [verify=BOOL] fields=TARGET\_FIELD guid=BLACKLIST\_GUID

Required Parameter

**guid=BLACKLIST\_GUID**

IP blacklist GUID

**fields=TARGET\_FIELD**

Fields to match values against IP blacklist. Use comma (,) without any leading or trailing whitespaces as a separator.

**invert=BOOL**

Option to invert the result of matching the value specified by the fields option against the IP blacklist (default: f).

* t: Returns records that do not contain the value specified by fields in the matching result.
* f: Returns records that contain the value specified by fields in the matching result.

**verify=BOOL**

Option to activate the validation for the IP blacklist at the query parsing stage (default: t).

* t: validates the IP blacklist at the query parsing stage.
* f: NOT validate the IP blacklist at the query parsing stage. This option prevents the system from issuing syntax errors at the policy synchronization stage.

#### Description

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| blackip\_guid | String | IP blacklist GUID |
| blackip\_name | String | IP blacklist name |
| blackip\_field | String | Black IP discovery field name |
| blackip\_invert | Boolean | invert option value |

### node-ip-blacklist

Loads the IP blacklist item of the identifier synchronized with the control node in the data node. This command is available only in the data node.

#### Syntax

node-ip-blacklist [guid=BLACKLIST\_GUID]

Optional Parameter

**guid=BLACKLIST\_GUID**

IP blacklist GUID. If you specify an identifier, the command loads the IP blacklist item of the corresponding identifier synchronized with the control node in the data node. If you do not, the command loads a synchronized IP blacklists.

#### Description

The output fields of when the IP blacklist GUID is specified are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| ip | IP address | IP address in the corresponding blacklist |
| description | String | Reason why the IP is blacklisted |

The output fields of when the IP blacklist GUID is not specified are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| id | Integer | Integer identifier |
| guid | String | IP blacklist GUID |
| name | String | IP blacklist name |
| description | String | IP blacklist description |
| count | Integer | Number of IP addresses in the IP blacklist |
| version | Integer | IP blacklist version |
| company\_guid | String | Company GUID |
| company\_name | String | Company name |
| created | String | Createion time of the IP blacklist |
| updated | String | Last modification time of the IP blacklist |

#### Usage

Load the list of IP blacklists synchronized on the data node.

node-ip-blacklist

Load the specific IP blacklist items synchronized on the data node.

node-ip-blacklist guid=efd0c9cf-8582-4d5a-938d-9bb6a990579c

## Subnet Groups

### matchnet

Checks whether the IP address value of the field is included in the specified IP subnet and returns the result.

#### Syntax

matchnet [OPTIONS] field=TARGET\_FIELD guid=NET\_GUID [tag=BOOL]

tag option is supported since 4.0.2312.0 version.

Required Parameter

**field=FIELD**

Target field name. The value of the target field must be an IPv4 address, a string, or a 32-bit integer type. Any other key value is considered a search failure. In the case of string and 32-bit integer type values, the command attempts to search only when these are converted to valid IPv4 addresses.

**guid=NET\_GUID**

IP subnet GUID

Optional Parameter

**invert=BOOL**

Option to invert the result of matching the value specified by the field option against the IP subnet (default: f)

* t: Returns records that do not contain the value specified by the field option in the matching result.
* f: Returns records that contain the value specified by the field option in the matching result.

**verify=BOOL**

Option to activate the validation for the IP subnet at the query parsing stage (default: t).

* t: validates the IP subnet at the query parsing stage.
* f: Does not validate the IP subnet at the query parsing stage. This option prevents the system from issuing syntax errors at the policy synchronization stage.

**tag=BOOL**

Option to output information about the matched IP subnet (default: f).

* t: Outputs a map including id, start\_ip, end\_ip, cidr attributes in the \_matchnet\_result field.
* f: Does not output \_matchnet\_result field.

### node-subnet-group

Loads IP address range items synchronized with the control node in the data node. This command is only available in the data node.

#### Syntax

node-subnet-group [guid=NET\_GUID]

Optional Parameter

**guid=NET\_GUID**

IP subnet GUID. If you specify the GUID, the result displays the IP subnet corresponding to the specified GUID. If you do not specify the GUID, the command displays a list of all IP subnets synchronized to the data nodes.

#### Description

The output fields of when the GUID is specified are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| start\_ip | IP address | IPv4 address (when the object type is single), network address (when the object type is CIDR), and the first IP address of the IP subnet (when the object type is range). |
| end\_ip | IP address | the last IP address of the IP subnet when object type is range. |
| cidr | Integer | CIDR subnet mask between 0 and 32 |
| description | String | Description |

The output fields of when the GUID is not specified are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| id | Integer | Integer indentifier |
| guid | String | IP subnet GUID |
| name | String | IP subnet name |
| description | String | IP subnet description |
| subnet\_count | Integer | Number of items in the IP subnet |
| company\_guid | String | Company GUID |
| company\_name | String | Company name |
| user\_name | String | Account name |
| user\_guid | String | Account GUID |

#### Usage

Load the list of IP address ranges synchronized on the data node.

node-subnet-group

Load specific IP address range items synchronized on the data node.

node-subnet-group guid=96f342ee-0aa2-4234-ac7c-37a50c38b7bc

## Port Groups

### matchport

Checks whether a specified port group contains the value of the port field and outputs the result.

#### Syntax

matchport [invert=BOOL] [port=PORT\_FIELD] [protocol=PROTO\_FIELD] [verify=BOOL] guid=PORTS\_GUID

Required Parameter

**guid=PORT\_GUID**

Port group GUID

Optional Parameter

**invert=BOOL**

Option to invert the output condition (default: f)

* t: returns the result only when the value of the port field is not included in the port group specified with guid.
* f: returns the result only when the value of the port field is included in the port group specified with guid.

**port=PORT\_FIELD**

Name of the port field (default: port)

**protocol=PROTO\_FIELD**

Name of the protocol field (default: protocol)

**verify=BOOL**

Option to activate the validation for the port group at the query parsing stage (default: t).

* t: validates the port group at the query parsing stage.
* f: NOT validate the port group at the query parsing stage. This option prevents the system from issuing syntax errors at the policy synchronization stage.

### node-port-group

Loads port group items synchronized with the control node in the data node. The command is only available in the data node.

#### Syntax

node-port-group [guid=PORT\_GUID]

Optional Parameter

**guid=PORT\_GUID**

Port group GUID. If you specify the GUID, the command displays the port group information corresponding to the specified GUID. If you do not, the command displays a list of all port groups synchronized to the data nodes.

#### Description

The output fields of when the port group GUID is specified are as follows:

|  |  |  |
| --- | --- | --- |
| Fields | Type | Description |
| protocol | String | Protocol TCP or UDP |
| start | Integer | Port range start value |
| end | Integer | Port range end value |
| description | String | Description |

The output fields of when the port group GUID is not specified are as follows:

|  |  |  |
| --- | --- | --- |
| Fields | Type | Description |
| id | Integer | Integer identifier |
| guid | String | Port group GUID |
| name | String | Port group name |
| description | String | Port group description |
| port\_count | Integer | Number of items in the port group |
| company\_guid | String | Company GUID |
| company\_name | String | Company name |
| user\_name | String | Account name |
| user\_guid | String | Account GUID |
| created\_at | String | Creation time of the port group |
| updated\_at | String | Last modification time of the port group |

#### Usage

Load the list of port groups synchronized on the data node.

node-port-group

Load specific port group items synchronized on the data node.

node-port-group guid=2da1fa00-da63-4fb5-a443-46260c555697

## Pattern Groups

### matchsig

Checks whether the string value of the field is included in the specified pattern group and outputs the result.

#### Syntax

matchsig [invert=BOOL] [verify=BOOL] guid=SIG\_GUID field=FIELD

Required Parameter

**guid=SIG\_GUID**

Pattern group GUID

**field=FIELD**

Target field name. The value of the target field must be a string, and any other key value is considered a search failure.

Optional Parameter

**invert=BOOL**

Option to invert the result of matching the value specified by the field option against the parttern group (default: f).

* t: Returns the result only if the value of the target field is not included in the pattern group.
* f: Returns the result if the string value of the target field matches at least one of the patterns in the specified pattern group.

**verify=BOOL**

Option to activate the validation for the port group at the query parsing stage (default: t).

* t: validates the pattern group at the query parsing stage.
* f: NOT validate the pattern group at the query parsing stage. This option prevents the system from issuing syntax errors at the policy synchronization stage.

#### Description

The pattern group uses the Aho-Corasick algorithm to test thousands of keywords or more simultaneously, like network intrusion detection devices. By first matching all keywords belonging to the pattern group against the input string at once and then executing the filter expression of the selected patterns sequentially, the command eventually returns the rule name that matches the pattern.

**Example of patterns**

|  |  |  |
| --- | --- | --- |
| expr (required) | expr2 (optional) | rule (required) |
| Keyword pattern: Primary high-speed detection | Boolean expression: Secondary filtering | Pattern name |
| "addextendedproc" and "xp\_cmdshell" |  | xp\_cmdshell |
| "REMOTE\_ADDR" and ("fputs" or "fwrite") | path == "lib.php" | zb now\_connect |

For example, if you set pattern "REMOTE\_ADDR" and ("fputs" or "fwrite"), filter expression path == "lib.php", rule zb now connect, the command checks whether the REMOTE\_ADDR string and the fputs or fwirte string are searched at the same time in the target field value, and then checks whether the path field value matches the lib.php string.

### node-pattern-group

Retreives pattern group items synchronized with the control node in the data node. This command is only available in the data node.

#### Syntax

node-pattern-group [guid=SIG\_GUID]

Optional Parameter

**guid=SIG\_GUID**

GUID of the pattern group. If you specify the GUID, the command displays the pattern group information corresponding to the specified GUID. If you do not specify the GUID, the command displays a list of all pattern groups synchronized to the data nodes.

#### Description

The output fields of when the pattern group GUID is specified are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| expr | String | A boolean combination of keywords for Aho-Corasick multi-pattern matching |
| expr2 | String | Secondary inspection expression after expr matching |
| rule | String | Pattern name (name to be tagged in the output when matching) |

The output fields of when the pattern group GUID is not specified are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| id | Integer | Integer identifier |
| guid | String | Pattern group GUID |
| name | String | Pattern group name |
| description | String | Pattern group description |
| pattern\_count | Integer | Number of items in the pattern group |
| company\_guid | String | Company GUID |
| company\_name | String | Company name |
| user\_name | String | Account name |
| user\_guid | String | Account GUID |

#### Usage

Load the list of pattern groups synchronized on the data node.

node-pattern-group

Retrieve specific pattern group items synchronized on the data node.

node-pattern-group guid=b5ce2e95-67b9-4d64-8f6e-2746264a58d2

# Forensic Commands

## Windows Artifacts

### evtx-file

Retrieves information such as the event channel, event provider, event ID, event operation and the like from an EVTX Windows event log file.

#### Syntax

evtx-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the Windows event log file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. D:\data\evtx\*.evtx). If you provided the zippath option, input the EVTX file path in the zip file.

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding. Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath**

Path to the ZIP file.

#### Description

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Time at which the event occurred |
| computer | String | Computer name |
| channel | String | Event channel |
| provider | String | Event provider |
| event\_id | Integer | Event ID |
| task | Integer | Event task |
| level | Integer | Event level |
| record\_id | Integer | Recored ID |
| msg | String | Event message |
| event\_data | Map | Event data |

#### Usage

Retrieve information by providing the file path.

evtx-file D:\data\evtx\System.evtx

Retrieve information when the zippath option is provided.

evtx-file zippath=D:\data\evtx.zip evtx\System.evtx

Retrieve an event whose event provider is MySQL.

evtx-file D:\data\evtx\application.evtx | search provider=="MySQL"

Retrieve events that do not match the EVTX\_WHITE message pattern.

evtx-file D:\data\evtx\application.evtx | mpsearch msg [ lookuptable EVTX\_WHITE ] | search len(\_mp\_result) == 0

### hive-file

Retrieves information such as the account and group, security policy, OS information, USB device, and program usage history from a registry hive file.

#### Syntax

hive-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the registry hive file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. D:\data\registry\*). If you provided the zippath option, input the registry hive file path in the ZIP file. There are five registry hive files.

|  |  |  |  |
| --- | --- | --- | --- |
| File | Purpose | Registry Path | Extraction Information |
| SAM | Account and access records | HKEY\_LOCAL\_MACHINE\SAM | Account and group |
| SECURITY | Security policy and privilege | HKEY\_LOCAL\_MACHINE\Security | Security policy |
| SOFTWARE | Installation program | HKEY\_LOCAL\_MACHINE\Software | OS version, OS installation date, OS installation directory and owner account |
| SYSTEM | System settings | HKEY\_LOCAL\_MACHINE\System | Host name, time zone, system shutdown time, USB device and the like |
| NTUSER.DAT | User settings | HKEY\_USERS\.DEFAULT | List of files you opened |

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding. Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

Path to the ZIP file

#### Description

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| key | String | Subkey |
| type | String | Type |
| name | String | Registry name |
| value | Object | Registry data |
| last\_written | Date | Last written time |

#### Usage

Retrieve information by providing the file path.

hive-file D:\data\registry\SYSTEM

Retrieve information when the zippath option is provided.

hive-file zippath=D:\data\registry.zip registry\SYSTEM

Check the Windows OS information.

hive-file D:\data\registry\SOFTWARE | search key=="ROOT\\Microsoft\\Windows NT\\CurrentVersion"

### ntfs-logfile

Retrieves information such as file name, creation/modification/access time, and redo/undo operation type in a NTFS transaction log file. With this command, you can see the history of file creation, deletion and renaming.

#### Syntax

ntfs-logfile [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the NTFS log file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. D:\data\NTFS\*). If you provided the zippath option, input the NTFS file path in the ZIP file.

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding. Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

The path to the ZIP file.

#### Description

Output Fields

After running the ntfs-logfile command, the output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| access\_at | Date | Last access time |
| client\_data\_length | Integer | Record volume |
| created\_at | Date | Time of creation |
| file\_name | String | File name |
| flags | Integer | Flag |
| lsn | Integer | Log file sequence number |
| mft\_head | String | MFT header |
| mft\_link\_count | Integer | Number of entries referencing the corresponding file |
| mft\_lsn | Integer | MFT log file sequence number |
| modified\_at | Date | Last modification time |
| page | Integer | Page number |
| prev\_lsn | Integer | Previous log file sequence number |
| record\_type | Integer | Record type (2: checkpoint record, 1: other records) |
| redo\_len | Integer | Size of redo data |
| redo\_offset | Integer | Offset of redo data |
| redo\_op | String | Redo operation code |
| undo\_len | Integer | Size of undo data |
| undo\_offset | Integer | Offset of undo data |
| undo\_op | String | Undo operation code |

Redo\_op and Undo\_op Operation Codes

Operation Codes output to the redo\_op and undo\_op fields are as follows:

|  |  |
| --- | --- |
| `redo\_op`/`undo\_op` | Hex value |
| 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 |

#### Usage

Retrieve by providing a file path.

ntfs-logfile D:\data\NTFS\test\_LogFile

Retrieve when the zippath option is provided.

ntfs-logfile zippath=D:\data\NTFS.zip NTFS\test\_LogFile

Retrieve logs where the redo\_op is related to any initialize\_file\_record\_segment or delete operation.

ntfs-logfile D:\data\NTFS\test\_LogFile | sort lsn | search redo\_op == "initialize\_file\_record\_segment" or redo\_op == "\*delete\*"

### ntfs-mft

Retrieve information such as file path and name, file volume, disk allocation volume, file creation/modification/access time, and the existence of directories in an NTFS master file. Using the retrieved data, you can analyze the entire file and folder structure, extract deleted files or folders and browse alternate data stream (ADS) hidden information.

#### Syntax

ntfs-mft [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the NTFS MFT file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. D:\data\NTFS\*). If you provided the zippath option, input the NTFS MFT file path in the ZIP file.

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding. Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

Path to the ZIP file

#### Description

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| no | Integer | File number |
| file\_name | String | File name |
| file\_path | String | File path |
| file\_size | Integer | File size |
| alloc\_size | Integer | Allocated size |
| in\_use | Boolean | In-use flag |
| is\_dir | Boolean | Directory flag |
| link\_count | Integer | Number of hard links referencing the file |
| created\_at | Date | Creation time of the $FILE\_NAME attribute |
| modified\_at | Date | Last modification time of the $FILE\_NAME attribute |
| access\_at | Date | Last access time of the $FILE\_NAME attribute |
| mft\_modified\_at | Date | Last MFT modification time of the $FILE\_NAME attribute |
| std\_created\_at | Date | Creation time of the $STANDARD\_INFORMATION attribute |
| std\_modified\_at | Date | Last modification time of the $STANDARD\_INFORMATION attribute |
| std\_access\_at | Date | Last access time of the $STANDARD\_INFORMATION attribute |
| std\_mft\_modified\_at | Date | Last MFT modification time of the $STANDARD\_INFORMATION attribute |
| is\_readonly | Boolean | Read-only flag |
| is\_hidden | Boolean | Hidden flag |
| is\_system | Boolean | System flag |
| is\_archive | Boolean | Archive flag |
| is\_device | Boolean | Device flag |
| is\_normal | Boolean | Normal flag |
| is\_temp | Boolean | Temporary flag |
| is\_sparse | Boolean | Sparse file flag |
| is\_reparse | Boolean | Reparse point flag |
| is\_compressed | Boolean | Compression flag |
| is\_offline | Boolean | Offline flag |
| is\_indexed | Boolean | Index flag |
| is\_encrypted | Boolean | Encryption flag |
| lsn | Integer | Log sequence number |
| seq | Integer | Sequence |
| file\_ref | Integer | File reference |
| parent\_file\_ref | Integer | Parent file reference |
| parent\_no | Integer | Parent file number |

#### Usage

Retrieve information by providing the file path.

ntfs-mft D:\data\NTFS\test\_MFT

Retrieve information when the zippath option is provided.

ntfs-mft zippath=D:\data\NTFS.zip NTFS\test\_MFT

Load the list of deleted hidden files.

ntfs-mft D:\data\NTFS\test\_MFT | search not(in\_use) and not(is\_dir) and is\_hidden

### ntfs-usnjrnl

Retrieves operational information such as the time of the event occurrence, file path and name, and file creation/deletion. You can perform analyze the timeline by joining the retrieved data with the MFT file.

#### Syntax

ntfs-usnjrnl [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the USNJRNL journal file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. D:\data\NTFS\*). If you provided the zippath option, input the USNJRNL journal file path in the ZIP file.

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

Path to the ZIP file

#### Description

Output Fields

The output field is as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Time at which the event occurred |
| file\_name | String | File name |
| file\_no | Integer | File number |
| file\_ref | Integer | File reference |
| parent\_file\_no | Integer | Parent file number |
| parent\_file\_ref | Integer | Parent file reference |
| reason | List | Event behavior. Refer to Reason Flags. |
| usn | Integer | Update sequence number |

Reason Flags

For more information, see the document **USN\_RECORD\_V3 structure (winioctl.h)**: <https://docs.microsoft.com/en-us/windows/win32/api/winioctl/ns-winioctl-usn_record_v2>

|  |  |  |
| --- | --- | --- |
| Reason flag | Hex | Description |
| DATA\_OVERWRITE | 0x00000001 | Data has been overwritten on the default $DATA attribute. |
| DATA\_EXTEND | 0x00000002 | Data has been added to the default $DATA. |
| DATA\_TRUNCATION | 0x00000004 | Data has been truncated on the default $DATA attribute. |
| NAMED\_DATA\_OVERWRITE | 0x00000010 | Data has been overwritten on the named default $DATA attribute. |
| NAMED\_DATA\_EXTEND | 0x00000020 | Data has been added to the named default $DATA attribute. |
| NAMED\_DATA\_TRUNCATION | 0x00000040 | Data has been truncated on the named default data attribute. |
| FILE\_CREATE | 0x00000100 | The file or directory has been created for the first time. |
| FILE\_DELETE | 0x00000200 | The file or directory is deleted. |
| EA\_CHANGE | 0x00000400 | The extended attributes of the file or directory have changed. |
| SECURITY\_CHANGE | 0x00000800 | The access permission has been changed. |
| RENAME\_OLD\_NAME | 0x00001000 | The old name when the file or directory is renamed. |
| RENAME\_NEW\_NAME | 0x00002000 | The new name when the file or directory is renamed. |
| INDEXABLE\_CHANGE | 0x00004000 | The index status has been changed. |
| BASIC\_INFO\_CHANGE | 0x00008000 | One or more file/directory attributes or time stamps have been changed. |
| HARD\_LINK\_CHANGE | 0x00010000 | A hard link has been created or removed. |
| COMPRESSION\_CHANGE | 0x00020000 | The compression status has been changed (compressed or decompressed). |
| ENCRYPTION\_CHANGE | 0x00040000 | The encryption status has been changed (encrypted or decrypted). |
| OBJECT\_ID\_CHANGE | 0x00080000 | The object ID has been changed. |
| REPARSE\_POINT\_CHANGE | 0x00100000 | The reparse point has been changed. |
| STREAM\_CHANGE | 0x00200000 | An attribute of the named $DATA has been added, removed, or renamed. |
| TRANSACTED\_CHANGE | 0x00400000 | The given stream is modified through a committed TxF transaction. |
| INTEGRITY\_CHANGE | 0x00800000 | The integrity setting has been changed. |
| CLOSE | 0x80000000 | The file or directory is closed. |

#### Usage

Retrieve information by providing the file path.

ntfs-usnjrnl D:\data\NTFS\test\_UsnJrnl

Retrieve information when the zippath option is provided.

ntfs-usnjrnl zippath=D:\data\NTFS.zip NTFS\test\_UsnJrnl

Load the deletion history of executable files.

ntfs-usnjrnl D:\data\NTFS\test\_UsnJrnl | search file\_name == "\*.exe" and string(reason) == "\*DELETE\*"

Analyze the timeline by joining with the NTFS MFT file.

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

### reg-opensave-files

Loads the "information on the files recently opened or saved with the Windows Explorer common dialog box" stored in the registry file and information on the files you opened or saved through web browsers and applications. Using this command, you can see files recently opened or saved by the user.

#### Syntax

reg-opensave-files [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the registry file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. :\data\registry\*.DAT). If you provided the zippath option, input the DAT file path in the ZIP file.

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding. Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

Path to the ZIP file

#### Description

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_path | String | File path |
| file\_ext | String | File extension |
| file\_size | String | File volume |
| access\_at | Date | Last access time |
| created\_at | Date | Creation time |
| modified\_at | Date | Last modification time |
| mft\_entry\_index | Binary | MFT entry index |
| ntfs\_seq | Integer | NTFS sequence |
| last\_written | Date | Last written time |
| order | Integer | File order by extension |

#### Usage

Retrieve information by providing the file path.

reg-opensave-files D:\data\registry\NTUSER.DAT

Retrieve information when the zippath option is provided.

reg-opensave-files zippath=D:\data\registry.zip registry\NTUSER.DAT

Sort the order field by file extension.

reg-opensave-files D:\data\registry\NTUSER.DAT | sort file\_ext, order

### reg-recent-docs

Loads the "information on the files and folders recently opened or executed by the user with Windows Explorer" stored in the registry file. With the loaded data, you can check information on files and folders opened or executed by the user and whether documents and folders have been executed. You can also use it to identify the user's behavior.

#### Syntax

reg-recent-docs [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the registry file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. D:\data\registry\*.DAT). If you provided the zippath option, input the registry file path in the ZIP file.

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding. Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

Path to the ZIP file

#### Description

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_name | String | File name |
| file\_ext | String | File extension |
| last\_written | Date | Last written time |
| order | Integer | File order by extension |

#### Usage

Retrieve information by providing the file path.

reg-recent-docs D:\data\registry\NTUSER.DAT

Retrieve information when the zippath option is provided.

reg-recent-docs zippath=D:\data\registry.zip registry\NTUSER.DAT

Sort the order field by file extension.

reg-recent-docs D:\data\registry\NTUSER.DAT | sort file\_ext, order

### reg-shellbags

Loads the "information on the folder accessed by the user from local, network and removable storage devices" stored in the registry. Using the loaded data, you can check the time information of when a user accessed a specific folder, track evidence of deletion/overwriting of existing folders and track the MAC time for folder access through Explorer.

#### Syntax

reg-shellbags [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the registry file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. D:\data\registry\*). If you provided the zippath option, input the registry file path in the ZIP file.

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding. Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

Path to the ZIP file

#### Description

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_name | String | File name |
| file\_ext | String | File extension |
| last\_written | Date | Last written time |
| order | Integer | File order by extension |

#### Usage

Retrieve information by providing the file path.

reg-shellbags D:\data\registry\NTUSER.DAT

Retrieve information when the zippath option is provided.

reg-shellbags zippath=D:\data\registry.zip registry\NTUSER.DAT

Sort the order field by file extension.

reg-shellbags D:\data\registry\NTUSER.DAT | sort file\_ext, order

### reg-shim-cache

Loads information such as the path, volume, and last run time of all executable files using the **AppCompatCache** key (Path: HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache\AppCompatCache) data stored in the registry file. You can check the name, path, volume information, and last run time of executable files with the loaded data and use it to analyze infringement accidents.

#### Syntax

reg-shim-cache [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the registry file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. D:\data\registry\*). If you provided the zippath option, input the registry file path in the ZIP file.

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding. Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

Path to the ZIP file

#### Description

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_path | String | Executable file path |
| modified\_at | Date | Last modification time |

#### Usage

Retrieve information by providing the file path.

reg-shim-cache D:\data\registry\SYSTEM

Retrieve information when the zippath option is provided.

reg-shim-cache zippath=D:\data\registry.zip registry\SYSTEM

### reg-user-assists

Loads information such as "the list of programs recently executed, the last run time, and the execution count" stored in the registry file. Using the loaded data, you can check the name and list of recently executed applications and use the time and number of recently executed applications for analysis.

#### Syntax

reg-user-assists [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the registry file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. D:\data\registry\*). If you provided the zippath option, input the registry file path in the ZIP file.

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding. Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

Path to the ZIP file

#### Description

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| key | String | Executable file path |
| session\_num | Integer | Session number |
| exec\_count | Integer | Number of times of execution |
| focus\_time | Integer | Time activated |
| last\_execution | Date | Last run time |
| last\_written | Date | Last written time |

#### Usage

Retrieve information by providing the file path.

reg-user-assists D:\data\registry\NTUSER.DAT

Retrieve information when the zippath option is provided.

reg-user-assists zippath=D:\data\registry.zip registry\NTUSER.DAT

### zipfile-entries

Imports a list of compressed files and directories in the specified ZIP file.

#### Syntax

zipfile-entries ZIPFILE\_PATH

Required Parameter

**ZIPFILE\_PATH**

Path to the ZIP file. If you use a wildcard (\*) in the file name, you can retrieve all the files containing a specific string pattern in the file name at once. The Logpresso daemon must have the read permission to the file.

#### Description

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| zip\_path | String | Path to ZIP file |
| entry | String | ZIP file name |
| file\_size | Integer | File size |
| compressed\_size | Integer | Compressed size |
| modified\_at | Date | Last modification time |
| comment | String | Comment |

#### Usage

Load the list of files and directories in the entry.zip file under D:\data.

zipfile-entries D:\data\entry.zip

Load the list of files and directories of all ZIP files under D:\data.

zipfile-entries D:\data\\*.zip

Retrieve file entries with the .evtx extension in the entry.zip file.

zipfile-entries D:\data\entry.zip | search entry=="\*.evtx"

## Linux Artifacts

### linux-arp-entries

Retrieves the ARP cache from the /proc/net/arp file.

#### Syntax

linux-arp-entries

#### Description

After running the linux-arp-entries command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| type | String | Hardware type. In most cases, Ethernet(1). |
| ip | IP address | Associated IP address |
| mac | String | MAC address |
| iface | String | Network interface name |

#### Usage

Look up the ARP cache.

linux-arp-entries

### linux-connections

Looks up TCP/IP network connection information.

#### Syntax

linux-connections

#### Description

This command retrieves TCP/IP network connection information from the following files:

* /proc/net/icmp
* /proc/net/icmp6
* /proc/net/raw
* /proc/net/raw6
* /proc/net/tcp
* /proc/net/tcp6
* /proc/net/udp
* /proc/net/udp6

After running the linux-connections command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| pid | Integer | Process ID |
| rx\_queue | Integer | Rx queue length |
| tx\_queue | Integer | Tx queue length |
| protocol | String | Protocol |
| local\_ip | IP address | Local IP address |
| local\_port | Integer | Local port |
| remote\_ip | IP address | Remote IP address |
| remote\_port | Integer | Remote port |
| state | String | Status (LISTEN, ESTABLISHED, TIME\_WAIT, etc.) |

### linux-cron-jobs

List all scheduled cron jobs.

#### Syntax

linux-cron-jobs

#### Description

This command collects information about cron jobs from the following files:

* /var/cron/tabs/
* /var/spool/cron/
* /var/spool/cron/crontabs/

After running the linux-system-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| user | String | User ID |
| cron\_schedule | String | Schedule |
| cmd\_line | String | Command to run |

### linux-env

Lists the environment variables in Linux.

#### Syntax

linux-env

#### Description

After running the linux-env command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| name | String | Environment variable name |
| value | String | Value of environment variable |

### linux-failed-logins

Retrieves the log of failed login attempts from the /var/log/btmp file.

#### Syntax

linux-failed-logins [ignore-error=BOOL]

Optional Parameter

**ignore-error=BOOL**

Boolean option to handle error when the /var/log/btmp file cannot be read (default: f).

* t: Exits gracefully if an error occurs
* f: Fails if an error occurs

#### Description

After running the linux-system-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Last failed login time |
| src\_ip | IPv4 address | Remote IP address from which login was attempted |
| user | String | User ID |

### linux-hidden-files

Recursively lists the hidden files in the /tmp, /dev, and /home directories.

#### Syntax

linux-hidden-files

#### Description

After running the linux-hidden-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_path | String | File path |
| file\_name | String | File name |
| file\_type | String | File type |
| permissions | String | Permissions |
| file\_size | Long | File size |
| file\_ctime | Date | Last change time |
| file\_mtime | Date | Last modification time |
| file\_atime | Date | Last access time |
| owner\_read | Boolean | Whether owner read permission is given |
| owner\_write | Boolean | Whether owner write permission is given |
| owner\_execute | Boolean | Whether owner execute permission is given |
| group\_read | Boolean | Whether group read permission is given |
| group\_write | Boolean | Whether group write permission is given |
| group\_execute | Boolean | Whether group execute permission is given |
| others\_read | Boolean | Whether others read permission is given |
| others\_write | Boolean | Whether others write permission is given |
| others\_execute | Boolean | Whether others execute permission is given |

### linux-logins

Extracts login and logout histories of all users from the /var/log/wtmp log.

#### Syntax

linux-logins

#### Description

After running the linux-logins command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Login time |
| src\_ip | IPv4 address | Access IP address |
| host | String | Hostname of client (IP or domain name) |
| pid | Integer | Process ID |
| user | String | User ID |
| tty | String | Terminal |
| login\_time | Date | Login time |
| logout\_time | Date | Logout time |

### linux-network-interfaces

Retrieves network interface settings and statistics information.

#### Syntax

linux-network-interfaces

#### Description

This command collects network interface settings and statistics information based on the interface flags in files under /sys/class/net/, each statistic value in files under the statistics directory, and the execution result of the ip address show command.

After running the linux-network-interfaces command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| iface | String | Interface name |
| status | String | Interface status |
| mtu | Integer | Maximum transfer unit |
| ip\_addr | IPv4 address | IPv4 address |
| netmask | IPv4 address | IPv4 netmask |
| ip\_addr6 | IPv6 address | IPv6 address |
| prefix\_len | Long | IPv6 prefix |
| rx\_pkts | Long | Number of packets received |
| rx\_bytes | Long | Received data in bytes |
| rx\_errors | Long | Number of packet received with errors, such as checksum mismatch |
| rx\_drops | Long | Number of packet received discarded for reasons such as destination mismatch |
| rx\_overruns | Long | Number of packet received but discarded for reasons such as receive queue overrun |
| tx\_pkts | Long | Number of packets tranferred |
| tx\_bytes | Long | Transferred data in bytes |
| tx\_errors | Long | Number of packet not transmitted due to an error |
| tx\_drops | Long | Number of packet not transmitted for reasons such as destination mismatch |
| tx\_overruns | Long | Number of packet not transmitted for reasons such as transfer queue overrun |
| tx\_carrier | Long | Number of disconnections |
| tx\_collisions | Long | Packets with collision |

### linux-no-owner-files

Lists files that do not have an owner.

#### Syntax

linux-no-owner-files

#### Description

After running the linux-no-owner-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_path | String | File path |
| file\_name | String | File name |
| file\_type | String | File type |
| permissions | String | Permissions |
| file\_size | Long | File size |
| file\_ctime | Date | Last change time |
| file\_mtime | Date | Last modification time |
| file\_atime | Date | Last access time |
| owner\_read | Boolean | Whether owner read permission is given |
| owner\_write | Boolean | Whether owner write permission is given |
| owner\_execute | Boolean | Whether owner execute permission is given |
| group\_read | Boolean | Whether group read permission is given |
| group\_write | Boolean | Whether group write permission is given |
| group\_execute | Boolean | Whether group execute permission is given |
| others\_read | Boolean | Whether other read permission is given |
| others\_write | Boolean | Whether other write permission is given |
| others\_execute | Boolean | Whether other execute permission is given |

### linux-non-device-files

Lists non-device files in the /dev directory.

#### Syntax

linux-non-device-files

#### Description

After running the linux-non-device-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_path | String | File path |
| file\_name | String | File name |
| file\_type | String | File type |
| permissions | String | Permissions |
| file\_size | Long | File size |
| file\_ctime | Date | Last change time |
| file\_mtime | Date | Last modification time |
| file\_atime | Date | Last access time |
| owner\_read | Boolean | Whether owner read permission is given |
| owner\_write | Boolean | Whether owner write permission is given |
| owner\_execute | Boolean | Whether owner execute permission is given |
| group\_read | Boolean | Whether group read permission is given |
| group\_write | Boolean | Whether group write permission is given |
| group\_execute | Boolean | Whether group execute permission is given |
| others\_read | Boolean | Whether other read permission is given |
| others\_write | Boolean | Whether other write permission is given |
| others\_execute | Boolean | Whether other execute permission is given |

### linux-open-files

Collects the /proc/PID/fd file lists and returns the open file list by process in Linux.

#### Syntax

linux-open-files

#### Description

After running the linux-open-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| pid | Integer | Process ID |
| cmd\_line | String | Complete command line for the process |
| user | String | User ID |
| fd | Integer | File discriptor |
| type | String | File type (REG = regular file) |
| file\_size | Long | File size |
| target | String | File path or socket information |

### linux-partitions

Parses the fdisk -l command execution result to list all disk partitions in Linux.

#### Syntax

linux-partitions

#### Description

After running the linux-partitions command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| device | String | Storage device name |
| start | Long | Start number of sector |
| end | Long | End number of sector |
| sector\_count | Long | Total number of sector |
| total\_bytes | Long | Total partition size in bytes |
| type | String | Partition type |

### linux-pipes

Parse the find / -type p command execution result and lists the pipe files.

#### Syntax

linux-pipes

#### Description

After running the linux-pipes command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_path | String | File path |
| file\_name | String | File name |
| file\_type | String | File type |
| permissions | String | Permissions |
| file\_size | Long | File size |
| file\_ctime | Date | Last change time |
| file\_mtime | Date | Last modification time |
| file\_atime | Date | Last access time |
| owner\_read | Boolean | Whether owner read permission is given |
| owner\_write | Boolean | Whether owner write permission is given |
| owner\_execute | Boolean | Whether owner execute permission is given |
| group\_read | Boolean | Whether group read permission is given |
| group\_write | Boolean | Whether group write permission is given |
| group\_execute | Boolean | Whether group execute permission is given |
| others\_read | Boolean | Whether other read permission is given |
| others\_write | Boolean | Whether other write permission is given |
| others\_execute | Boolean | Whether other execute permission is given |

### linux-processes

Retrieves process information and system resource usage from the /proc file system.

#### Syntax

linux-processes

#### Description

After running the linux-processes command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| user | String | Process user |
| pid | Integer | Process ID |
| ppid | Integer | Parent process ID |
| cpu\_usage | Double | CPU utilization in percent |
| mem\_usage | Double | Memory utilization in percent |
| vsz | Integer | Virtual memory size(KiB) |
| rss | Integer | Resident set size(KiB) |
| status | String | Status |
| is\_deleted | Boolean | Whether the original executable file is deleted |
| cmd\_line | String | Complete command line |
| start\_time | Date | Process start time |

### linux-recent-files

Lists all recently created or modified files in all directories under the root directory. If no from or span options are specified, it only lists files created or modified in the last 1 day by default.

#### Syntax

linux-recent-files [OPTION]

Optional Parameter

**from=yyyyMMddHHmmss**

Start date and time of the search period in the form of yyyyMMddHHmmss (default: none). The time period for the search includes the specified time point. If you provide only the first part, the command recognizes the remaining digits as 0. For example, if you provide 20130605, the command recognizes it as 20130605000000 (June 5, 2013, 00:00:00). This option cannot be used with span.

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

Time range to search the recent files based on the current time (default: 1d). You can specify time in units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second). For example, 10s refers to "the last 10 seconds" based on the current time.

#### Description

After running the linux-recent-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_path | String | File path |
| file\_name | String | File name |
| file\_type | String | File type |
| permissions | String | Permissions |
| file\_size | Long | File size |
| file\_ctime | Date | Last change time |
| file\_mtime | Date | Last modification time |
| file\_atime | Date | Last access time |
| owner\_read | Boolean | Whether owner read permission is given |
| owner\_write | Boolean | Whether owner write permission is given |
| owner\_execute | Boolean | Whether owner execute permission is given |
| group\_read | Boolean | Whether group read permission is given |
| group\_write | Boolean | Whether group write permission is given |
| group\_execute | Boolean | Whether group execute permission is given |
| others\_read | Boolean | Whether other read permission is given |
| others\_write | Boolean | Whether other write permission is given |
| others\_execute | Boolean | Whether other execute permission is given |

#### Usage

View a list of files created or modified in the last day

linux-recent-files

View a list of files created or modified in the last 7 days

linux-recent-files span=7d

View a list of files created or modified after January 1, 2021

linux-recent-files from=20210101

### linux-rkhunter

Runs rkhunter and returns the results. In order to use this command, the rkhunter command must be available on your system. [rhhunter](http://rkhunter.sourceforge.net/) is open-source software for monitoring rootkits on POSIX compliant systems.

#### Syntax

linux-rkhunter [ignore-error=BOOL]

Optional Parameter

**ignore-error=BOOL**

Boolean option to handle error (default: f).

* t: Exits gracefully if an error occurs (e.g. rkhunter is not installed).
* f: Fails if an error occurs.

#### Description

After running the linux-rkhunter command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| target | String | Target |
| reason | String | Diagnosis |
| description | String | Description |

### linux-routes

Retrieves routing information from /proc/net/route.

#### Syntax

linux-routes

#### Description

After running the linux-routes command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| destination | IPv4 address | Destination network address |
| mask | IPv4 address | Netmask |
| forward | IPv4 address | Gateway address |
| flags | String | Flags (C: Cache entry, D: Dynamically installed by daemon or redirect, G: Use Gateway, H: Target is a host, M: Modified from routing daemon or redirect, U: Route is up) |
| mss | Integer | Maximum segment size |
| irtt | Integer | Initial round trip time |
| iface | String | Network interface name |

### linux-setuid-files

Runs the find / -user root -perm -4000 -print command and lists the files with setuid permissions.

#### Syntax

linux-setuid-files [md5=BOOL]

Optional Parameter

**md5=BOOL**

Boolean option to output the MD5 hash of files (default: f)

* t: outputs the MD5 hash of files
* f: NOT output the MD5 hash of files

#### Description

After running the linux-setuid-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_path | String | File path |
| file\_name | String | File name |
| file\_type | String | File type |
| permissions | String | Permissions |
| file\_size | Long | File size |
| file\_ctime | Date | Last change time |
| file\_mtime | Date | Last modification time |
| file\_atime | Date | Last access time |
| md5 | String | MD5 hash of the file |
| owner\_read | Boolean | Whether owner read permission is given |
| owner\_write | Boolean | Whether owner write permission is given |
| owner\_execute | Boolean | Whether owner execute permission is given |
| group\_read | Boolean | Whether group read permission is given |
| group\_write | Boolean | Whether group write permission is given |
| group\_execute | Boolean | Whether group execute permission is given |
| others\_read | Boolean | Whether other read permission is given |
| others\_write | Boolean | Whether other write permission is given |
| others\_execute | Boolean | Whether other execute permission is given |

### linux-shell-sessions

Retrieves a list of all currently logged in users (sessions) from the /var/run/utmp log.

#### Syntax

linux-shell-sessions

#### Description

After running the linux-shell-sessions command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| user | String | User ID |
| tty | String | Terminal |
| host | String | Host name |
| src\_ip | IPv4 address | Source IP address from which the user logged in |
| idle\_time | Integer | Idle time in seconds |
| jcpu | Double | CPU time consumed by all processes connected to the terminal in seconds |
| pcpu | Double | CPU time used by the current process in seconds |
| cmd\_line | String | Running command line |

### linux-system-files

Retrieves permissions and MD5 hashes of files in the /usr/bin directory.

#### Syntax

linux-system-files [md5=BOOL]

Optional Parameter

**md5=BOOL**

Boolean option to output the MD5 hash of files (default: f)

* t: outputs the MD5 hash of files
* f: NOT output the MD5 hash of files

#### Description

After running the linux-system-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_path | String | File path |
| file\_name | String | File name |
| file\_type | String | File type |
| permissions | String | Permissions |
| file\_size | Long | File size |
| file\_ctime | Date | Last change time |
| file\_mtime | Date | Last modification time |
| file\_atime | Date | Last access time |
| md5 | String | MD5 hash of the file |
| owner\_read | Boolean | Whether owner's read permission is given to the file. |
| owner\_write | Boolean | Whether owner's write permission is given to the file. |
| owner\_execute | Boolean | Whether owner's execute permission is given to the file. |
| group\_read | Boolean | Whether group's read permission is given to the file. |
| group\_write | Boolean | Whether group's write permission is given to the file. |
| group\_execute | Boolean | Whether group's execute permission is given to the file. |
| others\_read | Boolean | Whether others' read permission is given to the file. |
| others\_write | Boolean | Whether others' write permission is given to the file. |
| others\_execute | Boolean | Whether others' execute permission is given to the file. |

### linux-shell-commands

Retrieves the commands executed by each user from the bash\_history and .zsh\_history logs in /etc/passwd.

#### Syntax

linux-shell-commands

#### Description

After running the linux-shell-commands command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| user | String | User ID |
| cmd\_line | String | command line executed |

### linux-system-info

Retrieves system information, including hostname, kernel version, uptime, average load, and UMASK information.

#### Syntax

linux-system-info

#### Description

After running the linux-system-info command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| hostname | String | Host name |
| kernel\_ver | String | Kernel version |
| kernel\_build\_time | Date | Date and time when the kernel was built |
| uptime | Integer | Uptime |
| load\_avg\_1m | Double | 1-minute average load |
| load\_avg\_5m | Double | 5-minute average load |
| load\_avg\_15m | Double | 15-minute average load |
| umask | String | Permission mask applied when creating a file or directory |
| console\_banner | String | Console banner |
| telnet\_banner | String | Telnet banner |
| system\_banner | String | System banner |

### linux-systemd-services

Runs the systemctl -at service command to list Linux systemd services.

#### Syntax

linux-systemd-services

#### Description

After running the linux-systemd-services command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| service | String | Service unit |
| load\_status | String | Service unit load status |
| active\_status | String | High-level unit activation state |
| sub\_status | String | Low-level unit activation state |

### linux-systemd-timers

Runs the systemctl list-unit-files --type=timer command to list all systemd timers.

#### Syntax

linux-systemd-timers

#### Description

After running the linux-systemd-timers command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| timer | String | Timer name |
| status | String | Activation status |

### linux-tmp-files

Recursively lists the hidden files (those with names beginning with '.') under the /tmp, /dev, and /home directories.

#### Syntax

linux-tmp-files

#### Description

After running the linux-tmp-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| file\_path | String | File path |
| file\_name | String | File name |
| file\_type | String | File type |
| permissions | String | Permissions |
| file\_size | Long | File size |
| file\_ctime | Date | Last change time |
| file\_mtime | Date | Last modification time |
| file\_atime | Date | Last access time |
| owner\_read | Boolean | Whether owner read permission is given |
| owner\_write | Boolean | Whether owner write permission is given |
| owner\_execute | Boolean | Whether owner execute permission is given |
| group\_read | Boolean | Whether group read permission is given |
| group\_write | Boolean | Whether group write permission is given |
| group\_execute | Boolean | Whether group execute permission is given |
| others\_read | Boolean | Whether other read permission is given |
| others\_write | Boolean | Whether other write permission is given |
| others\_execute | Boolean | Whether other execute permission is given |

### linux-users

Retrieves a list of users from the /etc/passwd file.

#### Syntax

linux-users

#### Description

After running the linux-users command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| user | String | User ID |
| uid | Integer | UID |
| gid | Integer | GID |
| description | String | Description |
| home\_path | String | Path to the home directory |
| shell | String | Login shell |

### linux-user-groups

Retrieves a list of users belonging to the root and wheel groups by group.

#### Syntax

linux-user-groups

#### Description

After running the linux-user-groups command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| gid | Integer | Group ID |
| group | String | Group name |
| users | String | Users in the group |

### linux-vmstats

Reports the I/O status of the system. In order to use this command, the vmstat command must be available on your system.

#### Syntax

linux-vmstats

#### Description

After running the linux-vmstats command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| swap\_size | Long | Amount of virtual memory used |
| free\_size | Long | Amount of free virtual memory |
| buffer\_size | Long | Amount of memory used as buffers |
| cache\_size | Long | Amount of memory used as page cache |
| swap\_in | Long | Amount of memory swapped in from disk per second |
| swap\_out | Long | Amount of memory swapped to disk per second |
| block\_in | Long | Blocks received from a block device per second |
| block\_out | Long | Blocks sent to a block device per second |

## Web Browser Artifacts

### chrome-downloads

Retrieves the download history from the Google Chrome 'History' SQLite file.

#### Syntax

chrome-downloads FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to 'History' SQLite file

#### Description

After running the chrome-downloads command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Time of visit |
| start\_time | Date | Download start time |
| end\_time | Date | Download end time |
| mime\_type | String | MIME type |
| file\_open | Boolean | Whether the file has ever been opened or not |
| file\_path | String | Path to the file |
| file\_size | Integer | File size |
| url | String | File URL |
| referer | String | Referer URL |

#### Usage

Retirieve the download history from the file at /opt/logpresso/History.

chrome-downloads /opt/logpresso/History

### chrome-search-terms

Retrieves the search history from the Google Chrome 'History' SQLite file.

#### Syntax

chrome-search-terms FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to 'History' SQLite file

#### Description

After running the chrome-search-terms command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Time of visit |
| keywords | String | Searched keywords |
| title | String | Website title |
| url | String | Web page URL |

#### Usage

Retrieve the search history from the file at /opt/logpresso/History.

chrome-search-terms /opt/logpresso/History

### chrome-visits

Retrieves the website visit history from the Google Chrome 'History' SQLite file.

#### Syntax

chrome-visits FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to 'History' SQLite file

#### Description

After running the chrome-visits command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Time of visit |
| title | Integer | Website title |
| visit\_count | Integer | Number of website visits |
| typed\_count | Integer | Number of website visits by entering a URL directly |
| hidden | Boolean | Whether a webpage is loaded inside an iframe |
| url | String | URL visited |

#### Usage

Retrieve the website visit history from the file at /opt/logpresso/History.

chrome-visits /opt/logpresso/History

### esedb-columns

Retrieves a list of column definitions for a table from an Extensible Storage Engine (ESE) database file.

#### Syntax

esedb-columns table=TABLE FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the ESE database file.

**table=TABLE**

Name of the table from which to retrieve the column definition list.

#### Description

After running the esedb-columns command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_file | String | File name |
| table\_name | String | Table name |
| column\_id | Integer | Column identifier |
| column\_name | String | Column name |
| column\_type | String | Colum data type |

The **column\_type** field represents one of the following values: bit, unsigned byte, short, unsigned short, long, unsigned long, long long, binary , long binary, or text

#### Usage

Retrieve the list of columns in the MSysObjects table from the /opt/logpresso/WebCacheV01.dat file.

esedb-columns table=MSysObjects /opt/logpresso/WebCacheV01.dat

### esedb-records

Retrieves records in a specified table from an ESE database file.

#### Syntax

esedb-records table=TABLE FILE\_PATH

Required Parameter

**table=TABLE**

Name of the table in which to look up records.

**FILE\_PATH**

Path to the ESE database file.

#### Usage

Retrieve records in the MSysObjects table from the /opt/logpresso/WebCache/webCacheV01.dat file.

esedb-records table=MSysObjects /opt/logpresso/WebCacheV01.dat

### esedb-tables

Retrieves a list of all tables and columns from an ESE database file.

#### Syntax

esedb-tables FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the ESE database file.

#### Description

After running the esedb-tables command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_file | String | File name |
| table\_name | String | Table name |
| columns | List | List of columns |

#### Usage

Retrieve the list of all tables and columns from the /opt/logpresso/WebCacheV01.dat file.

esedb-tables /opt/logpresso/WebCacheV01.dat

### eml-file

Parse an [RFC 822](https://www.rfc-editor.org/rfc/rfc822) compliant EML file to extract email information such as headers, subject, and content.

#### Syntax

eml-file [zipcharset=CHARSET] [zippath=ZIPFILE\_PATH] [raw=BOOL] FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the EML file. Using a wildcard (\*) in the file name, you can retrieve all files containing a specific string pattern in the file name (e.g. D:\data\eml\*.eml). If you provided the zippath option, input the EML file path in the zip file.

Optional Parameter

**zipcharset=CHARSET**

Character set to be used to decode the ZIP entry name and comment that are not encoded by UTF-8 encoding. Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

**zippath=ZIPFILE\_PATH**

Path to the ZIP file.

**raw=t**

HTML format to be applied to mail\_content (default: f).

* t: HTML
* f: plain text

**Description**

The output fields are as follows:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_file | String | EML file name |
| \_time | Date | Sent date from "Date" header |
| mail\_from | String | Address from "From" header |
| mail\_from\_name | String | Display name from "From" header |
| mail\_to | String | Addresses from "To" header, delimited by a new line. |
| mail\_to\_name | String | Display names from "To" header, delimited by a new line. |
| mail\_cc | String | Addresses from "Cc" header, delimited by a new line. |
| mail\_cc\_name | String | Display names from "Cc" header, delimited by a new line. |
| mail\_bcc | String | Address from "Bcc" header, delimited by a new line. |
| mail\_bcc\_name | String | Display names from "Bcc" header, delimited by a new line. |
| mail\_subject | String | Subject |
| mail\_content | String | Text or HTML content. |
| mail\_attachments | String | Attachment file names, delimited by a new line. |
| mail\_headers | String | Header name-value pairs, delimited by a new line. |
| attachments | Array | Elements with file\_name and file\_size properties. |

#### Usage

Retrieve information by providing the EML file path.

eml-file sample.eml

Retrive information when the zippath option is provided.

eml-file zippath=image.zip sample.eml

### firefox-downloads

Retrieves the download history from the Firefox places.sqlite file.

#### Syntax

firefox-downloads FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the places.sqlite file.

#### Description

After running the firefox-downloads command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Time of visit |
| start\_time | Date | Download start time |
| end\_time | Date | Download end time |
| file\_path | String | Path to the file |
| file\_size | Integer | File size |
| url | String | File URL |

#### Usage

Retrieve the download history from the /opt/logpresso/places.sqlite file.

firefox-downloads /opt/logpresso/places.sqlite

### firefox-visits

Retrieves the website visit history from the Firefox places.sqlite file.

#### Syntax

firefox-visits FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the places.sqlite file.

#### Description

After running the firefox-visits command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Time of visit |
| title | String | Website title |
| visit\_count | Integer | Number of website visits |
| typed\_count | Integer | Number of website visits by entering a URL directly |
| hidden | Boolean | Whether to allow iframe to be set to transparent |
| url | String | URL visited |

#### Usage

Retrieve the website visit history from the /opt/logpresso/places.sqlite file.

firefox-visits /opt/logpresso/places.sqlite

### ie-cache-files

Retrieves cache file data from an ESE database file.

#### Syntax

ie-cache-files FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the ESE database file.

#### Description

After running the ie-cache-files command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Time of visit |
| container\_id | Integer | Container ID |
| entry\_id | Integer | Entry ID |
| cache\_id | Integer | Cache ID |
| visit\_count | Integer | Number of website visits |
| url | String | Download URL |
| file\_name | String | File name |
| file\_size | Integer | File size |
| sync\_time | Date | Last access time |
| creation\_time | Date | Creation time |
| expiry\_time | Date | Expiry time |
| modified\_time | Date | Last modification time |
| request\_headers | String | HTTP request headers |
| response\_headers | String | HTTP response headers |
| group | Binary | - |
| url\_hash | Integer | URL Hash |
| secure\_dir | Integer | Directory index |

#### Usage

Retrieves cache file data from the /opt/logpresso/WebCacheV01.dat file.

ie-cache-files /opt/logpresso/WebCacheV01.dat

### ie-cookies

Retrieves cookie data from an ESE database file.

#### Syntax

ie-cookies

Required Parameter

**FILE\_PATH**

Path to the ESE database file.

#### Usage

Retrieve cookie data from the /opt/logpresso/WebCacheV01.dat file.

ie-cookies /opt/logpresso/WebCacheV01.dat

### ie-downloads

Retrieves the download history from an ESE database file.

#### Syntax

ie-downloads FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the ESE database file.

#### Description

After running the ie-downloads command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Time of visit |
| container\_id | Integer | Container ID |
| entry\_id | Integer | Entry ID |
| cache\_id | Integer | Cache ID |
| visit\_count | Integer | Number of website visits |
| url | String | Download URL |
| file\_path | String | File path |
| file\_size | Integer | File size |
| sync\_time | Date | Downloaded time |
| response\_headers | String | HTTP Response header |
| url\_hash | Integer | URL Hash |
| secure\_dir | Integer | Directory index |

#### Usage

Retrieve the download history from the /opt/logpresso/WebCacheV01.dat file.

ie-downloads /opt/logpresso/WebCacheV01.dat

### ie-visits

Retrieves the website visit history from an ESE database file.

#### Syntax

ie-visits FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the ESE database file

#### Description

The output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Date | Time of visit |
| container\_id | Integer | Container ID |
| entry\_id | Integer | Entry ID |
| cache\_id | Integer | Cache ID |
| visit\_count | Integer | Number of website visits |
| user | String | NT account |
| url | String | URL visited |
| file\_size | Integer | File size |
| sync\_time | Date | Last access time |
| expiry\_time | Date | Expiry time |
| modified\_time | Date | Last modification time |
| response\_headers | String | HTTP response headers |
| url\_hash | Integer | URL hash |
| secure\_dir | Integer | Directory index |

#### Usage

Retrieve the website visit history from the /opt/logpresso/WebCacheV01.dat file.

ie-visits /opt/logpresso/WebCacheV01.dat

### sqlite-records

Retrieves records in a specified table from an SQLite database file.

#### Syntax

sqlite-records table=TABLE FILE\_PATH

Required Parameter

**table=TABLE**

Name of the table in which to look up records.

**FILE\_PATH**

Path to the SQLite file.

#### Usage

Retrieve records in the visits table in the /opt/logpresso/sqlite file.

sqlite-records table=visits /opt/logpresso/sqlite

### sqlite-tables

Retrieves the table scheme from an SQLite database file.

#### Syntax

sqlite-tables FILE\_PATH

Required Parameter

**FILE\_PATH**

Path to the SQLite file

#### Description

After running the sqlite-records command, the output fields are as follows:

**Output Fields**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| type | String | Object type |
| name | String | Object name |
| table\_name | String | Table name |
| root\_page | Integer | Root page number |
| sql | String | Table scheme |

#### Usage

Retrieve the table scheme from the /opt/logpresso/sqlite file.

sqlite-tables /opt/logpresso/sqlite

# System Commands

## System Congifuration

### confdb

Retrieves the documents or metadata stored in the confdb. This command receives the name of the database instance or collection as an argument. Administrative privileges are required to execute this command.

#### Syntax

confdb SUB-COMMAND

SUB-COMMAND

**databases**

Retrieve a list of all confdb database instances.

**cols DB\_NAME**

List all collection names of the specified confdb database instance.

**docs DB\_NAME COL\_NAME**

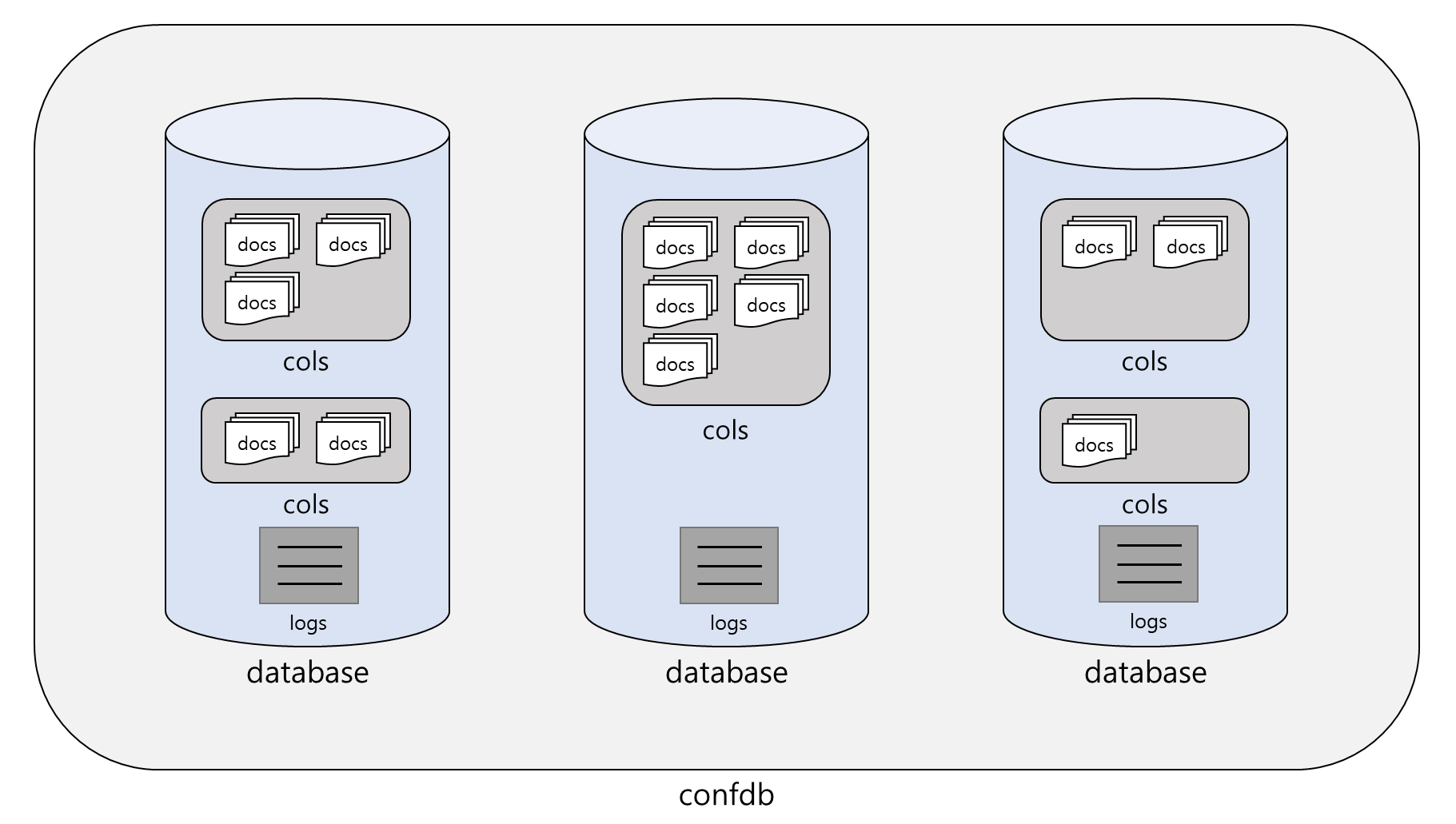
Retrieve all document objects of the specified confdb collection of the confdb database instance.

**logs DB\_NAME**

Retrieve commit logs of the specified confdb database instance.

#### Description

confdb is the database storing the (platform) settings of the Logpresso. The logical structure is as follows:



* confdb consists of multiple database instances.
* A database instance consists of more than one collection. A collection is similar to a table in a relational database (RDBMS).
* A collection consists of more than one document. A document is similar to a record in an RDBMS.
* A log is generated once a document is committed to confdb or database revision is updated.

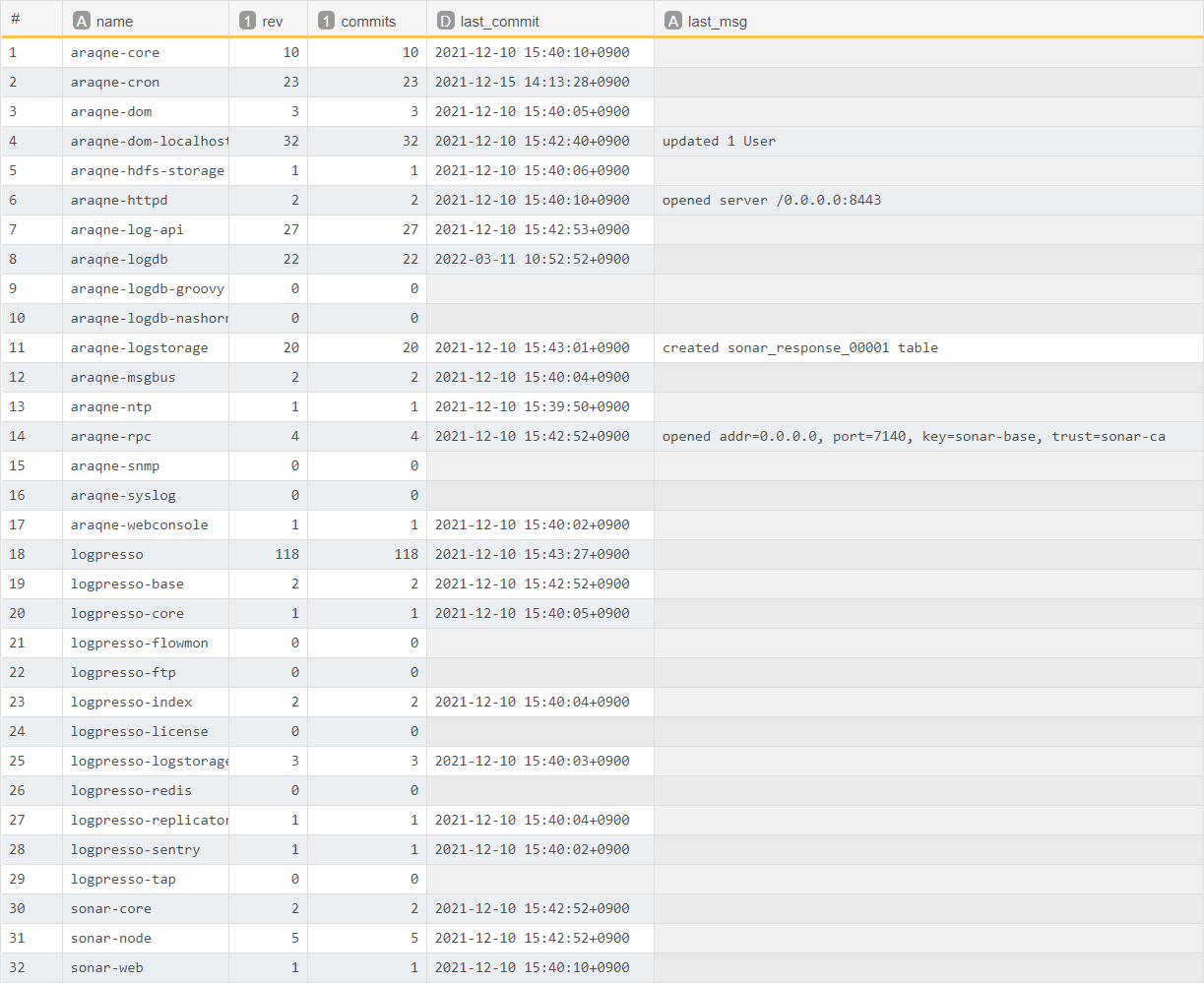
When you provide the unit of the database (databases, cols, docs) as an argument, it extracts the corresponding data.

Retrieve database instances

The following is an example command to get the list of database instances of confdb in Logpresso Sonar.

confdb databases| # change the output field order| order name, rev, commits, last\_commit, last\_msg| # sort in ascending order by name field| sort name

Result of the command confdb databases:

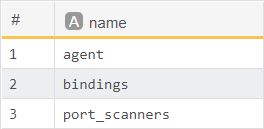


|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| name | String | Name of the database instance |
| rev | Integer | Revision number of the database instance |
| commits | Interger | Number of commits to the database |
| last\_commit | Date | Date of the last commit |
| last\_msg | String | Last commit message |

List all collection names

The following is an example command to get the list of collections for the database instances araqne-rpc. This collection list has name field only.

confdb cols araqne-rpc

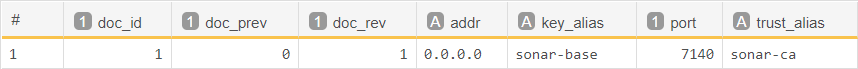


Retrieve documents from collection

The following is an example command to retrieve all documents in the bindings collection in the araqne-rpc database instance.

confdb docs araqne-rpc bindings

A document consists of common attribute fields such as doc\_id, doc\_prev, and doc\_rev, as well as document-specific fields.



For common attribute fields, refer to the below table.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| doc\_id | Integer | Serial number of document |
| doc\_prev | Integer | Previous revision number of document |
| doc\_rev | Integer | Current revision number of document |

Retrieve commit logs

You can view the commit history of the database instance.

confdb logs araqne-rpc

For commit log fields, refer to the below table.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| \_time | Time | Time of commit |
| rev\_id | Integer | Commit revision ID |
| committer | String | Name of committer module (araqne-rpc in the above example) |
| msg | String | Summary of commit |
| manifest\_id | Integer | Manifest ID for changeset |

## Log collectors

### system loggers

Loads the status of all currently configured log collectors. Administrative privileges are required to execute this command.

#### Syntax

system loggers

#### Description

Output fields are as follows:

* **namespace**: Name of the set to which the logger belongs (the logger executed in the Logpresso local means the local set, otherwise it is displayed as the name (guid) of the sentry where the logger is located)
* **name**: Name of the logger
* **factory\_namespace**: Name of the set to which the logger factory belongs
* **factory\_name**: Name of the logger factory
* **status**: Status of the current logger (running: being executed, stopped: stopped)
* **interval**: Logger execution cycle (unit: ms)
* **cron\_schedule**: Scheduled execution schedule
* **log\_count**: Number of logs collected so far
* **drop\_count**: Number of dropped logs
* **log\_volume**: Total volume of the log data (bytes)
* **drop\_volume**: Total volume of the dropped log data (bytes)
* **last\_start\_at**: Last time the logger started
* **last\_run\_at**: Last time the logger was executed
* **last\_log\_at**: Time at which the last log was generated
* **last\_write\_at**: Last time the log was received

A logger factory refers to a group of different loggers working together to perform a function. The logger factory divides its areas for each node (Logpresso (local) or sentry (guid)). Even with the same logger factory, what belongs to the local namespace and what belongs to the namespace of a sentry are different logger factories.

## Tables and Data

### system tables

Loads the table list in the system. The administrator can load the list of all tables in the system. Normal user accounts can only load the list of tables to which they have been granted privileges.

#### Syntax

system tables

#### Description

Output fields are as follows.

* **java table**: Table name
* **metadata**: Metadata of the table
* **primary\_configs**: Setting information on primary storage
* **replica\_configs**: Setting information on backup storage
* **lock\_owner**: Lock owner
* **lock\_purpose**: Purpose of locking
* **lock\_reentcnt**: Number of times the locking process locked while re-entering
* **retention\_policy**: Table archiving cycle (unit: day)
* **data\_path**: Table data file path
* **is\_locked**: Whether to lock the table (true: locked, false: unlocked)
* **privileges**: Read permission to the the user account table
* **security\_groups**: Read permission to the user group table

To check the records stored in the table, use the [table](https://docs.logpresso.com/en/query/table-command) command.

### system count

Loads the number of records stored until the time of the query by date.

#### Syntax

system count [from=yyyyMMdd] [to=yyyyMMdd] [diskonly=BOOL] [TABLE, ...]

Optional Parameter

**from=yyyyMMdd**

Start date of the search period in the form of yyyyMMdd. The time period for the search includes the specified time point.

**to=yyyyMMdd**

End date of the search period in the form of yyyyMMdd. Unlike the to option of the [table](https://docs.logpresso.com/en/query/table-command) command, the time period for the search includes the specified time point.

**diskonly=BOOL**

Option to load the number of records (default: f).

* t: Loads the number of records written to the disk only.
* f: Loads the number of records, including the number of records buffered in the memory.

**TABLE, ...**

Tables in which to check the number of records, separated by a comma (,). If you do not specify a table name, the command checks the number of records in all tables for which the user is granted read permission.

#### Description

Output fields are as follows.

* **\_time**: Partition date
* **table**: Table name
* **count**: Number of records in the table

### checktable

Checks the integrity of the table data in the specified date range. Administrative privileges are required to execute this command.

#### Syntax

checktable [from=yyyyMMdd] [to=yyyyMMdd] [trace=BOOL] [TABLE, ...]

Optional Parameter

**from=yyyyMMdd**

Specify the start date (check including the start date) of the integrity check in the form of yyyyMMdd.

**to=yyyyMMdd**

Specify the last date (check including the last date) of the integrity check in the form of yyyyMMdd.

**trace=BOOL**

If you set it to t, the command also returns normal data block information that has no abnormality in integrity. If you do not use this option or set it to f, the command only displays data block information with corrupted integrity.

**TABLE, ...**

Specify the tables to be checked for their integrity by separating them using commas (,). If you do not specify a table, the command checks the integrity of all tables for which the user is granted read permission. The table name supports wildcards (\*).

#### Description

An integrity check is performed only if the target table uses an "encryption profile in which a digest algorithm is set". The tables that do not contain the HMAC signature required for the integrity check are automatically excluded from the check.

The fields displayed when executing the command are as follows.

* **table**: Table name
* **day**: Date partition name
* **block\_id**: Block ID
* **last\_block\_id**: The last block ID, which appears only if the integrity is corrupted
* **signature**: The hash value calculated at the time of data creation
* **hash**: The hash value calculated at the time of the integrity check. If this value differs from the signature field value, it is considered a tampered one.
* **msg**: Displayed as a valid, modified, or corrupted string. If the data is tampered or corrupted, it is skipped because the data block cannot be read when executing the data retrieval query.
* valid: Integrity is validated.
* modified: Data is tampered.
* corrupted: The file structure is corrupted.

If no abnormality is found during the integrity check, there is no specific output result.

#### Usage

Check the integrity of data for all tables of September 2014.

checktable from=20140901 to=20140930 \*

Check the Integrity of all table data starting with syslog\_.

checktable syslog\_\*

### copytable

Copies or moves table data and index data files in the specified date range to the specified path. Administrative privileges are required to execute this command.

#### Syntax

copytable [from=yyyyMMdd] [to=yyyyMMdd] [incremental=BOOL|overwrite=BOOL|worm=BOOL] [move=BOOL] [tables="TABLE, ..."] [indexpath="PATH"] path="PATH"

Required Parameter

**path="PATH"**

Path to save the table backup by enclosing it in a pair of double quotes (" "). If the backup path has special characters such as backslashes (\) or whitespaces, you need to escape it using backslashes (\).

Optional Parameter

**from=yyyyMMdd**

Start date of record to be backed up in the form of yyyyMMdd. The command backs up all the records within the specified period including the start date.

**to=yyyyMMdd**

End date of last record to be backed up in the form of yyyyMMdd. The command backs up all the records within the specified period including the end date.

**incremental=BOOL**

Option to enable incremental backup (default: f).

* t: Enables incremental backup. When incremental=t, the command appends data to the end of the file when there is an identical file in the path specified by the path option. This option cannot be used with the worm or overwrite option.
* f: Disables incremental backup.

**overwrite=BOOL**

Option to enable overwriting the file specified by the path option, if it exists (default: f).

* t: Overwrites the file specified by the path option if it exists. You can keep the old backup file even if you cancel it while the backup is in progress. This option cannot be used with the incremental and the worm option.
* f: NOT overwrite the file specified by the path option if it exists. The query fails if the file exists.

**worm=BOOL**

Option to copy tables to WORM (Write Once Read Many) storage or CD (default: f).

* t: Copies the source files to the backup media without creating a temporary file with the .transfer extension. This option cannot be used with the incremental or the overwrite option.
* f: Writes data to a temporary file with the .transfer extension, deletes the source file when writing to this file is complete, and renames this file to that of the source file.

**move=BOOL**

Option to delete/keep the source file after copying is complete.

* t: Deletes the source file after copying is complete. If the file size of the backup media does not match the file volume of the source, the command does not delete the source file.
* f: Keeps the source file.

**tables="TABLE, ..."**

Tables to be backed up, separated by a comma (a) (default: all the tables). Define the entire list by enclosing it in a pair of double quotes (" ").

**indexpath="PATH"**

Path to save the full-text index file by enclosing it in a pair of double quotes (" ") (default: none). If the backup path has special characters such as backslashes (\) or whitespaces, you need to escape it using backslashes (\).

#### Description

This command is typically used to periodically back up table data and index data files to storage such as NAS.

This command returns the current progress while copying each data file. If exceptions such as the existence of a file with the same name, failure to rename, or insufficient capacity of the backup media occur, the command displays the contents of the error in the **error\_msg** field. This allows you to perform other post-processing, such as sending an alert email when an error occurs. If the backup of some data files fails during the process, the query runs without interruption until the user explicitly cancels it.

#### Usage

Copy all table data files to the /backup path.

copytable path="/backup"

Copy all table data files from June 24, 2015, to June 25, 2015, to the e:\backup path.

copytable from=20150624 to=20150625 path="e:\\backup"

Move all table data files from June 24, 2015, to June 25, 2015, to the /backup path.

copytable from=20150624 to=20150625 move=t path="/backup"

Copy test table and index data files from June 24, 2015, to June 25, 2015, to the /backup path.

copytable from=20150624 to=20150625 tables="test" path="/backup" indexpath="/backup"

### purge

Removes the data records within the specified date range from the table. Administrative privileges are required to execute this command.

This command is only available if you add the -Daraqne.logdb.purge=enabled switch as a Logpresso boot option.

#### Syntax

purge from=yyyyMMdd to=yyyyMMdd TABLE, ...

Required Parameter

**from=yyyyMMdd**

Start date of the first record to be discarded in the form of yyyyMMdd. The command discards all the records within the specified period including the start date.

**to=yyyyMMdd**

End date of the last record to be discarded in the form of yyyyMMdd. The command discards all the records within the specified period including the end date.

**TABLE, ...**

Tables for which data is to be discarded, separated by a comma (,). This is used when you want to discard old data and provide new data each time you execute a query.

#### Usage

Discard sample table data on September 10, 2014, and September 11, 2014

purge from=20140910 to=20140911 sample

### system logdisk

Checks the disk usage of the compressed table data file by date.

#### Syntax

system logdisk [from=yyyyMMdd] [to=yyyyMMdd] [TABLE, ...]

Required Parameter

**from=yyyyMMdd**

Start date (including the start date) of the target to load in the form of yyyyMMdd.

**to=yyyyMMdd**

End date (including the last date) of the target to load in the form of yyyyMMdd.

**TABLE, ...**

Tables for which to check the disk usage of the table data file, separated by a comma (,). If you omit the table list, the command loads the usage of all tables for which the user account that executes the command has read permission.

#### Description

Output fields are as follows.

* **\_time**: Partition date
* **table**: Partition date
* **disk\_usage**: Disk usage in bytes

### system indexdisk

Loads the disk usage of all index files stored up to the time of the query by date and type.

#### Syntax

system indexdisk [from=yyyyMMdd] [to=yyyyMMdd] [TABLE, ...]

Optional Parameter

**from=yyyyMMdd**

Specify the start date (including the start date) of the target to load in the form of yyyyMMdd.

**to=yyyyMMdd**

Specify the last date (including the last date) of the target to load in the form of yyyyMMdd.

**TABLE, ...**

Tables for which to check the disk usage of the index file, separated by a comma (,). If you omit the table list, the command loads the usage of the index file on all tables for which the user account that executes the command has read permission.

#### Description

Output fields are as follows.

* **\_time**: Partition date
* **table**: Table name
* **index**: Index name
* **type**: Index type
* **disk\_usage**: Disk usage in bytes

## Lookup

### system lookups

Loads a list of all lookup names currently registered in the system.

#### Syntax

system lookups

#### Description

Output fields are as follows.

* name: Lookup name

Lookup refers to data in the form of a file or database used for data mapping. You can retrieve the lookup in the **Query > Lookup** menu in the web console (ENT, STD) and see the same information as this command displays.

For related commands, refer to the following:

* [lookup](https://docs.logpresso.com/en/query/lookup-command): Converts a specific field value into another value by importing the mapping table.
* [lookuptable](https://docs.logpresso.com/en/query/lookuptable-command): Loads the contents of the lookup table added in the form of a file.
* [memlookup](https://docs.logpresso.com/en/query/memlookup-command): Creates an in-memory mapping table and loads metadata.

## Queries

### system queries

Loads the status of all queries that are currently being executed. The administrator can load all queries executed on the system, and the normal user account can only load queries executed by itself.

#### Syntax

system queries

#### Description

When you execute this command, the command returns a record containing the execution history of the query.

The record consists of the following fields.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| **id** | Integer | Query identifier |
| **query\_string** | String | Query string |
| **is\_eof** | Boolean | Whether the query is terminated (true: terminated, false: being executed) |
| **is\_end** | Boolean | Whether the query is terminated (true: terminated, false: being executed) |
| **is\_cancelled** | Voolean | Whether the query is canceled (true: canceled, false: not canceled) |
| **start\_time** | Long | Query start time (unit: epoch) |
| **finish\_time** | Long | Query finish time (unit: epoch) |
| **last\_started** | Date | Last refresh time |
| **elapsed** | Long | Time required to execute the query (unit: ms), If the query has not started, null |
| **background** | Boolean | Whether to execute in the background (true: Background query, false: Not a background query) |
| **commands** | Object | Execution status for each detailed command |
| **sub\_queries** | Array | Subquery list |
| **is\_scheduled\_query** | Boolean | Whether the query is scheduled (true: Scheduled query, false: Not a scheduled query) |
| **login\_name** | String | The account that executes the query |
| **remote\_ip** | String | The access IP address of the account that executes the query |
| **rows** | Long | The number of records returned as a result of executing the query |

'is\_end' is a legacy field left for backward compatibility. Use 'is\_eof' instead of 'is\_end' when referencing the results of the "system queries" command in a query.

### system streams

Loads the stream list in the system. Administrative privileges are required to execute this command.

#### Syntax

system streams

#### Description

Output fields are as follows.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| source\_type | String | The type of data source(logger: log collector, table: table, stream: stream query). |
| name | String | Stream query name |
| running | Boolean | Whether to execute (true: being executed, false: stopped) |
| enabled | Boolean | Whether to activate (true: activated status, false: deactivated status) |
| async | Boolean | Asynchronous mode (true: asynchronous mode, false: synchronous mode) |
| description | String | Description of the stream |
| interval | Integer | Refresh cycle (unit: ms) |
| query\_string | String | Query string |
| input\_count | Integer | Number of inputs |
| output\_count | Integer | Number of outputs |
| owner | String | The account that executes the query |
| created\_at | Date | Creation time |
| modified\_at | Date | Last modification time |
| last\_refresh\_at | Date | Last refresh time |
| input\_tables | String | A list of tables coming in as input |
| input\_loggers | String | A list of log collectors coming in as input |
| input\_streams | String | A list of streams coming in as input |

You can see the data coming into the stream with the following commands: logger, table, and stream.

### system ceptopics

Loads statistics of the currently registered event context by topic. CEP is an abbreviation for complex event processing.

#### Syntax

system ceptopics

#### Description

Output fields are as follows.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| **topic** | String | Event context topic |
| **count** | Integer | The amount of currently existing event contexts |

#### See also

* [evtctxlist](https://docs.logpresso.com/en/query/evtctxlist-command)
* [evtctxadd](https://docs.logpresso.com/en/query/evtctxadd-command)
* [evtctxdel](https://docs.logpresso.com/en/query/evtctxdel-command)
* [evtctxdrop](https://docs.logpresso.com/en/query/evtctxdrop-command)
* [evtctxget()](https://docs.logpresso.com/en/query/evtctxget-function)
* [evtctxgetvar()](https://docs.logpresso.com/en/query/evtctxgetvar-function)
* [evtctxsetvar()](https://docs.logpresso.com/en/query/evtctxsetvar-function)

### system cepclocks

If you use an external clock using the log timestamp based on the expiration and timeout for the CEP context, this reads the clock for each host that has been registered so far.

#### Syntax

system cepclocks

#### Description

Output fields are as follows.

* **host**: Host name
* **time**: Host clock
* **timeout\_queue\_len**: Timeout queue length
* **expire\_queue\_len**: Expire queue length

## PCAP Devices

### system pcapdevices

Loads the list of PCAP network interfaces available in Logpresso.

#### Syntax

system pcapdevices

#### Description

Output fields are as follows.

* **name**: Network interface name (string)
* **description**: A description of the network interface (string)
* **ip**: The IP address (IP address object) assigned to the network interface
* **mac**: The MAC address (string) of the network interface
* **subnet**: The network address of the connected network
* **netmask**: The netmask of the connected network

## Sentries

### system sentries

Loads all sentry status information registered in the Logpresso server. Administrative privileges are required to execute this command.

#### Syntax

system sentries

#### Description

You can see the overview of the sentry process (guid, connection status, installation path, and Java version), information of the installation server, and performance snapshots (OS, CPU, memory, and network) and more.

Output fields are as follows.

* **guid**: Unique sentry identifier
* **host\_name**: Host name
* **remote\_ip**: The IP address of the sentry
* **is\_connected**: true if connected, false if disconnected
* **description**: Descriptive information of the sentry (if it exists)
* **cpu\_usage**: Cpu usage (cpu\_kernel + cpu\_user)
* **mem\_usage**: Current memory usage (%)
* **disk\_usage**: Disk usage (bytes)
* **nic\_rx\_usage**: Number of network receive packets (receive)
* **nic\_tx\_usage**: Number of network transmit packets (transmit)
* **user\_dir**: Location of the execution directory
* **cpu\_kernel**: CPU usage of the kernel (%)
* **cpu\_user**: CPU usage of the user process (%)
* **phy\_used**: Physical memory usage (bytes)
* **phy\_free**: Amount of physical memory remaining (bytes)
* **phy\_total**: Total physical memory size (bytes)
* **swap\_used**: Swap memory usage (bytes)
* **swap\_free**: Amount of swap memory remaining (bytes)
* **swap\_total**: Total size of the swap memory (bytes)
* **last\_connect\_at**: Last connection time
* **os**: Operating system name
* **os\_ver**: Operating system version
* **arch**: Architecture
* **jvm\_name**: Java virtual machine (JVM) name
* **jvm\_version**: Java virtual machine (JVM) version
* **ip\_addrs**: Fully allocated IP address array
* **disks**: Displays total disk usage by separating it into total, available, and used (bytes)
* **nics**: Network card list and transmission/reception volume

### sentry-arp-cache

Retrieves ARP cache information of a sentry.

#### Syntax

sentry-arp-cache [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query command requires that the input record contains a guid field value.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |

Output Fields

This query command returns the fields below in addition to the fields from input record.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| type | String | ARP cache entry type (static, dynamic, invalid) |
| ip | IP Address | IP address |
| mac | String | MAC address |

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Get ARP cache information from all sentries.

sentry | fields guid | sentry-arp-cache

### sentry-bundles

Loads the bundle list of a sentry.

#### Syntax

sentry-bundles [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query command requires that the input record contains a guid field value.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |

Output Fields

This query command returns the fields below in addition to the fields from input record.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| bundle\_id | 64-bit Integer | Bundle ID |
| symbolic\_name | String | Bundle symbolic name |
| version | String | Bundle version |
| state | String | Byndle status |
| last\_modified | Date | Last modification |
| integrity | String | Integrity status |

Possible bundle statuses in the state field are as follows:

|  |  |
| --- | --- |
| Status | Description |
| ACTIVE | Bundle is running. |
| INSTALLED | Bundle is installed but bundle's dependencies have not been met. |
| RESOLVED | Bundle dependencies are all resolved and bundle is ready to start. |
| STARTING | Bundle is starting. |
| STOPPING | Bundle is stopping. |
| UNINSTALLED | Bundel has been removed. |

Possible integrity statuses in the integrity field are as follows:

|  |  |
| --- | --- |
| Status | Description |
| no signature | Digitally signed hash value is not found. |
| hash error | SHA-512 hashing of the bundle failed (e.g. I/O error) |
| verified | The hash values match, the data has not been altered. |
| modified | The hash values do not match, the data has been modified. |

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Load bundle lists of all sentries.

sentry | fields guid | sentry-bundles

### sentry-jstack

Retrieves stack status information of sentry threads. This command is used to diagnose operation status of logger.

#### Syntax

sentry-jstack [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query command requires that the input record contains a guid field value.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |

Output Fields

This query command returns the fields below in addition to the fields from input record.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| tid | 64-bit Integer | Thread ID |
| name | String | Thread name |
| state | String | Thread status (RUNNABLE, BLOCKED, TIMED\_WAITING, WAITING) |
| stacktrace | String | Threas stack |

Possible thread statuses in state field are as follows:

|  |  |
| --- | --- |
| Status | Description |
| RUNNABLE | Thread is ready to run. |
| BLOCKED | Thread is temporarily inactive until target object is unlocked. |
| TIMED\_WAITING | Thread is in a timed waiting state. |
| WAITING | Thread is temporarily inactive to sync with other thread. |

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

View thread stack statuses of all sentries.

sentry | fields guid | sentry-jstack

### sentry-logger-configs

Retrieves the logger settings of a sentry.

#### Syntax

sentry-logger-configs [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query requires that the input record contains guid and name field values.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |
| name | String | Unique logger identifier in the sentry name space |

Output Fields

This query command returns the fields below in addition to the fields from input record.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| configs | Map | Logger settings in key-value pair |

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| name should be not null | name field value from input record is null. |
| name should be string | name field value from input record is not a string. |
| name should be non empty string | name field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Retrieves logger settings of all sentries.

sentry | fields guid | sentry-loggers | sentry-logger-configs

### sentry-logger-connect

Sets a specific logger of a sentry to send log data to the Logpresso server.

#### Syntax

sentry-logger-connect [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query requires that the input record contains guid and name field values.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |
| name | String | Unique logger identifier in the sentry namespace |

Output Fields

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| name should be not null | name field value from input record is null. |
| name should be string | name field value from input record is not a string. |
| name should be non empty string | name field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Create a wtmp logger on all connected sentries and set the loggers to send the log to a remote server.

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

Creates a logger on a sentry.

#### Syntax

sentry-logger-create [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This command requires that the input record contains guid, name, factory\_name and configs field values.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required | Description |
| guid | String | O | Unique sentry identifier (Not relevant to GUID in JAVA) |
| name | String | O | Unique logger identifier in the sentry namespace |
| description | String | X | Logger description |
| factory\_name | String | O | Logger factory identifier |
| configs | Map | O | Logger configuration |
| table\_name | String | X | Name of table to which to store the logged data |
| host\_tag | String | X | Host tag (\_host field tag value) |

* factory\_name: Using the logapi.loggerFactories command on the Logpresso shell loads the logger factory list. Logger factories available may vary depending on the operating system or the installed app.
* configs: Use the [sentry-logger-configs](https://docs.logpresso.com/en/query/sentry-logger-configs-command) query command to check the existing logger configuration before creating a new logger.

Output Fields

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:llows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| name should be not null | name field value from input record is null. |
| name should be string | name field value from input record is not a string. |
| name should be non empty string | name field value from input record is empty. |
| factory\_name should be not null | factory\_name field value from input record is null. |
| factory\_name should be string | factory\_name field value from input record is not a string. |
| factory\_name should be non empty string | factory\_name field value from input record is empty. |
| unsupported factoryname: factory\_name | factory\_name logger factory is not supported. |
| configs should be not null | configs field value from input record is null. |
| configs should be dict type | configs field value from input record is not a map type. |
| all values of configs should be string type | Key-value pair of the configs map is not a string. |
| missing config key: name | Required configuration key (name) is missnig. |
| table\_name should be non empty string | table\_name field value from input record is empty. |
| host\_tag should be non empty string | host\_tag field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Creates wtmp loggers on all connected linux sentries.

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

Bulk deploys a set of loggers defined in a logger provisioning profile to a sentry.

#### Syntax

sentry-logger-deploy

#### Description

Logpresso supports a logger provisioning profile function that automatically sets up loggers on a sentry upon connecting to the sentry. This enables collecting logs of instances that are dynamically generated from the cloud. However, if you want logger provisioning to run automatically upon connection, you first need to set the environment variable logpresso.sentry.logger\_provisioning\_profile when starting the sentry. The sentry-logger-deploy command automatically deploys a set of loggers defined by a provisioning profile to the specified Logpresso sentry, even if you do not specify a logger provisioning profile when the Logpresso sentry boots up.

This command only forwards the provisioning start request and does not wait. The logger configuration may still be waiting or processing even after the command ends. If a failure occurs during logger provisioning, it will not be notified other than the system log.

You can check the number of queued logger provisioning tasks using the logpresso.loggerProvisioningTasks command in the logpresso shell.

Input Field

This query requires that the input record contains guid and profile\_guid field values.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |
| profile\_guid | String | Logger provisioning profile identifier (32-character long GUID) |

Output Fields

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| profile\_guid is null | profile\_guid field value from input record is null. |
| profile\_guid should be string | profile\_guid field value from input record is not a string. |
| profile\_guid should be non empty string | profile\_guid field value from input record is empty. |
| not connected | Sentry is not connected. |
| profile not found: profile\_guid | Logger provisioning profile does not exist. |

#### Usage

Provision window loggers on the window sentries.

sentry | search os == "Windows\*" | eval profile\_guid="448c0422-7a30-42ef-b73a-e855e538f779" | sentry-logger-deploy

### sentry-logger-disconnect

Disconnects a specific logger of a sentry so as not to send log data to the Logpresso server.

#### Syntax

sentry-logger-disconnect [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query requires that the input record contains guid and name field values.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |
| name | String | Unique logger identifier in the sentry namespace |

Output Fields

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| name should be not null | name field value from input record is null. |
| name should be string | name field value from input record is not a string. |
| name should be non empty string | name field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Disconnect a specific logger from all connected sentries.

sentry | sentry-loggers | search name == "wtmp\_linux" | sentry-logger-disconnect

### sentry-logger-remove

Removes a specific logger from a sentry.

#### Syntax

sentry-logger-remove [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query requires that the input record contains guid and name field values.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Name | Description |
| guid | String | Unique sentry identifier | Not relevant to GUID in JAVA |
| name | String | Logger identifier | Unique logger identifier in the sentry namesapce |

Output Fields

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| name should be not null | name field value from input record is null. |
| name should be string | name field value from input record is not a string. |
| name should be non empty string | name field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Remove the 'wtmp\_linux' logger from all connected sentries.

sentry | sentry-loggers | search name == "wtmp\_linux" | sentry-logger-remove

This action is irreversible. Check the loggers to remove before execution by commenting out ('#') the last query sentence 'sentry-logger-remove'.

### sentry-logger-set-interval

Updates the logging interval of a specific sentry logger.

#### Syntax

sentry-logger-set-interval [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query requires that the input record contains guid, name and interval field values.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |
| name | String | Unique logger identifier in the sentry namespace |
| interval | 32-bit Integer | Logging interval (Unit: millisecond) |

Output Fields

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:llows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| name should be not null | name field value from input record is null. |
| name should be string | name field value from input record is not a string. |
| name should be non empty string | name field value from input record is empty. |
| interval should be not null | interval field value from input record is null. |
| interval should be integer | interval field value from input record is not integer. |
| logger not found: name | Specified logger is not found. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |
| sentry method not found: setLoggerInterval | Sentry version is earlier than 3.10.2106.0. This command is not supported. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Set the logging interval of the 'wtmp\_linux' logger in all connected sentries to 5 seconds.

sentry | sentry-loggers | search name == "wtmp\_linux" | eval interval = 5000 | sentry-logger-set-interval

### sentry-logger-set-schedule

Updates the cron-format logging schedule of a specific sentry logger.

#### Syntax

sentry-logger-set-schedule [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query requires that the input record contains guid, name and cron\_schedule field values.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required | Description |
| guid | String | O | Unique sentry identifier (Not relevant to GUID in JAVA) |
| name | String | O | Unique logger identifier in the sentry namespace |
| cron\_schedule | String | X | Logging schedule in cron job format (If null is given, the existing schedule is deleted.) |

* cron\_schedule: A schedule is defined using the unix-cron string format. For more details on scheduling, use man 5 crontab on your Linux.

Output Fields

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| name should be not null | name field value from input record is null. |
| name should be string | name field value from input record is not a string. |
| name should be non empty string | name field value from input record is empty. |
| cron\_schedule should be string | cron\_schedule field value from input record is not a string. |
| wrong cron expression format: expr | cron\_schedule field value from input record is not of valid cron format. |
| logger not found: name | Logger of the specified name (name) is not found. |
| logger is running: name | You cannot update the schedule of the running logger (name). |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |
| sentry method not found: setLoggerSchedule | Sentry version is earlier than 3.10.2106.0. This command is not supported. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Set 'wtmp\_linux' loggers in all connected sentries to run at minute 0.

sentry | sentry-loggers | search name == "wtmp\_linux" | eval cron\_schedule="0 \* \* \* \*" | sentry-logger-set-schedule

### sentry-logger-set-time-range

Updates the logging time range of a specific sentry logger.

#### Syntax

sentry-logger-set-time-range [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query requires that the input record contains guid, name, start\_time and end\_time field values.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required | Description |
| guid | String | O | Unique sentry identifier (Not relevant to GUID in JAVA) |
| name | String | O | Unique logger identifier in the sentry namespace |
| start\_time | String | X | Logging start time (Format: HH:mm) |
| end\_time | String | X | Logging end time (Format: HH:mm) |

You can remove the existing time range setting by providing null to both start\_time and end\_time fields.

Output Fields

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:llows:llows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| name should be not null | name field value from input record is null. |
| name should be string | name field value from input record is not a string. |
| name should be non empty string | name field value from input record is empty. |
| invalid start\_time format | start\_time field value from input record is not in valid format (HH:mm). |
| invalid end\_time format | end\_time field value from input record is not in valid format (HH:mm). |
| start\_time is not null but end\_time is null | end\_time field value is missing. start\_time field value is set. |
| end\_time is not null but start\_time is null | start\_time field value is missing. end\_timefield value is set. |
| logger not found: name | Logger of the specified name (name) is not found. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Set 'weblog' loggers of all connected sentries to run from 10:00 PM to 06:00 AM only.

sentry | sentry-loggers | search name == "weblog" | eval start\_time="22:00", end\_time="06:00" | sentry-logger-set-time-range

### sentry-logger-start

Activates a specific logger of a sentry to run at the specified interval.

#### Syntax

sentry-logger-start [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This command receives a sentry identifier field (guid) value, a logger name field (name) value and a logging interval field (interval) value from an input record and then sends an asynchronous RPC message to the sentry to activate the logger to run every specified interval.

Logpresso environment variable logpresso.core.sentry\_rpc\_parallel determines the length of the asynchronous RPC request message queue. The default value is 100 and you can adjust the length by modifying this environment variable.

The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send an RPC message to 150 sentries, the server first sends RPC messages to 100 sentries and waits until RPC responds. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries have responded or timeout has been exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

This command outputs results in the order that the server received the sentry's RPC response message, which means the order of the output records may differ from the order of the input records. Use the output data but do not rely on the order of the records.

Input Field

This query requires that the input record contains guid, name and interval field values.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |
| name | String | Unique logger identifier in the sentry namespace |
| interval | 32-bit Integer | Logging interval (Unit: millisecond) |

Output Fields

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| name should be not null | name field value from input record is null. |
| name should be string | name field value from input record is not a string. |
| name should be non empty string | name field value from input record is empty. |
| interval should be not null | interval field value from input record is null. |
| interval should be integer | interval field value from input record is not an integer. |
| logger is already running | Logger is already running. |
| timeout | RPC request timeout exceeded. |
| disconnected | Disconnected while processing RPC request. |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Activate loggers of all sentries to run every 5 seconds.

sentry | sentry-loggers | eval interval = 5000 | fields guid, name, interval | sentry-logger-start

### sentry-logger-stop

Deactivates a specific logger of a sentry.

#### Syntax

sentry-logger-stop [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query requires that the input record contains guid and name field values.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |
| name | String | Unique logger identifier in the sentry namespace |

Output Fields

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:llows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| name should be not null | name field value from input record is null. |
| name should be string | name field value from input record is not a string. |
| name should be non empty string | name field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Deactivate loggers of all sentries.

sentry | sentry-loggers | fields guid, name | sentry-logger-stop

### sentry-loggers

Loads the logger list of a sentry.

#### Syntax

sentry-loggers [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query command requires that the input record contains a guid field value.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |

Output Fields

This query command returns the fields below in addition to the fields from input record.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| name | String | Logger identifier |
| description | String | Logger Description |
| factory\_name | String | Logger type identifier (e.g. syslog, wtmp) |
| status | String | Logger status (e.g. running, stopped) |
| interval | 32-bit Integer | Logging cycle (Unit: millisecond) |
| cron\_schedule | String | Log collection schedule in cron format |
| transformer | String | Transformer identifier |
| stop\_reason | String | Logger stop reason |
| log\_count | 64-bit Integer | Collected log count (Total count of collected logs) |
| drop\_count | 64-bit Integer | Dropped log count |
| log\_volume | 64-bit Integer | Collected log volume (Total volume of collected logs. Unit: byte) |
| drop\_volume | 64-bit Integer | Dropped log volume (Unit: byte) |
| last\_start\_at | Date | Last activated time |
| last\_run\_at | Date | Last executed time |
| last\_log\_at | Date | \_time field value of the last collected log |
| last\_write\_at | Date | Last log output time (Use case: tracking system idle time) |
| start\_time | String | Logging start time (Format: HH:mm) |
| end\_time | String | Logging end time (Format: HH:mm) |
| failure | String | Logger failure reason |

Possible failure reason in failure field is as follows:

|  |  |
| --- | --- |
| Failure reason | Description |
| USER\_REQUEST | Inactivated upon the user request. |
| SYSTEM\_REQUEST | Stopped due system request such as failover |
| LOW\_DISK | Stopped due to the lack of disk space. |
| TRANSFORMER\_DEPENDENCY | Transformer dependency is not resolved. |
| FACTORY\_DEPENDENCY | Factory dependency is not resolved. Bundle is not running. |
| STOP\_EXCEPTION | The table to store data is not found or the storage is in the read-only mode. |
| LOGGER\_EXCEPTION | Unexpected error during logging. |
| LOGGER\_DEPENDENCY | Logger dependency is not resolved. |

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| timeout | RPC request timeout exceeded. |
| disconnected | Disconnected while processing RPC request. |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Load logger lists of all sentries.

sentry | fields guid | sentry-loggers

### sentry-netstat

Retreives the current network connection status.

#### Syntax

sentry-netstat [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query command requires that the input record contains a guid field value.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |

Output Fields

This query command returns the fields below in addition to the fields from input record.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| pid | 32-bit Integer | Process ID |
| protocol | String | Protocol (tcp, tcp6, udp or udp6) |
| local\_ip | IP Address | Local IP address |
| local\_port | 32-bit Integer | Local Port |
| remote\_ip | IP Address | Remote IP address |
| remote\_port | 32-bit Integer | Remote port |
| state | String | Connection staus (LISTEN, ESTABLISHED or TIMEWAIT) |

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Retreives the network connection status of all sentries.

sentry | fields guid | sentry-netstat

### sentry-processes

Loads the process list of a sentry.

#### Syntax

sentry-processes [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query command requires that the input record contains a guid field value.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |

Output Fields

This query command returns the fields below in addition to the fields from input record.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| pid | 32-bit Integer | Process ID |
| name | String | Process name |
| cpu\_usage | 32-bit Integer | CPU usage (Unit: %) |
| working\_set | Double | Physical memory usage (Unit: byte) |

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Load process list from all sentries.

sentry | fields guid | sentry-processes

### sentry-routing-table

Loads the rounting table entry list of a sentry.

#### Syntax

sentry-routing-table [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query command requires that the input record contains a guid field value.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |

Output Fields

This query command returns the fields below in addition to the fields from input record.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| type | String | Routing entry type (Direct, Indirect) |
| protocol | String | Protocol (Local, NetMgmt) |
| destination | IP Address | Destination IP address |
| mask | IP Address | Netmask |
| forward | IP Address | Gateway IP address |
| metric | 32-bit Integer | Routing metric |

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Loads the rounting table entry lists of all sentries.

sentry | fields guid | sentry-routing-table

### sentry-top-threads

Retrieves the stack status information of the thread with highest CPU usage in a sentry. This command is useful when remotely diagnosing CPU load issue of sentry.

#### Syntax

sentry-top-threads [timeout=INT]

Parameter

**timeout=INT**

RPC timeout in seconds (default: 30 seconds)

#### Description

This query command sends asynchronous RPC request messages to 100 sentries at the same time and waits for responses.The RPC request message queue operates on a first-in, first-out (FIFO) basis. For example, if the Logpresso server needs to send RPC messages to 150 sentries, the server sends an RPC message to 100 sentries first and waits until RPC responses. If the waiting time exceeds the time specified in the timeout (default: 30 seconds), it is considered that an error has occurred on the sentry side. If 32 out of 100 sentries respond or timeout is exceeded, the Logpresso server sends an RPC message to the additional 32 sentries.

The query command outputs the results of RPC response messages in the order they are received. Depending on the load or network conditions of the host where Sentry is installed, the order of the responses may vary with each execution. Use the output data but do not rely on the order of the records.

Logpresso environment variable `logpresso.core.sentry\_rpc\_parallel` determines the length of the asynchronous RPC request message queue. The default value is `100` and you can adjust the length by modifying the value of this environment variable.

Input Field

This query command requires that the input record contains a guid field value.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| guid | String | Unique sentry identifier (Not relevant to GUID in JAVA) |

Output Field

This query command returns the fields below in addition to the fields from input record.

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| tid | 64-bit Integer | Thread ID |
| name | String | Thread name |
| state | String | Thread status (RUNNABLE, BLOCKED, TIMED\_WAITING, WAITING) |
| priority | 32-bit Integer | Execution priority (from 1 to 10, default: 5) |
| usage | 64-bit Integer | CPU usage time for the last one second (Unit: nanosecond) |
| stacktrace | String | Thread stack |

If any error occurs, this query command returns an \_error field in addition to the fields from input record. Possible errors are as follows:

|  |  |
| --- | --- |
| Error Message | Description |
| guid is null | guid field value from input record is null. |
| guid should be string | guid field value from input record is not a string. |
| guid should be non empty string | guid field value from input record is empty. |
| timeout | RPC request timeout exceeded |
| disconnected | Disconnected while processing RPC request |
| not connected | Sentry is not connected. |

Depending on the system where the sentry is installed, an RPC exception message other than the above can be issued.

#### Usage

Retrieve the stack status of the thread with the highest CPU usage from all sentries.

sentry | fields guid | sentry-top-threads

### sentryswap

Loads the data swapped in the sentry transmission queue. This is a command that can be used when a sentry is installed on the server that operates the Logpresso server.

#### Syntax

sentryswap [base=NAME]

Optional Parameter

**base=NAME**

Unique identification name for the base server connected to the sentry. If you do not specify a name, it loads the swap data of all sentries.

Base server refers to the Logpresso server (ENT, STD, and SNR) that receives logs from the sentry. The sentry can send data to different Logpresso servers. The transmission queue of the sentry must be operated separately for each base server.

#### Description

This is typically used to check the amount of data waiting in the transmission queue when the queue is full, or to immediately restore the system connection status by deleting the swap after backing up the data in the transmission queue.

Output fields are as follows.

* **\_time**: Timestamp
* **\_logger**: Log collector name
* Key-value pairs of records in the transmission buffer

You can use 'sentryswap' when Logpresso and sentry are installed on the same host. To check the swap on a host where only the sentry is installed, check it in the Logpresso shell using the 'sentry.swapStats' command.

## Threads

### system threads

Checks the thread stack and lock status of the system. Administrative privileges are required to execute this command.

#### Syntax

system threads

#### Description

Output fields are as follows.

* **tid**: Thread number
* **name**: Thread name
* **state**: Thread status
* **stacktrace**: Thread stack (the same format as jstack)

### system topthreads

Checks the thread stack under heavy load on the system. Administrative privileges are required to execute this command.

#### Syntax

system topthreads

#### Description

Output fields are as follows.

* **tid**: Thread number
* **name**: Thread name
* **state**: Thread status
* **priority**: A priority value between 1 and 10 (default: 5).
* **usage**: CPU usage time in nanoseconds
* **stacktrace**: Thread stack (the same format as jstack)

## Federation Nodes

### system nodes

Checks the status and settings of the system nodes constituting the Logpresso cluster. Administrative privileges are required to execute this command.

#### Syntax

system nodes

#### Description

You can also see the federation node status in the web console.

Output fields are as follows:

* **name**: Node name
* **description**: Node description
* **address**: Node address
* **port**: Node port number
* **failure**: Whether the node connection fails
* **paired**: Whether nodes are paired within a cluster
* **last\_alive**: The last time a response was received
* **last\_connect**: The last connected time
* **login\_name**: Login account
* **secure**: Whether it is an encrypted communication
* **skip\_cert\_check**: Whether to not validate the server certificate
* **connect\_timeout**: Connection timeout setting
* **read\_timeout**: Read timeout setting

## License

### system license-usages

Checks the license usage status of the system node. Administrative privileges are required to execute this command.

#### Syntax

system license-usages

#### Description

Output fields are as follows:

* **node**: Node name
* **volume**: License usage in bytes
* **count**: Number of log collections

# Functions

## Reference Functions

### $()

Returns the value of a query parameter.

#### Syntax

$(EXPR, [DEFAULT\_EXPR])

#### Required Parameter

**EXPR**

Expression that returns a query parameter name.

#### Optional Parameter

**DEFAULT\_EXPR**

Expression or value that specifies the default return value of the function when EXPR is null.

#### Description

A [query parameter](https://docs.logpresso.com/en/query/query-syntax) is a variable that can be called in a query or a [procedure](https://docs.logpresso.com/en/query/query-syntax). You can declare query parameters using the [set](https://docs.logpresso.com/en/query/set-command) or [evalc](https://docs.logpresso.com/en/query/evalc-command) query commands. Using a query parameter is useful when dynamically assigning values to a query and executing it; a query or procedure can reference a value of a query parameter by using the $() function.

#### Usage

Retrieve data recorded in the YOUR\_TABLE table for the last 7 days from the current time.

set from=ago("7d") | set to=str(now()) | table from=$("from") to=$("to") YOUR\_TABLE

Retrieve data recorded in the YOUR\_TABLE table for the period specified by the query parameters \_from and \_to. If the query parameters have no value (null), retrieve data for the last 1 day from the current time.

table from=$("\_from", ago("1d")) to=$("\_to", now()) YOUR\_TABLE

### field()

Receives the field name as an expression and returns the field value. It is also used to refer to the field name containing the blank.

#### Syntax

field(EXPR)

Required Parameter

**EXPR**

Expression that returns a field name

#### Usage

json "[ {'Registered No.': 1, 'Item':'Fender Precision Bass'}, {'Registered No.': 2, 'Item':'Gibson Jazz'}]" | search field("Registered No.") == 2

### whoami()

Returns the name of the account that is currently executing the query.

#### Syntax

whoami()

#### Description

The procedure is executed with owner privileges. If you call this function within a procedure, the name of the owner account is returned. The procedure is executed with owner privileges.

#### Usage

Return the current execution account name.

json "{}" | eval user=whoami() => "root"

## Type Conversion Functions

### array()

Returns an array created by evaluating all expressions specified as arguments.

#### Syntax

array(EXPR, ...)

Required Parameter

**EXPR, ...**

An expression that returns the value to be provided for each array item

#### Usage

json "{}" | eval array=array(1) => [1]json "{}" | eval array=array("hello", "world") => ["hello", "world"]json "{}"| eval array=array(21 \* 2, "the answer to life, the universe, and everything")=> [42, "the answer to life, the universe, and everything"]json "{}" | eval array=array(null) => [null]

### binary()

Encodes a string into a byte array using utf-8 or specified charset.

#### Syntax

binary(STR\_EXPR[, CHARSET])

Required Parameter

**STR\_EXPR**

Target string expression to be converted to binary

Optional Parameter

**CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the IANA Charset. <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

#### Usage

json "{}" | eval blob=binary("hello, world!") => 68656c6c6f2c20776f726c6421json "{}" | eval blob=binary(null) => null

### date()

Converts the string to date type.

#### Syntax

date(DATE\_EXPR, DATE\_FMT, [LOCALE])

Required Parameter

**DATE\_EXPR**

Source string expression to be converted to date type.

**DATE\_FMT**

Pattern letters to specify the date and time format.

|  |  |  |
| --- | --- | --- |
| Units of time | Description | Example |
| G | BC/AD | AD |
| y | Year | 1996; 96 |
| M | Month | July; Jul; 07 |
| w | Week of the year | 27 |
| W | Week of the month | 2 |
| d | Day of the year | 189 |
| D | Day of the month | 10 |
| F | Day of week number | 2 |
| E | Day of the week | Tuesday; Tue |
| u | Number of the day of the week(1=Monday, ..., 7=Sunday) | 1 |
| a | AM/PM | PM |
| H | Hour(0-23) | 0 |
| k | Hour(1-24) | 24 |
| K | AM/PM Hour starting with 0 (0-11) | 0 |
| h | AM/PM Hour starting with 1(1-12) | 12 |
| m | Minute | 30 |
| s | Second | 55 |
| S | Millisecond | 978 |
| z | Time zone | Pacific Standard Time; PST |
| Z | Time zone(RFC 822) | -0800 |
| X | Time zone(ISO 8601) | -08;-0800;08:00 |

Optional Parameter

**LOCALE**

A 2-digit or 3-digit code locale specified by ISO 639. If you do not specify the locale, it is set to en. For the ISO 639 locale code, refer to the following link: <https://iso639-3.sil.org/code_tables/639/>.

#### Description

If the DATE\_EXPR is null or an empty string, it returns null. If it is a type other than a string, it attempts to convert the date after automatically converting it to a string.

#### Usage

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()

Returns the map created by provided key-value pairs.

#### Syntax

dict(KEY, VALUE, [KEY, VALUE,] ...])

Required Parameter

**KEY, VALUE, [KEY, VALUE,] ...]**

Key and value in order.

#### Description

The key can only be a string, not null, and must not overlap with other keys. If you provide multiple keys, the value you provide later is assigned. You can provide any type as a value.

Also, if the key-value pairs do not match (when the number of parameters is odd), an error will occur.

A map is a data type used in Java, and it refers to a dictionary used in languages such as Python.

#### Usage

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()

Converts the specified string representation of a number to an equivalent double-precision floating-point number.

#### Syntax

double(STR\_EXPR)

Required Parameter

**STR\_EXPR**

Target string expression to be converted to a real number

#### Description

If the expression is null, it returns null. If the real number conversion fails, it also returns null. If the value returned by the expression is not a string, the function automatically converts the value to a string and attempts a real number conversion.

#### Usage

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

### flatten()

Takes out all elements of the recursively nested array and converts them to a flatten array. Otherwise, it returns the input value as it is. This is used to convert nested array elements to a single array before merging an array into a single string using [strjoin()](https://docs.logpresso.com/en/query/strjoin-function).

#### Syntax

flatten(ARRAY\_EXPR)

Required Parameter

**ARRAY\_EXPR**

Expression that returns the value to be converted to a single array

#### Usage

json "{}" | eval array=flatten(array(1, array(2, 3), 4))=> [ 1, 2, 3, 4 ]

### foreach()

Performs operations on an array or between multiple arrays without taking the element out of the array.

#### Syntax

foreach(OP\_EXPR, LIST\_EXPR[, ...])

Required Parameter

**OP\_EXPR**

Expression to be performed between array elements. It uses \_1 for the elements of the first array, \_2 for the elements of the second array, and \_N for the elements of the Nth array.

**LIST\_EXPR, ...**

Expressions that return an array and separate them using commas (,).

#### Description

If the lengths of the arrays passed to the parameters are not the same, the function performs an operation after adding the elements to which null is assigned in the short array according to the number of elements constituting the long array. For example, if the first array consists of five elements and the second array consists of three elements, the command performs an operation after adding two elements whose values are null to the second array.

When a scalar value is passed as an argument instead of a list, the function replicates and extends the value to a list and performs an operation according to OP\_EXPR by replacing the first list in the manner of \_1 and the second list in the manner of \_2, respectively.

#### Usage

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]

### frombase64()

Converts a BASE64 string to a byte array.

#### Syntax

frombase64(BASE64\_STR)

Required Parameter

**BASE64\_STR**

String encoded with BASE64

#### Usage

json "{}" | eval str=decode(frombase64("aGVsbG8sIHdvcmxkIQ=="))=> "hello, world!"

### fromhex()

Converts a hexadecimal string to byte array.

#### Syntax

fromhex(STR\_EXPR)

Required Parameter

**STR\_EXPR**

String to be converted to byte array. This option is case-insensitive.

#### Description

It returns null in the following situations:

* When the input value is not a hexadecimal string
* When the string length is odd

#### Usage

json "{}" | eval blob=fromhex("68656c6c6f20776f726c64")=> 68656c6c6f20776f726c64json "{}" | eval blob=fromhex("616263646") => nulljson "{}" | eval blob=fromhex("test") => nulljson "{}" | eval blob=fromhex(null) => null

### groups()

Returns items that match a given group of regular expressions in a string to an array.

#### Syntax

groups(STR\_EXPR, REGEX\_PATTERN)

Required Parameter

**STR\_EXPR**

Source string expression to be extracted

**REGEX\_PATTERN**

Regular expression with grouping

#### Usage

json "{}"| eval array=groups("Mar 29 2004 09:54:39", "(.\*?) (.\*?) (.\*?) ") => [Mar, 29, 2004]

### int()

Converts a string to an integer.

#### Syntax

int(EXPR)

Required Parameter

**EXPR**

Expression that returns a string to be converted to an integer. The argument must be one of a string, double, float, IP address, or array.

#### Description

When evaluating an expression, it works as follows:

* When the value of the expression is null, this function returns null.
* Even when a string cannot be converted to an integer, this function also returns null.
* When the value of the expression is an array, this function converts each element of the array to an integer.
* If any other type is passed as an argument, this function performs an automatic conversion and then converts it to an integer.

#### Usage

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()

Converts a string to an IP address type.

#### Syntax

ip(EXPR)

Required Parameter

**EXPR**

Expression that returns a string to be converted to an IP address. The argument type must be one of a string, int, or long.

#### Description

When evaluating an expression, it works as follows:

* When it is null, the expression returns null.
* Even when it cannot be converted to an integer, the expression also returns null.
* If any other type is passed as an argument, the expression converts it into a string and then attempts to convert it to an IP address.

IP addresses are one of the basic data types of Logpresso. The IP address starts with "/" and can represent both ipv4 and ipv6 addresses.

#### Usage

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()

Converts a string to a 64-bit integer.

#### Syntax

long(EXPR)

Required Parameter

**EXPR**

Expression that returns a string to be converted to a 64-bit integer. The argument must be one of a string, int, or IP address.

#### Description

When evaluating an expression, it works as follows:

* When it is null, the function returns null.
* Even when it cannot be converted to a 64-bit integer, the function also returns null.
* If any other type is passed as an argument, the function automatically converts it into a string and then converts it to a 64-bit integer.

#### Usage

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()

Either converts an arbitrary expression to a string, or converts a date to a string in specified date format.

#### Syntax

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])

Required Parameter

**EXPR**

Expression that returns the value to be converted to a string

**DATE\_EXPR**

Source string expression to be converted to date type

**DATE\_FMT**

Define the format to use to parse strings using the same pattern letters as the ones used in date().

|  |  |  |
| --- | --- | --- |
| Units of time | Description | Example |
| G | BC/AD | AD |
| y | Year | 1996; 96 |
| M | Month | July; Jul; 07 |
| w | Week of the year | 27 |
| W | Week of the month | 2 |
| d | Day of the year | 189 |
| D | Day of the month | 10 |
| F | Day of week number | 2 |
| E | Day of the week | Tuesday; Tue |
| u | Number of the day of the week(1=Monday, ..., 7=Sunday) | 1 |
| a | AM/PM | PM |
| H | Hour(0-23) | 0 |
| k | Hour(1-24) | 24 |
| K | AM/PM Hour starting with 0 (0-11) | 0 |
| h | AM/PM Hour starting with 1(1-12) | 12 |
| m | Minute | 30 |
| s | Second | 55 |
| S | Millisecond | 978 |
| z | Time zone | Pacific Standard Time; PST |
| Z | Time zone(RFC 822) | -0800 |
| X | Time zone(ISO 8601) | -08;-0800;08:00 |

Optional Parameter

**LOCALE**

Alphabetical abbreviations of time zones are also supported, but note that they can have ambiguous meanings. For example, CST may be Chinese Standard Time, or US Central Standard Time or Cuban Standard Time. If you do not specify a time zone, the time zone according to the locale of the system where Logpresso is installed is used. For information on time zone abbreviations, refer to Time Zone Abbreviations – Worldwide List at the following address: <https://www.timeanddate.com/time/zones/>

**Examples of time zone abbreviations**

|  |  |  |
| --- | --- | --- |
| Abbreviation | GMT offset | Description |
| UTC | GMT+0 | Coordinated Universal Time |
| KST | GMT+9 | Korea Standard Time |
| CEST | GMT+2 | Central European Summer Time |
| MSK | GMT+3 | Moscow Standard Time |
| PST | GMT-7 | Pacific Standard Time |
| EST | GMT-5 | US Eastern Standard Time |

**TIMEZONE**

You can provide the time zone in the form of GMT+09, GMT+0900, GMT+09:00, or GMT+9:00.

#### Usage

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()

Returns a binary value as a BASE64 string.

#### Syntax

tobase64(BLOB\_EXPR)

Required Parameter

**BLOB\_EXPR**

Expression to be evaluated as binary. The function returns null if it receives a non-binary value.

#### Usage

json "{}" | eval str=tobase64(binary("hello, world!"))=> "aGVsbG8sIHdvcmxkIQ=="

### tohex()

Converts a binary value to a hexadecimal string. If the parameter value is non-binary, it returns null.

#### Syntax

tohex(BLOB\_EXPR)

Required Parameter

**BLOB\_EXPR**

Binary value to be converted to hexadecimal number

#### Usage

json "{}" | eval hex=tohex(encode("abcde")) => "6162636465"json "{}" | eval hex=tohex(1234) => nulljson "{}" | eval hex=tohex(null) => null

### unique()

If the value of the expression is an array, this function returns an array of nested elements. If it takes a single value as an argument, it returns an array containing only one element.

#### Syntax

unique(EXPR)

Required Parameter

**EXPR**

Expression to return an array to remove nested elements. The order of the arrays returned at this time is not guaranteed. If the expression is a scalar value, the function returns an array containing only one element. If the expression is null, it returns null.

#### Usage

Remove duplicate elements from the 1, 1, 2, "2" array

json "{}"| eval array=unique(array(1, 1, 2, "2"))| # Return value: ["2", 1, 2]

## Type Checking Functions

### isnum()

Returns true if the argument is of a numeric type (int, short, long, float, and double), and false otherwise. If the input value is null, it returns false.

#### Syntax

isnum(EXPR)

Required Parameter

**EXPR**

Expression that returns the value to be checked.

#### Usage

json "{}" | eval bool=isnum(1) => truejson "{}" | eval bool=isnum(1.2) => truejson "{}" | eval bool=isnum("string") => falsejson "{}" | eval bool=isnum(null) => false

### isnotnull()

Returns true if the argument value is not null and false otherwise.

#### Syntax

isnotnull(EXPR)

Required Parameter

**EXPR**

Expression that returns the value to be checked.

#### Usage

json "{}" | eval bool=isnotnull(1) => truejson "{}" | eval bool=isnotnull(null) => false

### isnull()

Returns true if the argument is null and false otherwise.

#### Syntax

isnull(EXPR)

Required Parameter

**EXPR**

Expression that returns the value to be checked.

#### Usage

json "{}" | eval bool=isnull(null) => truejson "{}" | eval bool=isnull(1) => false

### isstr()

Returns true if the expression is a string, and false otherwise. If the expression is null, it returns false.

#### Syntax

isstr(EXPR)

Required Parameter

**EXPR**

Expression that returns the value to be checked.

#### Usage

json "{}" | eval bool=isstr("string") => truejson "{}" | eval bool=isstr(0) => falsejson "{}" | eval bool=isstr(null) => false

### typeof()

Returns a string indicating the type of the given expression.

#### Syntax

typeof(EXPR)

Required Parameter

**EXPR**

Expression that returns a value to check the type.

#### Description

This returns the following strings depending on the data type.

* string: String
* short: 16-bit integer
* int: 32-bit integer
* long: 64-bit integer
* float: 32-bit single-precision decimal
* double: 64-bit single-precision decimal
* bool: Boolean
* ipv4: IPv4 address
* ipv6: IPv6 address
* date: Date
* map: Map
* null: null

#### Usage

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"

## Conditional Functions

### case()

Evaluates an expression with branching based on multiple conditions.

#### Syntax

case(CONDITION\_1, EXPR\_1[, CONDITION\_2, EXPR2, ...], DEFAULT\_EXPR)

Required Parameter

**CONDITION\_1, EXPR\_1[, CONDITION\_2, EXPR2, ...]**

Specify the conditional statement in pairs with the evaluation expression (EXPR\_N) to be executed when it is true or not null.

**DEFAULT\_EXPR**

Specify the expression to be performed when none of the evaluation conditions are met.

#### Usage

Evaluate a score over 90 as A, 80 as B, 70 as C, 60 as D, and others as F.

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

When the string length of the str field is greater than 9, cut it into 9 characters and apply an ellipsis.

json "[ {'str': 'Somewhere over the rainbow'}, {'str': 'Wonderful'} ]" | eval truncated=case(len(str) > 10, concat(left(str, 10), "…"), str)

### if()

Evaluates the expression to be executed according to the evaluation result (true/false).

#### Syntax

if(CONDITION, EXPR\_TRUE, EXPR\_FALSE)

Required Parameter

**CONDITION**

Expression to evaluate. If the conditional statement is true or not null, it is evaluated as true.

**EXPR\_TRUE**

Expression to be evaluated when the CONDITION is evaluated is true.

**EXPR\_FALSE**

Expression to be evaluated when the CONDITION is evaluated is false.

#### Usage

Evaluate the status code as ok if it is 200 or as an error if it is not.

if(status == "200", "ok", "error")

### in()

Checks whether the evaluated value exists in a later set of evaluated values.

#### Syntax

in(VAL\_EXPR, EXPR[, ...])

Required Parameter

**VAL\_EXPR**

Expression to be evaluated.

**EXPR[, ...]**

Expressions to be compared with VAL\_EXPR by separating them using commas (,). If it matches one of them, the function returns true, and false otherwise.

#### Usage

Check whether the value of the user\_agent field matches one of Internet Explorer, Chrome, Safari, or Firefox.

in(user\_agent, "msie", "chrome", "safari", "firefox")

Check whether the value of the user\_agent field contains a Google or Yahoo string.

in(user\_agent, "\*google\*", "\*yahoo\*")

Check whether the level is either 6 or 7.

in(level, 6, 7)

### match()

Returns whether any part of the string matches the regular expression.

#### Syntax

match(VAL\_EXPR, REGEX)

Required Parameter

**VAL\_EXPR**

Expression to be evaluated. If the value is not a string, it converts the value to a string and then compares it with REGEX.

**REGEX**

Regular expression to compare to the VAL\_EXPR value by enclosing it in a pair of double quotes (" "). If the expression is null, the function returns false.

#### Usage

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()

If the expression to be evaluated is not null, this returns the value of the expression, and if it is null, it returns the default value.

#### Syntax

nvl(VAL\_EXPR, DEFAULT\_EXPR)

Required Parameter

**VAL\_EXPR**

Expression to be evaluated. If the evaluated value is not null, the function returns the evaluated value.

**DEFAULT\_EXPR**

Value to be returned when the VAL\_EXPR value is null.

#### Usage

json "{}" | eval nvl=nvl("hello", "") => "hello"json "{}" | eval nvl=nvl(null, "") => ""

## String Functions

### concat

Concatenates multiple strings into a single string. Expressions other than strings are first converted into strings and then concatenated.

#### Syntax

concat(EXPR, ...)

Required Parameter

**EXPR, ...**

Expressions that return strings, separated by commas (,). The function combines all strings returned by the expression into one string.

#### Usage

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

### contains()

Returns whether the target string contains a specific substring.

#### Syntax

contains(STR\_EXPR, SEARCH\_STR)

Required Parameter

**STR\_EXPR**

String expression.

**SEARCH\_STR**

String to be checked if it is included in the string value of STR\_EXPR.

#### Usage

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()

Returns a new string created using given arguments.

#### Syntax

format(STR\_FMT, PARAM[, ...])format(STR\_FMT, ARRAY\_EXPR)

Required Parameter

**STR\_FMT**

Format string including the format specifier.

For available format specifiers, refer to the Class Formatter document at the following address: <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Formatter.html>

**PARAM, ...** or ARRAY\_EXPR

Input value to be represented in a specified format. You can use an expression that returns an array (such as an expression that uses [array()](https://docs.logpresso.com/en/query/array-function) and [groups()](https://docs.logpresso.com/en/query/groups-function)) to return the arguments to be applied to the format string.

#### Usage

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()

Always creates the GUID string value and returns it.

#### Syntax

guid()

#### Usage

guid() => "a1189dda-870e-4aea-8742-76dcb8398b49"

### indexof()

Returns the position of the first occurrence of the specific pattern in the specified string. It returns -1 if the pattern is not found, and null if the string or the pattern to be matched in the string is null.

#### Syntax

indexof(STR\_EXPR, SEARCH\_EXPR[, BEGIN\_EXPR])

Required Parameter

**STR\_EXPR**

String expression.

**SEARCH\_EXPR**

String to be checked if it is included in the string value of STR\_EXPR.

Optional Parameter

**BEGIN\_EXPR**

Position index to start searching. The index indicating the position starts with 0. The function finds the SEARCH\_EXPR value from the specified position.

#### Usage

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()

Concatenates all keys and values ​​into a single string.

#### Syntax

kvjoin(KV\_DELIMIT, PAIR\_DELIMIT[, REGEX])

Required Parameter

**KV\_DELIMIT**

String that delimits the key and the value.

**PAIR\_DELIMIT**

String that delimits each key-value pair.

Optional Parameter

**REGEX**

Regular expression to match against the key. The function concatenates a key and a value only if the key matches the regular expression. If you do not specify the expression, the function concatenates all key-value pairs.

#### Usage

Combine strings by using colons (:) as key-value delimiter and ^ as a key-value pair delimiter.

json "{}" | eval name="Kim", age=30 | eval result=kvjoin(":", "^") => "name:Kim^age:30"

Concatenate strings using colons (:) as key-value delimiter and ^ as a pair delimiter by extracting only fields that match src.\* regular expression.

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()

Returns the index of the last occurrence of a substring in a given string. It returns -1 if the substring is not found; null if either the substring or the string is null.

#### Syntax

lastindexof(STR\_EXPR, SEARCH\_EXPR, [FROM\_EXPR])

Required Parameter

**STR\_EXPR**

String expression

**SEARCH\_EXPR**

Substring to search for in the string value of STR\_EXPR

Optional Parameter

**FROM\_EXPR**

Index (position) from which to start the search. The index counter starts from 0. lastindexof() returns the index of the last occurrence of a substring within a given string, searching backwards to index position 0 starting at a specified index.

The SEARCH\_EXPR string does not have to be completely between index 0 and FROM\_EXPR. As long as the first index of the SEARCH\_EXPR string is within the index range, it will be searched. In usage 5, with FROM\_EXPR set to 30 (corresponding to "L"), the range of the "Lo" string is from 30 to 31. But "L" of "Lo" is found at index 30, resulting in the return of 30.

#### Usages

Find the last index of " ".

json "{}" | eval STR\_EXPR="Life is short. Use Logpresso." | eval LAST\_INDEX=lastindexof(STR\_EXPR, " ") | # Result: LAST\_INDEX = 18

Find the index of the last occurrence of "Logpresso".

json "{}" | eval STR\_EXPR="Life is short. Use Logpresso." | eval LAST\_INDEX=lastindexof(STR\_EXPR, "Logpresso") | # Result: LAST\_INDEX = 19

When a string does not contain a given substring.

json "{}" | eval STR\_EXPR="Life is short. Use Logpresso." | eval LAST\_INDEX=lastindexof(STR\_EXPR, "Python") | # Result: LAST\_INDEX = -1

Specify the search starting position (index) and find the index of the last occurrence of a substring "Lo". Try changing the FROM\_EXPR value to get different results.

# String: Life is short. Use Logpresso. Long live Logpresso! Index: 0123456789012345678901234567890 Range: ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ (0-30) | json "{}" | eval STR\_EXPR="Life is short. Use Logpresso. Long Live Logpresso!" | eval SEARCH\_EXPR="Lo" | eval FROM\_EXPR=30 | eval LAST\_INDEX=lastindexof(STR\_EXPR, SEARCH\_EXPR, FROM\_EXPR) | # Result: LAST\_INDEX = 30

Try the function [indexof()](https://docs.logpresso.com/en/query/indexof-function) to compare its result with the lastindexof() function.

json "{}" | eval STR\_EXPR = "Life is short. Use Logpresso." | eval FIRST\_INDEX = indexof(STR\_EXPR, " ") | eval LAST\_INDEX = lastindexof(STR\_EXPR, " ") | # Result: FIRST\_INDEX = 4, LAST\_INDEX = 18

#### Compatibility

lastindexof() function is available starting from the release 4.0.2312.0.

### left()

Extracts the substring with the specified length from the left side of the given string. If the length of the string is shorter than the specified length, this returns the entire string. If the argument is null, this returns null, and if this receives a value other than the string, it evaluates the value after converting it to a string.

#### Syntax

left(STR\_EXPR, CHAR\_LENGTH)

Required Parameter

**STR\_EXPR**

String expression

**CHAR\_LENGTH**

Number of characters to be extracted. The function returns the string after truncating as many characters from the left of the string as you specified. If the length of the string is shorter than the specified length, the function returns the entire string. You can only provide constants greater than or equal to 0.

#### Usage

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()

Returns the length of the string. If the argument is null, it returns 0, and if it receives a value other than the string, it evaluates the value after converting it to a string.

#### Syntax

len(STR\_EXPR)

Required Parameter

**STR\_EXPR**

String expression

#### Usage

json "{}" | eval length=len("sample") => 6json "{}" | eval length=len(null) => 0json "{}" | eval length=len(123) => 3json "{}" | eval length=len(1.2) => 3

### lower()

Converts a string to lowercase. If the argument is null, it returns null. If this receives a value other than the string, it evaluates that value after converting it to a string.

#### Syntax

lower(STR\_EXPR)

Required Parameter

**STR\_EXPR**

String expression

#### Usage

json "{}" | eval str=lower("Hello World") => "hello world"json "{}" | eval str=lower(1234) => "1234"json "{}" | eval str=lower(null) => null

### lpad()

Creates a string of a given length by inserting padding characters to the left of the string. If the argument is null, it returns 0, and if it receives a value other than the string, it evaluates the value after converting it to a string.

#### Syntax

lpad(STR\_EXPR, OUTPUT\_LENGTH, [PADDING\_EXPR])

Required Parameter

**STR\_EXPR**

String expression

OUTPUT\_LENGTHExpression to specify the length of the result string after padding If the STR\_EXPR value is longer than OUTPUT\_LENGTH, the function cuts the string according to OUTPUT\_LENGTH and returns it.

Optional Parameter

**PADDING\_EXPR**

Expression to specify the padding character (default: whitespace).

#### Usage

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()

Finds all the specified patterns in the string, replaces them with the specified string.

#### Syntax

replace(STR\_EXPR, PATTERN, REPLACE\_WITH\_THIS[, REGEX\_FLAG])

Required Parameter

**STR\_EXPR**

Source string expression

**PATTERN**

String to search for a match. If you specify "re" as REGEX\_FLAG, you can use the regular expression to search for patterns.

**REPLACE\_WITH\_THIS**

Replacement string.

Optional Parameter

**REGEX\_FLAG**

If you provide "re" as a regular expression pattern flag, the function uses the regular expression to search for patterns.

#### Usage

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()

Returns the octets of the given IP address in reverse order. For example, if you provide 127.0.0.1, it returns 1.0.0.127. This returns null if any invalid IPv4 address string is provided.

#### Syntax

reverseip(EXPR)

Required Parameter

**EXPR**

IP address type value or an expression that returns a string in an IPv4 address format

#### Usage

Reverse the octet order of an IP address and concatenate it with .in-addr.arpa for reverse domain lookup.

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

The output fields are as follows:

* ip: 172.217.14.238
* domain: 238.14.217.172.in-addr.arpa
* status: NO\_ERROR
* answers: "PTR sea30s02-in-f14.1e100.net"

#### See Also

* [concat()](https://docs.logpresso.com/en/query/concat-function)
* [nslookup](https://docs.logpresso.com/en/query/nslookup-command)

### right()

Extracts the substring with the specified length from the right side of the given string. If the length of the string is shorter than the specified length, this returns the entire string. If the argument is null, this returns null, and if it receives a value other than the string, it evaluates that value after converting it to a string.

#### Syntax

right(EXPR, LENGTH)

Required Parameter

**STR\_EXPR**

Source string expression

**LENGTH**

Number of characters to be extracted. The function returns the string after truncating as many characters from the right of the string as you specified. If the length of the string is shorter than the specified length, the function returns the entire string. You can only provide constants greater than or equal to 0.

#### Usage

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()

Creates a string of a given length by inserting padding characters to the right of the string. If the argument is null, this returns 0, and if it receives a value other than the string, it evaluates that value after converting it to a string.

#### Syntax

rpad(STR\_EXPR, OUTPUT\_LENGTH, [PADDING\_EXPR])

Required Parameter

**STR\_EXPR**

String expression

**OUTPUT\_LENGTH**

Expression to specify the length of the result string after padding. If the STR\_EXPR value is longer than OUTPUT\_LENGTH, the function cuts the string according to OUTPUT\_LENGTH and returns it.

Optional Parameter

**PADDING\_EXPR**

Expression to specify the padding character (default: whitespace).

#### Usage

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()

Splits a string into an array of substrings based on a delimiter.

#### Syntax

split(STR\_EXPR, DELIMITER\_EXPR)

Required Parameter

**STR\_EXPR**

Source string expression

**DELIMITER\_EXPR**

String to be used as an array element separator in STR\_EXPR. The function separates the strings by using it as separators.

#### Usage

json "{'url': 'ko.logpresso.com/documents'}" | eval array=split(field("url"), "/") => ["ko.logpresso.com", "documents"]json "{}" | eval array=split("logpresso", "a") => ["logpresso"]json "{}" | eval array=split("a,b,c,d", ",") => ["a","b","c","d"]

### strjoin()

Joins the elements of the given array into a single string. The elements of the string are separated with a specified separator.

#### Syntax

strjoin(DELIMIT\_CHAR, ARRAY)

Required Parameter

**DELIMIT\_CHAR**

String to use as an element separator. If the separator contains an expression instead of a constant, a syntax error occurs.

**ARRAY**

Array whose elements are to be joined. The function returns null if the array is null, and if the element of the array is null, null is displayed in the merged string.

#### Usage

json "{}" | eval merged=strjoin(",", null) => nulljson "{}" | eval merged=strjoin(",", array(1, 2, 3)) => "1,2,3"

### substr()

Returns the substring of the source string starting at the specified position. If this receives a value other than the string, it evaluates the value after converting it to a string.

#### Syntax

substr(STR\_EXPR, START\_INDEX[, END\_INDEX])

Required Parameter

**STR\_EXPR**

Source string expression. If the expression is null, the function returns null.

**START\_INDEX**

Starting index of the substring. The index starts at 0. If the index is negative, the position is calculated from the end of the string. The function returns null if the start position is greater than the length of the string.

Optional Parameter

**END\_INDEX**

Last index of the substring. The character at the last index is not included in the returned substring. Omitting this index means that the last index of the substring is the same as the last index of the source string. If the index is negative, the position is calculated from the end of the string. If the end position is greater than the length of the string, it returns from the start position to the end of the string.

#### Usage

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()

Removes whitespaces (including tabs and newlines) from the left and right of the string. If it receives a value other than the string, it trims the value after converting it to a string.

#### Syntax

trim(STR\_EXPR)

Required Parameter

**STR\_EXPR**

Source string expression. If the expression is null, the function returns null.

#### Usage

json "{}" | eval trimed=trim(" hello world ") => "hello world"json "{}" | eval trimed=trim(123) => "123"json "{}" | eval trimed=trim(null) => null

### upper()

Converts a string to uppercases. If it receives a value other than the string, it evaluates the value after converting it to a string.

#### Syntax

upper(STR\_EXPR)

Required Parameter

**STR\_EXPR**

Source string expression. If the expression is null, the function returns null.

#### Usage

json "{}" | eval UPPER=upper("Hello World") => "HELLO WORLD"json "{}" | eval UPPER=upper(1234) => "1234"json "{}" | eval UPPER=upper(null) => null

### urldecode()

Decodes the given URL. For example, %20 is converted to a whitespace.

#### Syntax

urldecode(STR\_EXPR[, CHARSET])

Required Parameter

**STR\_EXPR**

Source string expression. If the expression is null, the function returns null.

Optional Parameter

**CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

#### Usage

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()

Translates a string into application/x-www-form-urlencoded format using a specific encoding scheme.

#### Syntax

urlencode(STR\_EXPR[, CHARSET])

Required Parameter

**STR\_EXPR**

Source string expression. If the expression is null, the function returns null.

Optional Parameter

**CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the following document: <http://www.iana.org/assignments/character-sets/character-sets.xhtml>

#### Usage

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

## Mathmatical Functions

### abs()

Returns the absolute value of any number.

#### Syntax

abs(NUM\_EXPR)

Required Parameter

**NUM\_EXPR**

Expression that returns int, short, long, float, and double. This returns null when it receives an argument value that is not a number.

#### Usage

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()

Returns the angle, in radians, whose cosine is the specified radian expression. This is also called arccosine.

#### Syntax

acos(RADIAN\_EXPR)

Required Parameter

**RADIAN\_EXPR**

Radian expression. This function returns null when it receives an argument value that is not a number.

#### Usage

json "{}" | eval acos=acos(0.866158094405463) => 0.5233333333333333 (30 \* 3.14 / 180)json "{}" | eval acos=acos(1) => 0json "{}" | eval acos=acos("0") => null

### asin()

Returns the angle, in radians, whose sine is the specified radian expression. This is also called arcsine.

#### Syntax

asin(RADIAN\_EXPR)

Required Parameter

**RADIAN\_EXPR**

Radian expression. This function returns null when it receives an argument value that is not a number.

#### Usage

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()

Returns the angle, in radians, whose tangent is the specified radian expression. This is also called arctangent.

#### Syntax

atan(RADIAN\_EXPR)

Required Parameter

**RADIAN\_EXPR**

Radian expression. This function returns null when it receives an argument value that is not a number.

#### Usage

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()

Returns the smallest integer value that is bigger than or equal to a given number. If you specify a digit after the decimal point, it rounds up from that digit. If the argument value is an integer, it returns the input as it is. This function takes only the numeric data type as an argument. If it received any other type, it returns null.

#### Syntax

ceil(NUM\_EXPR[, NUM\_DIGITS])

Required Parameter

**NUM\_EXPR**

An expression that returns int, short, long, float, or double.

**NUM\_DIGITS**

Number of digits to which you want to round up the number. If you specify a negative number as NUM\_DIGITS, the function rounds to the left of the decimal point.

#### Usage

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()

Returns the cosine of the specified angle.

#### Syntax

cos(RADIAN\_EXPR)

Required Parameter

**RADIAN\_EXPR**

Radian expression. This function returns null when it receives an argument value that is not a number.

#### Usage

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()

Returns the Euler's number(𝑒) raised to the power of the specified number. It returns null when it receives an argument value that is not a number.

#### Syntax

exp(NUM\_EXPR)

Required Parameter

**NUM\_EXPR**

Expression that returns an exponent

#### Usage

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()

Returns the largest integer that is less than or equal to a given number, Numbers with decimal places are rounded down to the nearest integer by default. This function takes only the numeric data type as an argument. If an integer comes in as an argument, this returns the input as it is. If it receives any other type, it returns null.

#### Syntax

floor(NUM\_EXPR[, NUM\_DIGITS])

Required Parameter

**NUM\_EXPR**

Expression that returns int, short, long, float, or double.

**NUM\_DIGITS**

Number of digits to which you want to round down the number. If you specify a negative number as NUM\_DIGITS, the function rounds down to the left of the decimal point.

#### Usage

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()

Returns a natural logarithm of the specified number. If a negative number is passed to the parameter, it returns NaN (Not a Number). It returns null when it receives an argument value that is not a number.

#### Syntax

log(NUM\_EXPR)

Required Parameter

**NUM\_EXPR**

Expression that returns int, short, long, float, or double.

#### Usage

json "{}" | eval log=log(10) => 2.302585092994046

### log10()

Returns the base-10 logarithm of the specified number. If a negative number is passed to the parameter, this returns NaN (Not a Number). It returns null when it receives an argument value, not a number.

#### Syntax

log10(NUM\_EXPR)

Required Parameter

**NUM\_EXPR**

Expression that returns int, short, long, float, or double.

#### Usage

json "{}" | eval result=log10(10) => 1json "{}" | eval result=log10(100) => 2json "{}" | eval result=log10(-10) => NaN

### max()

Returns the maximum value of the given expression. It ignores any expression that is null.

#### Syntax

max(NUM\_EXPR, ...)

Required Parameter

**NUMBER, ...**

Expression list that returns int, short, long, float, or double

#### Usage

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()

Returns the minimum value of the given expression. It ignores any expression that is null.

#### Syntax

min(NUM\_EXPR, ...)

Required Parameter

**NUMBER, ...**

Expression list that returns int, short, long, float, or double

#### Usage

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()

Returns the remainder after the number is divided by divisor.

#### Syntax

mod(NUM\_EXPR, DIVISOR)

Required Parameter

**NUM\_EXPR**

Expression that returns int or long.

**DIVISOR**

Divisor expression

#### Usage

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()

Returns the result of a number raised to a power.

#### Syntax

pow(NUM\_EXPR, POWER)

Required Parameter

**NUM\_EXPR**

Expression that returns a base number.

**POWER**

Expression that returns a power exponent.

#### Usage

json "{}" | eval pow=pow(2, 0) => 1json "{}" | eval pow=pow(2, 1) => 2json "{}" | eval pow=pow(2, 2) => 4

### round()

Rounds the number to the specified number of digits. It returns null if an argument value or the digit cannot be processed.

#### Syntax

round(NUM\_EXPR[, NUM\_DIGITS])

Required Parameter

**NUM\_EXPR**

Expression that returns int, short, long, float, or double

**NUM\_DIGITS**

Number of digits to which you want to round the number argument. If you specify a negative number as NUM\_DIGITS, the function rounds to the left of the decimal point

#### Usage

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()

Returns a number that increases sequentially from 1 each time it is called.

#### Syntax

seq()

### sin()

Returns the sine of the specified angle.

#### Syntax

sin(RADIAN\_EXPR)

Required Parameter

**RADIAN\_EXPR**

Radian expression. This function returns null when it receives an argument value that is not a number.

#### Usage

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()

Returns the square root of the specified number. If a negative number is passed to the parameter, this returns NaN (Not a Number). It returns null when it receives an argument value that is not a number.

#### Syntax

sqrt(NUM\_EXPR)

Required Parameter

**NUM\_EXPR**

Expression that returns int, short, long, float, or double

#### Usage

sqrt(4) => 2sqrt(-1) => NaN

### tan()

Returns the tangent of the specified angle.

#### Syntax

tan(RADIAN\_EXPR)

Required Parameter

**RADIAN\_EXPR**

Radian expression. This function returns null when it receives an argument value that is not a number.

#### Usage

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

## Time Functions

### ago()

Subtracts time (years, months, weeks, days, hours, minutes, and seconds) from the current date time.

#### Syntax

ago("NUM{y|mon|w|d|h|m|s}")

Required Parameter

**NUM{y|mon|w|d|h|m|s}**

Time in units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second).

#### Usage

This is an example based on 2019-04-26 14:31:21.

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()

Adds time (years, months, weeks, days, hours, minutes, and seconds) to the current date-time.

#### Syntax

dateadd(DATE, "{year|mon|day|hour|min|sec|msec}", INT)

Required Parameter

**DATE**

Expression that returns the time type.

**"{year|mon|day|hour|min|sec|msec}"**

Time unit to add to the INT value, enclosed in a pair of double quotes(" "). For the meaning of each unit of time, refer to the table below.

**Unit of Time**

|  |  |
| --- | --- |
| Unit of Time | Description |
| year | Year |
| mon | Month |
| day | Day |
| hour | Hour |
| min | Minute |
| sec | Second |
| msec | Millisecond |

**INT**

Integer value to add to the date in the given units.

#### Usage

json "{}" | eval \_time = date("2013-09-28 11:47:00", "yyyy-MM-dd HH:mm:ss"), add\_1\_year = dateadd(date("2013-09-28 11:47:00", "yyyy-MM-dd HH:mm:ss"), "year", 1), subtract\_1\_mon = dateadd(date("2013-09-28 11:47:00", "yyyy-MM-dd HH:mm:ss"), "mon", -1), subtract\_3\_days = dateadd(date("2013-09-28 11:47:00", "yyyy-MM-dd HH:mm:ss"), "day", -3), add\_2\_hours = dateadd(date("2013-09-28 11:47:00", "yyyy-MM-dd HH:mm:ss"), "hour", 2), input\_null = dateadd(null, "sec", 10), input\_str = dateadd("invalid", "sec", 10)

### datediff()

Returns the difference between the start date and end date in the specified time unit.

#### Syntax

datediff(START\_DATE, END\_DATE, "{year|mon|day|hour|min|sec|msec}")

Required Parameter

**START\_DATE**

Expression that returns the start date. The function returns null if a value of any other type is received.

**END\_DATE**

Expression that returns the last date. The function returns null if a value of any other type is received.

**"{year|mon|day|hour|min|sec|msec}"**

Time unit to use when calculating the difference between START\_DATE and END\_DATE, enclosed in a pair of double quotes. For the meaning of each unit of time, refer to the table below.

**Unit of Time**

|  |  |
| --- | --- |
| Unit of Time | Description |
| year | Year |
| mon | Month |
| day | Day |
| hour | Hour |
| min | Minute |
| sec | Second |
| msec | Millisecond |

#### Usage

Calculate the difference between September 29, 2014, and September 29, 2013.

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")

In the case of an incorrect input

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()

Returns an integer representing the specific part (century, year, month, day, day of the week, and the rest) of the given date.

#### Syntax

datepart(DATE, DATEPART)

Required Parameter

**DATE**

Expression that returns a date type value. The function returns null if a value of any other type is received.

**DATEPART**

Constant string representing the part of the date argument to return. For a list of time units, refer to the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| Units of time | Type | Description | Example |
| century | int | Century | 21 |
| day | int | Date (1–31) | 12 |
| decade | int | Portion of the year divided by 10 | 201 |
| dow | int | Number of days in a week. Sunday (0), Saturday (6) | 1 |
| doy | int | Number of days in a year | 163 |
| epoch | long | Seconds elapsed from January 1, 1970, to the date | 1497269156 |
| hour | int | 24-hour standard time (0–23) | 21 |
| isodow | int | Number of days in a week based on ISO 8601. Monday (1), Sunday (7) | 1 |
| isoyear | int | The year in which the first Monday of a year is recognized as the first day of the new year (ISO 8601) | 2017 |
| microseconds | int | Microseconds including seconds | 56371000 |
| millenium | int | Millennium (in millennia) | 3 |
| milliseconds | int | Milliseconds including seconds | 56371 |
| min, minute | int | Minute (0–59) | 5 |
| mon, month | int | Month (1–12) | 6 |
| msec | int | Milliseconds not including seconds | 377 |
| quarter | int | Quarter (1–4) | 2 |
| sec, seconds | int | Second (0–59) | 56 |
| timezone | int | UTC standard time zone (second) | 32400 |
| timezone\_hour | int | UTC standard time zone (hour) | 9 |
| timezone\_minute | int | UTC standard time zone (minute) | 0 |
| week | int | Number of weeks in which the first Monday of a year is recognized as the first day of the new year based (ISO 8601) | 24 |
| year | int | Year | 2017 |

#### Usage

json "{}"| eval time= datepart( date("Jun 1 2020 12:34:56", "MMM dd yyyy HH:mm:ss", "ko"), "year") => 2020json "{}"| eval time= datepart( date("Jun 1 2020 12:34:56", "MMM dd yyyy HH:mm:ss", "ko"), "mon") => 6json "{}"| eval time=datepart( date( "Jun 1 2020 12:34:56", "MMM dd yyyy HH:mm:ss", "ko"), "epoch") => 1590982496

### daterange()

Generates a list of dates with the specified interval between the start and end dates. It does not include the end date in the list to be returned.

#### Syntax

daterange(START\_DATE, END\_DATE, [INTERVAL{y|mon|w|d|h|m|s}])

Required Parameter

**START\_DATE**

Expression that returns the start date. The function returns null if a value of any other type is received.

**END\_DATE**

Expression that returns the end date. The function returns null if a value of any other type is received.

Optional Parameter

**INTERVAL{y|mon|w|d|h|m|s}**

Time interval in a unit of time such as y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second) (default: 1d).

To avoid system overload, if the result of the 'daterange()' function exceeds 100,000, it causes an exception to fail the query.

#### Usage

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()

Truncates a date-time value to the specified time unit.

#### Syntax

datetrunc(DATE, INT{y|mon|w|d|h|m|s})

Required Parameter

**DATE**

Expression that returns a date type value. If it receives a value of a different type, it returns null.

**INT**

Time in units of y (year), mon (month), w (week), d (day), h (hour), m (minute), and s (second).

#### Usage

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()

Receives seconds or milliseconds that have elapsed since January 1, 1970, and converts them to a date type. This interprets the parameter value as milliseconds if the parameter value is interpreted as seconds to be greater than 1 January 9999.

#### Syntax

epoch(NUM\_EXPR)

Require Parameter

**NUM\_EXPR**

Expression that returns the natural number in seconds or milliseconds.

#### Usage

json "{}" | eval time=epoch(1435196373492) => 2015-06-25 10:39:33+0900

### now()

Returns the system time at which the function executes.

#### Syntax

now()

#### Usage

json "{}" | eval time=now() => Sat Sep 28 23:58:41 KST 2013

## IP Address Functions

### ip2int()

Converts an IPv4 address object or IPv4 address string to a signed integer.

#### Syntax

ip2int(IP4\_ADDR)

Required Parameter

**IP4\_ADDR**

Expression that returns an IPv4 address string or IPv4 address object. The function returns null if it receives an invalid value of any other type.

#### Usage

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()

Converts any IPv4 address string to a numeric (long) type.

#### Syntax

ip2long(IP4\_ADDR)

Required Parameter

**IP4\_ADDR**

IPv4 address string expression. The function returns null if it receives an invalid value of any other type.

#### Usage

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()

Converts any integer to an IPv4 address string.

#### Syntax

long2ip(LONG\_INT)

Required Parameter

**LONG\_INT**

Expression that returns a long integer. The function returns null if it receives an invalid value of any other type.

#### Usage

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()

Returns the network address value to the given IPv4/IPv6 address and CIDR.

#### Syntax

network(IP\_ADDR, CIDR)

Required Parameter

**IP\_ADDR**

String or IP address of IPv4 and IPv6. IPv6 addresses are case-insensitive.

**CIDR**

CIDR value (integer). The CIDR value range is from 0 to 32 for IPv4 or 0 to 128 for IPv6.

#### Usage

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

## Encryption & Encoding Functions

### decode()

Converts a binary value to a string based on the specified encoding.

#### Syntax

decode(BLOB\_EXPR[, CHARSET])

Required Parameter

**BLOB\_EXPR**

Expression that returns a binary value. The function returns null if a non-binary value is received.

Optional Parameter

**CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the IANA Charset. <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

#### Usage

json "{}" | eval encoded=encode("hello, world!"), decoded=decode(encoded) => encoded: 68656c6c6f2c20776f726c6421 # Binary => decoded: "hello, world!" # String

### decrypt()

Decrypts a encrypted binary value using the Cipher class provided by Java.

#### Syntax

decrypt(CIPHER, KEY, DATA, [IV])

Required Parameter

**CIPHER**

Expression that returns a string in the form of algorithm/mode/padding. If you omit the mode and padding and provide only the algorithm, the default cryptographic algorithm is applied:

* If you provide only AES, AES/CBC/NoPadding are used.
* If you provide only RSA, RSA/ECB/PKCS1Padding are used.

For the algorithms, modes, and paddings available, refer to the Java Security Standard Algorithm Names document: <https://docs.oracle.com/en/java/javase/11/docs/specs/security/standard-names.html>

* You can see the algorithms in the **Cipher Algorithm Names** section.
* You can see the modes in the **Cipher Algorithm Modes** section.
* You can see the paddings in the **Cipher Algorithm Paddings** section.

Logpresso supports various cryptographic algorithms provided by Java for compatibility. However, it is recommended not to use insecure cryptographic algorithms or modes such as DES or ECB. For information on cipher classes used in Java, refer to the documentation at the following address: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/Cipher.html

The following are the formats that all Java implementations must support. DES, DESede algorithms, and ECB modes are insecure and should only be used when there are compatibility issues with external systems. The numbers in parentheses indicate the key length in bits. - 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**

Binary key of a size that matches the specified cryptographic algorithm. The key length according to the algorithm is as follows:

* AES: 16 characters (128 bits), 24 characters (192 bits), and 32 characters (256 bits).
* RSA: 128 characters (1,024 bits) and 256 characters (2,048 bits).
* The encryption key length is the value obtained by dividing the key length required by the encryption algorithm into 8 bytes.

**DATA**

Binary data to decrypt.

Optional Parameter

**IV**

Binary value as an Initial Vector(IV) required when operating mode such as CBC.

#### Usage

json "{}" | eval decrypted= decode( decrypt("AES",frombase64("mRcOlK9V47rjVL/RBYQYRw=="), frombase64("y7+NQQ9/9xGtbBq5pgBvCA==") )) => "hello, world!" # Compare with the usages of encrypt ().

### encode()

Encodes a string to a binary object based on the specified encoding.

#### Syntax

encode(STR, [CHARSET])

Required Parameter

**STR**

Expression that returns a string. The function returns null if a non-string value is received.

Optional Parameter

**CHARSET**

Character set (default: utf-8). Use the preferred MIME name or aliases registered in the IANA Charset. <https://www.iana.org/assignments/character-sets/character-sets.xhtml>

#### Usage

json "{}" | eval encoded=encode("hello, world!"), decoded=decode(encoded)=> encoded: 68656c6c6f2c20776f726c6421 # Binary=> decoded: "hello, world!" # String

### encrypt()

Encrypts the binary value using the specified algorithm and key.

#### Syntax

encrypt(CIPHER, KEY, DATA[, IV])

Required Parameter

**CIPHER**

Expression that returns a string in the form of algorithm/mode/padding. If you omit the mode and padding and provide only the algorithm, the default cryptographic algorithm is applied:

* If you provide only AES, AES/CBC/NoPadding are used.
* If you provide only RSA, RSA/ECB/PKCS1Padding are used.

For the algorithms, modes, and paddings available, refer to the Java Security Standard Algorithm Names document: <https://docs.oracle.com/en/java/javase/11/docs/specs/security/standard-names.html>

* You can see the algorithms in the **Cipher Algorithm Names** section.
* You can see the modes in the **Cipher Algorithm Modes** section.
* You can see the paddings in the **Cipher Algorithm Paddings** section.

Logpresso supports various cryptographic algorithms provided by Java for compatibility. However, it is recommended not to use insecure cryptographic algorithms or modes such as DES or ECB.For information on cipher classes used in Java, refer to the documentation at the following address: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/javax/crypto/Cipher.html

The following are the formats that all Java implementations must support. DES, DESede algorithms, and ECB modes are insecure and should only be used when there are compatibility issues with external systems. The numbers in parentheses indicate the key length in bits.- 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**

Binary key of a size that matches the specified cryptographic algorithm. The key length according to the algorithm is as follows:

* AES: 16 characters (128 bits), 24 characters (192 bits), and 32 characters (256 bits).
* RSA: 128 characters (1,024 bits) and 256 characters (2,048 bits).

The encryption key length is the value obtained by dividing the key length required by the encryption algorithm into 8 bytes.

**DATA**

Binary data to encrypt.

Optional Parameter

**IV**

Binary value as an Initial Vector(IV) required in operating mode such as CBC.

#### Usage

json "{}" | eval encrypted=tobase64( encrypt("AES", frombase64("mRcOlK9V47rjVL/RBYQYRw=="), binary("hello, world!") )) => "y7+NQQ9/9xGtbBq5pgBvCA==" # Refer to the usage of decrypt ().

### hash()

Returns the result of performing the unidirectional hash algorithm as a binary value.

#### Syntax

hash(HASH\_ALGO, BIN\_DATA)

Required Parameter

**HASH\_ALGO**

Hash algorithm (md5, sha1, sha256, sha384, and sha512).

**BIN\_DATA**

Data to be hashed. The data MUST be in binary form. The function returns null if a non-binary value is received.

#### Usage

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()

Returns any integer greater than or equals to 0 and less than the specified boundary value.

#### Syntax

rand(NUM[, SEED])

Required Parameter

**NUM**

Integer greater than 0.

Optional Parameter

**SEED**

Integer value. If you provide this value, the function always returns a fixed value, not a random integer value.

The SEED parameter is used to return a value whenever a query command is executed for purposes such as debugging query commands that use this function and testing to verify the behavior of query commands. Be aware of use in the actual operating environment.

#### Usage

json "{}" | eval rand=rand(1000) => Any value from 0 to 999

### randbytes()

Generates random bytes of the specified length.

#### Syntax

randbytes(LEN)

Required Parameter

**LEN**

Length of the binary. Only lengths from 1 to 10,240 are allowed.

#### Usage

json "{}" | eval rand\_blob=randbytes(32) => 6765eab83af980a266461427739cc37a8b4ee60dab09b091282c1070a3191e2d # Binary value example

### flatten()

Takes out all elements of the recursively nested array and converts them to a flatten array. Otherwise, it returns the input value as it is. This is used to convert nested array elements to a single array before merging an array into a single string using [strjoin()](https://docs.logpresso.com/en/query/strjoin-function).

#### Syntax

flatten(ARRAY\_EXPR)

Required Parameter

**ARRAY\_EXPR**

Expression that returns the value to be converted to a single array

#### Usage

json "{}" | eval array=flatten(array(1, array(2, 3), 4))=> [ 1, 2, 3, 4 ]

### foreach()

Performs operations on an array or between multiple arrays without taking the element out of the array.

#### Syntax

foreach(OP\_EXPR, LIST\_EXPR[, ...])

Required Parameter

**OP\_EXPR**

Expression to be performed between array elements. It uses \_1 for the elements of the first array, \_2 for the elements of the second array, and \_N for the elements of the Nth array.

**LIST\_EXPR, ...**

Expressions that return an array and separate them using commas (,).

#### Description

If the lengths of the arrays passed to the parameters are not the same, the function performs an operation after adding the elements to which null is assigned in the short array according to the number of elements constituting the long array. For example, if the first array consists of five elements and the second array consists of three elements, the command performs an operation after adding two elements whose values are null to the second array.

When a scalar value is passed as an argument instead of a list, the function replicates and extends the value to a list and performs an operation according to OP\_EXPR by replacing the first list in the manner of \_1 and the second list in the manner of \_2, respectively.

#### Usage

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()

Returns the subarray of a given array.

#### Syntax

subarray(ARRAY\_EXPR, INT\_START, [INT\_END])

**ARRAY\_EXPR**

Expression that retruns an array

**INT\_START**

Start index of the array to be sliced. Index numbers start from 0.

**INT\_END**

Last index of the array to be sliced. The element in the last index is not included in the subarray.

#### Description

Index of the array start from 0. If an array contains 5 elements, index numbers are 0, 1, 2, 3, 4 from left to right. You can also use negative number to index the elements. If an array contains 5 elements, index number is -5, -4, -3, -2, -1 from left to right.

#### Usage

Typical example usages.

json "{}" | eval parent=array(1, 2, 3, 4, 5) | eval child=subarray(parent, 2) | # Return value: parent: [1, 2, 3, 4, 5] child: [3, 4, 5]

json "{}" | eval arr=subarray(array(1, 2, 3, 4, 5), 2, 4) | # Return value: arr: [3, 4]

json "{}" | eval arr=subarray(array(1, 2, 3, 4, 5), 1, -1) | # Return value: arr: [2, 3, 4]

When INT\_START or INT\_END is specified outside valid range.

json "{}" | eval arr=subarray(array(1, 2, 3, 4, 5), 5) | # Return value: null

json "{}" | eval arr=subarray(array(1, 2, 3, 4, 5), 0, 5) | # Return value: arr: [1, 2, 3, 4, 5]

### sumarray()

Returns the sum of all elements in a given array. sumarray() function will always exclude the null and non-numeric value.

#### Syntax

subarray(ARRAY\_EXPR)

**ARRAY\_EXPR**

Expression that returns an array

#### Usage

json "{}"| eval sum=sumarray(array(1, 2, 3, 4, 5))| # Return value: 15

json "{}"| eval arr=sumarray(array(1, 2, 3, null, "a", 4, 5))| # Retrun value: 15

json "{}"| eval arr=sumarray(array(null, null, null, "a", "b"))| # Return value: 0

### unique()

If the value of the expression is an array, this function returns an array of nested elements. If it takes a single value as an argument, it returns an array containing only one element.

#### Syntax

unique(EXPR)

Required Parameter

**EXPR**

Expression to return an array to remove nested elements. The order of the arrays returned at this time is not guaranteed. If the expression is a scalar value, the function returns an array containing only one element. If the expression is null, it returns null.

#### Usage

Remove duplicate elements from the 1, 1, 2, "2" array

json "{}"| eval array=unique(array(1, 1, 2, "2"))| # Return value: ["2", 1, 2]

### valueof()

Returns the value at the location corresponding to a specific key or index in the array or composite object specified as a parameter

#### Syntax

valueof(COMPOUND\_OBJ\_EXPR, KEY\_EXPR)

Required Parameter

**COMPOUND\_OBJ\_EXPR**

Expression that returns a composite object such as an associative array or an array

**KEY\_EXPR**

Expression that points to a value at a specific location, such as a key string in an associative array or the index number of an array

#### Description

This function returns a value corresponding to a specific key in the associative array or the array. It returns null for the following exceptions:

* When you provide an object other than an associative array or an array in a composite object expression
* When the key of the associative array and the type of the key expression do not match
* When the index number of the array and the type of key expression do not match

#### Usage

Extract the item number 2 from an array with three elements (the index number in the array starts from 0).

json "{}" | eval foods=array("Apple","Banana","Cucumber") | eval food=valueof(foods,2) => "Cucumber"

Extract an item with the key b from a map object.

json "{}" | eval foods=dict("a","Apple","b","Banana","c","Cucumber" ) | eval food = valueof(foods,"b") => "Banana"

## CEP Functions

### evtctxget()

Loads attribute information by selecting the event context associated with the key. If no event context exists with the specified key, it returns null.

#### Syntax

evtctxget(TOPIC, KEY, ATTR)

Required Parameter

**TOPIC**

Topic of the event context.

**KEY**

Unique key that distinguishes the event context.

**ATTR**

Attribute string can be one of the following strings.

* counter: returns an integer indicating the number of times an event has occurred for the same key. It always returns a value greater than or equal to 1.
* created: returns a date value indicating the time when the event first occurred.
* expire: If the expiration date is specified, this returns a date value indicating the time at which the event context is deleted.
* timeout: If the timeout is specified, tthis returns a date value indicating the time at which the context of the event is deleted due to the timeout.
* rows: returns an array of all records currently stored in the event context.

#### Usage

When there is an event context with a topic of txmatch, 001122 of txkey, and a timeout of 10 seconds

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()

Loads the user variable in the event context associated with the key. If no event context exists with the specified key or no variable exists, it returns null.

#### Syntax

evtctxgetvar(TOPIC, KEY, VARIABLE)

**TOPIC**

Topic of the event context.

**KEY**

Unique key that distinguishes the event context.

**VARIABLE**

User variable to load.

#### Usage

Use the **sessionkey** field value as the identifier of the web\_session context and load the client\_ip variable

evtctxgetvar("web\_session", sessionkey, "client\_ip")

### evtctxsetvar()

Sets the user variable in the event context associated with the key. It returns false if the event context does not exist or the result of evaluating the variable name expression is null, and returns true if the variable setting is successful.

#### Syntax

evtctxsetvar(TOPIC, KEY, VARIABLE, VALUE)

**TOPIC**

Topic of the event context.

**KEY**

Unique key that distinguishes the event context

**VARIABLE**

User variable

**VALUE**

Expression that returns the value to be assigned to a variable

#### Usage

Use the **sessionkey** field value as the identifier of the web\_session context and set the client\_ip variable as an **ip** field value.

evtctxsetvar("web\_session", sessionkey, "client\_ip", ip)

## Sonar Functions

### matchbehavior()

Searches for behavior profiles and returns true if there is any matching profiled data, and false otherwise.

#### Syntax

matchbehavior(PROFILE\_GUID, KEY\_EXPR[, ...])

Required Parameter

**PROFILE\_GUID**

Behavior profile GUID. The GUID string must be a valid behavior profile identifier. If you specify an invalid behavior profile GUID, the query fails.

**KEY\_EXPR, ...**

Key expressions that use a comma (,) as a separator. The order of key parameters must be the same as the order of key fields specified in the behavior profile. The number of key parameters must be the same as the number of key fields specified in the behavior profile settings. Only a string or IP address is allowed for the evaluated value of the key expression. If an unauthorized type is passed as an argument, it returns a false value.

### matchblackip()

Returns true if the IP blacklist contains a target value, and false otherwise.

#### Syntax

matchblackip(BLACKLIST\_GUID, IP\_ADDR\_EXPR)

Required Parameter

**BLACKLIST\_GUID**

IP blacklist GUID. The GUID string must be a valid IP blacklist identifier. If you specify an invalid IP blacklist GUID, the query fails.

**IP\_ADDR\_EXPR**

IP address expression. Only an IPv4 address or IPv4 string is allowed in the target expression. If it is a type not allowed in the target expression, this returns a false value.

### matchfeed()

Returns true if the threat intelligence feed contains a target value, and false otherwise.

#### Syntax

matchfeed(FEED\_ID, STR\_EXPR)

Required Parameter

**FEED\_ID**

Identifier of the threat intelligence feed. If you specify an invalid feed string constant, the query fails.

See the following table for available identifiers. In addition, you can use the feeds provided by apps installed on Logpresso Sonar or Logpresso Maestro.

|  |  |  |
| --- | --- | --- |
| FEED\_ID | Type | Description |
| otx | ip | Real-time IP address reputation feed in the format of OTX (Open Threat Exchange) |
| tor | ip | Real-time Tor exit node IP address information feed |
| mdl\_domain | domain | Real-time malicious domain name (e.g. C&C domain) |
| mdl\_ip | ip | Real-time malicious domain name (e.g. C&C IP address) |
| abusech | domain | Real-time malicious domain name (e.g. C&C domain) feed provided by abuse.ch |
| malc0de | md5 | Real-time Malware database provided by malc0de.com |

**STR\_EXPR**

Expression to return the string to be searched in the feed

### matchnet()

Returns true if an IP subnet contains an IP address, and false otherwise.

#### Syntax

matchnet(NET\_GUID, IP\_ADDR\_EXPR)

Required Parameter

**NET\_GUID**

IP subnet GUID. The GUID string must be a valid IP subnet identifier. If you specify an invalid GUID, the query fails.

**IP\_ADDR\_EXPR**

IP address expression. Only an IPv4 address or IPv4 strings are allowed in the target expression. If it is a type not allowed in the target expression, the function returns a false.

### matchport()

Returns true if the specified port group contains a combination of a port and a protocol and false otherwise.

#### Syntax

matchport(PORT\_GUID, PORT\_EXPR[, PROTO\_EXPR])

Required Parameter

**PORT\_GUID**

Port group GUID. The GUID string must be a valid port group identifier. If you specify an invalid port group GUID, the query fails.

**PORT\_EXPR**

Expression to specify the port number. The value must be an integer between 0 and 65535. The function returns false if the expression cannot be evaluated or the value is not valid.

Optional Parameter

**PROTO\_EXPR**

Expression to specify the protocol. Only TCP or UDP strings are allowed. The function returns false if the expression cannot be evaluated or the value is not valid. When the expression is null, it returns true if the port group contains either TCP or UDP.

### matchsig()

Returns true if the string matches one or more of the patterns in the pattern group, and false if no pattern is matched.

#### Syntax

matchsig(SIG\_GUID, STR\_EXPR)

Required Parameter

**SIG\_GUID**

Pattern group GUID. The GUID string must be a valid pattern group identifier. If you specify an invalid pattern group GUID, the query fails.

**STR\_EXPR**

Expression to return the string to be searched. The evaluated value must be a string. If the expression cannot be evaluated or the value is not valid, the function returns false.

#### Description

For example, if you set pattern "REMOTE\_ADDR" and ("fputs" or "fwrite"), filter expression path == "lib.php", rule zb now connect, the command checks whether the REMOTE\_ADDR string and the fputs or fwirte string are searched at the same time in the target field value, and then checks whether the path field value matches the lib.php string.

**Example of patterns**

|  |  |  |
| --- | --- | --- |
| expr (required) | expr2 (optional) | rule (required) |
| Keyword pattern: Primary high-speed detection | Boolean expression: Secondary filtering | Pattern name |
| "addextendedproc" and "xp\_cmdshell" |  | xp\_cmdshell |
| "REMOTE\_ADDR" and ("fputs" or "fwrite") | path == "lib.php" | zb now\_connect |

# Aggregate Functions

### array()

Creates an array of values that belong to the group. It collects up to 100 items for each group and creates an array as a set that may also have duplicate values.

If you want to extract all the unique sets of values that belong to a group, use the [values()](https://docs.logpresso.com/en/query/values-aggregate-function) function.

#### Syntax

array(EXPR)

Required Parameter

**EXPR**

Expression of any type

### avg()

Calculates the average of all expressions that belong to the group. It eliminates expression which returns null or a non-numeric value.

#### Syntax

avg(NUM\_EXPR)

Required parameter

**NUM\_EXPR**

Numeric expression

#### Usage

# Create 2 fields with 100 random records with numbers in the range of 1 to 100 and calculate the average.| json "{}" | repeat count=100 | eval n1=rand(101), n2=rand(101) | stats avg(n1) as n1\_avg, avg(n2) as n2\_avg

### corr()

Calculates the Pearson correlation coefficient for each group. It eliminates expression pairs where either expression in the pair returns null or a non-numeric value.

#### Syntax

corr(EXPR\_X, EXPR\_Y)

Required Parameter

**EXPR\_X**

Numeric expression

**EXPR\_Y**

Second numeric expression

#### Usage

json "{}" | repeat count=100| eval n1=seq(), n2=sqrt(seq()) | stats corr(n1, n2)

### count()

Count the number of rows in each group. If no expression is specified, it returns the total number of rows. If an expression is specified, it returns the number of rows with a non-null value.

#### Syntax

countcount(EXPR)

Optional Parameter

**EXPR**

An expression of any type

### cov()

Calculates the covariance for each group. It eliminates expression pairs where either expression in the pair returns null or a non-numeric value.

#### Syntax

cov(EXPR\_X, EXPR\_Y)

Required Parameter

**EXPR\_X**

Numeric expression

**EXPR\_Y**

Second numeric expression

### dc()

Count the number of unique values that belong to the group.

#### Syntax

dc(EXPR)

Required Parameter

**EXPR**

Expression of any type

### estdc()

Estimates distinct count of the values that belong to the group.

#### Syntax

estdc(EXPR[, BIT])

Required Parameter

**EXPR**

Expression of any type

Optional Parameter

**BIT**

Bit number between 4 and 24 (defaut: 16). The higher the bit number, the higher the accuracy and memory usage.

### first()

Returns the first value in a group.

#### Syntax

first(EXPR)

Required Parameter

**EXPR**

Expression of any type

### last()

Returns the last value in a group.

#### Syntax

last(EXPR)

Required Parameter

**EXPR**

Expression of any type

### max()

Returns the maximum value in the expression. It ignores an expression that returns null. Comparing different types results in undefined behavior.

#### Syntax

max(EXPR)

Required Parameter

**EXPR**

Expression of any type

### min()

Returns the minimum value in the expression. It ignores an expression that is null. Comparing different types results in undefined behavior.

#### Syntax

min(EXPR)

Required Parameter

**EXPR**

Expression of any type

### slope()

Calculates the slope of the X/Y linear regression line by group. It eliminates expression pairs where either expression in the pair returns null or a non-numeric value.

#### Syntax

slope(EXPR\_X, EXPR\_Y)

Required Parameter

**EXPR\_X**

Numeric expression to calculate the slope of the linear regression line.

**EXPR\_Y**

Second numeric expression to calculate the slope of the linear regression line.

### stddev()

Calculates the standard deviation of all expressions that belong to the group. It eliminates expression pairs where either expression in the pair returns null or a non-numeric value.

#### Syntax

stddev(NUM\_EXPR)

Required Parameter

**NUM\_EXPR**

Numeric expression

### sum()

Calculates the sum of all expressions that belong to the group. It eliminates expression pairs where either expression in the pair returns null or a non-numeric value.

#### Syntax

sum(NUM\_EXPR)

Required Parameter

**NUM\_EXPR**

Numeric expression

### values()

Extracts a set of all the unique values that belong to the group. It collects up to 100 items for the group and creates an array as a set of unique values.

#### Syntax

values(EXPR)

Required Parameter

**EXPR**

Expression of any type

### var()

Calculates the variance of all expressions that belong to the group. It eliminates expression pairs where either expression in the pair returns null or a non-numeric value.

#### Syntax

var(NUM\_EXPR)

Required Parameter

**NUM\_EXPR**

Numeric expression