### 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.