mfmod_openmetrics

version 1.2, 20 June 2023

Sergey Poznyakoff.

Table of Contents

1 Overview 1

2 Installation 1

2.1 Bootstrapping 1
2.2 Configuring the Package 2
2.3 Installing 2

3 Usage 3

3.1 Pairing Actions with Metrics 4

4 Functions 5

5 Database Errors 7

6 Configuration 7

7 Downloads and Other Links 9

8 Bug Reports 9

Index 10

1 Overview

This package provides OpenMetrics support for `mailfromd`, a powerful mail filtering utility designed for use with mail transport agents supporting the *milter* protocol (see *Mailfromd Manual*).

OpenMetrics is a plaintext protocol for representing telemetry data, which was originally introduced by `prometheus` systems monitoring and alerting toolkit. It is formally described in `https://openmetrics.io/`.

The `mfmod_openmetrics` module provides MFL functions for declaring and manipulating metric sets from `mailfromd` filter source file and installs an HTTP server for answering the metric queries.

2 Installation

To compile `mfmod_openmetrics`, you will need `mailfromd` version 8.15 or later and the following packages:

1. GNU dbm version 1.19 or later. `http://www.gnu.org.ua/software/gdbm`.
2. GNU Libmicrohttpd. `https://www.gnu.org/software/libmicrohttpd/`

For Debian users, that means installing the following additional packages: `libgdbm-dev`, `libmicrohttpd-dev`.

The installation procedure consists of the traditional three steps: *configure*, *make* and *install*.

2.1 Bootstrapping

If you have cloned the package from git, you will need to *bootstrap* it first. Otherwise, if you are installing the package from the tarball, skip this section and proceed directly to Section 2.2 [configure], page 2.

In order to *bootstrap* the package, you will need the following additional packages:

1. GNU autoconf `https://www.gnu.org/software/autoconf/`.
2. GNU automake `https://www.gnu.org/software/automake/`.
3. GNU libtool `https://www.gnu.org/software/libtool/`.

To bootstrap the package, run

```
autoreconf -f -i -s
```

If it cannot find the `AC_MFMOD` macro, use the `-I` option to point it to the directory where `mfmod.m4` file is installed, e.g.:

```
autoreconf -f -i -s -I /usr/local/share/aclocal/
```

2.2 Configuring the Package

To configure the package, run `configure` from its top-level source directory. If you prefer to build the package in a separate directory (e.g. `build` located within the package top-level directory), change to that directory and run `configure` via its relative path, e.g.:

```
mkdir build
cd build
../configure
```

For a detailed discussion of generic `configure` options, refer to See Section "Running configure Scripts" in *Autoconf*. The rest of this section concerns `configure` options specific for configuring `mfmod_openmetrics`.

By default, `configure` prepares the module parts to be installed in the library and module directories used by mailfromd. To find these directories, `configure` will invoke `mailfromd`. This means that it should be able to find `mailfromd` using the `PATH` environment variable. If it isn't the case, supply the modified `PATH` as argument to `configure`, like that:

```
./configure PATH=/usr/local/sbin:$PATH
```

If the need be, you can select the installation directories manually, using the following two command line options:

`--with-mfmoddir=`*dir*
: Install loadable library (`mfmod_openmetrics.so`) in *dir*.

`--with-mfldir=`*dir*
: Install interface module (`openmetrics.mfl`) in *dir*.

It is advisable to configure the module so as to use the system configuration directory used by the `mailfromd` server. For example, if your `mailfromd` looks for configuration in `/etc/mailfromd`, do:

```
./configure --sysconfdir=/etc/mailfromd
```

This will set the location where the module will look for its configuration file (see Chapter 6 [Configuration], page 7).

If you are unsure what system configuration directory your `mailfromd` installation uses, run

```
mailfromd --no-config --show-defaults
```

and examine the '`script file search path:`' line. The domain part of its first element is the system configuration directory.

2.3 Installing

Once successfully configured, run `make`. This should build the shared library and prepare the package for installation.

Finally, run `make install` as root, to install the package.

3 Usage

Add the following line at the beginning of your filter script:

```
require 'openmetrics'
```

This will cause module to be loaded and initialized. During initialization, the module will parse the configuration file, if it exists, create the metric database or open an existing one, and set up HTTP server thread for answering metric queries. If any of these actions fails, the module signals a `e_failure` exception.

Settings from the configuration file `mfmod_openmetrics.conf`, if it exists in the system configuration directory, override default module settings. See Chapter 6 [Configuration], page 7, for a detailed discussion.

The *metric database file* is a GNU dbm file that keeps current metric values. It can be *temporary*, i.e. created each time `mailfromd` starts up and destroyed when it terminates, or *persistent*, so that the data persist across program restarts. See [database configuration], page 8, for details.

By default HTTP server thread listens on IP address 127.0.0.1 (localhost), port 8080. These settings can also be changed in the configuration.

After loading the module, declare metric family names you are going to use in your script. A metric family is defined using the following function:

```
openmetrics_declare(string name, number type, string help)
```

The *name* parameter gives the metric family name.

The *type* parameter declares the type of the metric. It can be one of the following: `openmetrics_counter`, `openmetrics_gauge`, or `openmetrics_duration`. The first two are floating-point counters. The main difference between them is that `openmetrics_counter` can only increase, while `openmetrics_gauge` can both increase and decrease. The `openmetrics_duration` type is a special case of `openmetrics_gauge` that represents the number of seconds elapsed since the counter was created or reset.

The *help* parameter supplies the help text.

The best place for declaring your metric families is in the `startup` handler, e.g.:

```
prog startup
do
  openmetrics_declare("accept", openmetrics_counter, "Mails accepted")
  openmetrics_declare("reject", openmetrics_counter, "Mails rejected")
  openmetrics_declare("tempfail", openmetrics_counter, "Mails tempfailed")
done
```

If the supplied metric family already exists, its type and help text are checked against `openmetrics_declare` arguments. If types does not match, the `e_inval` exception is signaled. Otherwise, if types match but help texts don't, the help text is updated. Actual values of the metrics in the family are not changed.

Once the metric family is declared, you can manipulate the metrics in it using the following functions:

```
func openmetrics_add(string name, string labelset, number delta)
func openmetrics_incr(string name; string labelset)
func openmetrics_reset(string name; string labelset)
func openmetrics_set(string name, string labelset, number value)
```

Each of them identifies the metric to operate upon by its first two arguments: *name*, which gives the metric family name, and *labelset*, which supplies the *label set* to qualify the metric name (the latter parameter is optional if these are the only arguments the function takes). For the purposes of `mfmod_openmetrics`, a *label set* is a comma-delimited list of *labels*, which take form of '*`key=value`*' pairs. If label set is empty, metric name is same as the metric family name. Otherwise, metric name is '*`name{labelset}`*'. E.g. the following call increases the value of the (counter) metric '`action_total`':

```
openmetrics_incr("action_total")
```

Now consider the following call:

```
openmetrics_incr("action_total", 'action="reject"')
```

It is passed a non-empty label set as its second argument and therefore increases the value of the metric '`action_total{action="reject"}`'.

These functions are described in detail in chapter Chapter 4 [Functions], page 5.

Finally, add calls to the appropriate functions in the right places of your filter script. For example, to keep track of the number of `accept`, `reject`, and `tempfail` actions run, use `openmetrics_incr` before these actions, e.g.:

```
openmetrics_incr("reject")
reject 550 5.1.0 "Sender validity not confirmed"
```

Now, restart `mailfromd` and check if the metrics are returned by the HTTP endpoint:

```
curl -v http://127.0.0.1:8080/metric
```

3.1 Pairing Actions with Metrics

Placing a call to `openmetrics_incr` before each action, as proposed above, is cumbersome and error-prone. To make sure metrics are increased upon executing each action, use the `action` mailfromd hook (see Section "action hook" in *Mailfromd Manual*). For example:

```
prog action
do
  openmetrics_incr(milter_action_name($1))
done
```

This handler will be called before executing each action. Make sure to declare metric families for each `mailfromd` actions, like that:

```
prog startup
do
  openmetrics_declare("accept", openmetrics_counter, "Mails accepted")
  openmetrics_declare("continue", openmetrics_counter,
    "Flow continued to other handlers")
  openmetrics_declare("discard", openmetrics_counter, "Mails discarded")
  openmetrics_declare("reject", openmetrics_counter, "Mails rejected")
  openmetrics_declare("tempfail", openmetrics_counter, "Mails tempfailed")
  set openmetrics_safe 1
done
```

The `action` handler appeared in `mailfromd` version 8.16.90. If you run an earlier version, there is still a way to ensure that corresponding metrics are increased upon executing each action. This will require some preprocessor magic, though.

Define the following `m4` macro:

```
m4_define(‘ACTION’,
 ‘openmetrics_incr("$1")
$1‘’m4_ifelse(‘$2’,‘’,‘’,‘(m4_shift($@))’)’)
```

Place this definition near the top of your `mailfromd.mfl` file and replace literal calls to milter actions with calls to this macro. For example, instead of

```
openmetrics_incr("accept")
accept
```

write just

```
ACTION(accept)
```

Similarly, instead of

```
openmetrics_incr("reject")
reject 550 5.1.0 "Sender validity not confirmed"
```

use

```
ACTION(reject, 550, 5.1.0, "Sender validity not confirmed")
```

This way calling an action will always increase the corresponding metric.

4 Functions

`openmetrics_declare` (*string* `name`, *number* `type`, *string* `help`, ...) [function]

Declares the metric *name* with the given *type* and *help* text.

Allowed values for *type* are:

`openmetrics_counter` [type]

Counter value. Metrics of this type can only increase over time.

`openmetrics_gauge` [type]

Counter value that can both increase and decrease.

`openmetrics_duration` [type]

Special case of `openmetrics_gauge` that represents the number of seconds elapsed since the counter was created or reset. This is useful for reporting uptime and similar values.

If the supplied metric already exists, its type and help text are checked against `openmetrics_declare` arguments. If types does not match, the `e_inval` exception is signaled (see Section “Built-in Exceptions” in *Mailfromd Manual*). Otherwise, if types match but help texts don’t, the help text is updated. Actual values of the metric are not changed.

Any number of optional arguments can be given. They specify the LabelSets to initialize when creating the metric. To illustrate their effect, consider the following two statements:

```
openmetrics_declare("age", openmetrics_duration,
                    "Seconds since the database was created")
```

```
openmetrics_declare("uptime", openmetrics_duration,
                    "Mailfromd instance uptime", "")
```

When metric database is created, the '`age`' metric is created without any actual instances of the metric. Until you create one with `openmetrics_set`, `openmetrics_reset` or similar function, you won't see this metric in responses from the '`/metric`' endpoint.

In the contrast, the declaration of '`uptime`' creates the formal description of the metric, and defines a single instance with an empty label set. You will see the '`uptime`' variable in the responses right after creating the database.

The `openmetrics_declare` function can raise the following exceptions:

e_inval Bad number of arguments, or argument types, or bad metric type given.

e_failure Failed to create temporary database file.

e_dbfailure
GDBM error.

`openmetrics_incr` (*string* *`name`*; *string* *`labelset`*) [function]
Increment the value of the metric *name* by one. Optional *labelset* argument supplies a comma-separated list of '*`name=value`*' parameters (*labels*) that qualify the counter name. If it is given, the actual metric name is '*`name{labelset}`*'.

This function can raise the following exceptions:

`e_inval` Bad number or types of arguments.

`e_failure`
Failed to create temporary database file.

`e_dbfailure`
GDBM error. Most often this occurs because the database is locked by another process. See `database-retry` and `database-wait` in Chapter 6 [Configuration], page 7.

`e_not_found`
Metric not declared. This means the appropriate `openmetrics_declare` call is missing from `prog startup`.

`openmetrics_add` (*string* *`name`*, *string* *`labelset`*, *number* *`delta`*) [function]
Updates the value of the metric '*`name{labelset}`*' (counter or gauge) by adding *delta* to it.

This function can raise the same exceptions as `openmetrics_incr`.

`openmetrics_reset` (*string* *`name`*; *string* *`labelset`*) [function]
Resets the metric *name* (or '*`name{labelset}`*', if *labelset* is given) to its original value, i.e. 0, for `openmetrics_counter` and `openmetrics_gauge` types, and the current system local time (seconds since Epoch), for `openmetrics_duration`.

This function can raise the same exceptions as `openmetrics_incr`.

`openmetrics_set` (*string* `name`, *string* `labelset`, *number* `value`; *number* `ifnew`) [function]

Sets the '*`name`*`{`*`labelset`*`}`' metric to the given value.

If *ifnew* is not null, the value will be set only unless it already exits.

This function can raise the same exceptions as `openmetrics_incr`.

`string openmetrics_http_address()` [function]

Returns network *address* the module is listening on for HTTP requests. The return value is formatted as '*`ip`*`:`*`port`*'.

5 Database Errors

The module stores the collected metrics in a GNU DBM file. Normally only one process can have write access to the database. While it is updating the database, another processes wishing to obtain access (whether read-write or read-only) will wait until it has finished. This waiting consists in several *attempts*. During each attempt the process sleeps for certain amount of time, and then tries to lock and open the database. Two *configuration variables* help tune this process: `database-retry` sets the number of attempts to take, and `database-wait` defines the time to sleep within each attempt. These are discussed in detail in Chapter 6 [Configuration], page 7.

Even in most meticulously configured systems, a possibility of a timeout still remains. Such timeouts can happen in the following functions: `openmetrics_add`, `openmetrics_incr`, `openmetrics_set`, and `openmetrics_reset`. Beside this, another errors that can occur when accessing database files affect these functions. By default, when a database-related error occurs these functions raise the `e_dbfailure` exception. Unless properly caught and handled, this exception will cause the filter to return temporary failure and terminate prematurely. Obviously such behavior is not well-suited for production environments. To avoid it, `mfmod_openmetrics` provides the following variable:

`number openmetrics_safe` [Variable]

Control gracious handling of database-related expressions.

If set to 1, instead of raising exceptions in case of error, the above-mentioned functions will report the error using `echo` and return. The execution of the filter will continue normally.

By default this variable is initialized to 0. It is recommended to set it to 1 in the `startup` handler.

6 Configuration

The module looks for file `mfmod_openmetrics.conf` in the system configuration directory (see [sysconfdir], page 2). If the environment variable `MFMOD_OPENMETRICS_CONF` is defined, its value is used as the full pathname if the configuration file, instead of the default location.

The file format is described in Section "conf-syntax" in *GNU Mailutils Manual*.

The following statements are defined:

General Configuration

`namespace` *`string`* [Configuration]

Sets *namespace prefix*, a string which will be added (with an underscore) at the start of each metric name in the output.

Database Configuration

`persistent` *`bool`* [Configuration]

If '`true`', the metrics database will persist across restarts of `mailfromd`. By default the database is temporary and is deleted when `mailfromd` terminates. If you set the persistent mode, the default database name will be `/tmp/mfmetrics.db`. You can change it using the `database-name` configuration statement.

`database-name` *`string`* [Configuration]

Name of the metric database file. Normally this is important only when used together with `persistent on`. Unless *string* is absolute pathname, the file will be created in `/tmp`. When selecting directory for the database file, be sure it is writable for the user `mailfromd` runs as.

`database-retry` *`n`* [Configuration]

If the database file is in use by another process, retry opening it *n* times. Default value is 10.

`database-wait` *`s`* [Configuration]

Sleep *s* seconds (floating-point number) between two successive attempts to open the database file. The default wait time is 0.1 second.

HTTP Server Configuration

`listen "`*`ip:port`*`"` [Configuration]

Configure the *ip* and *port* to listen for HTTP requests. It can have the following forms:

`listen "`*`ip`*`:`*`port`*`";`
: Listen on the given *ip* and *port*.

`listen "`*`ip`*`";`
: Listen on the given *ip*, port 8080.

`listen ":`*`port`*`";`
: Listen on 127.0.0.1, port *port*.

`listen any;`
: Listen on the first available port on 127.0.0.1. This is for testing the module. The actual address the module is listening on for HTTP is returned by the function `openmetrics_http_address`.

`listen none;`
: Disable HTTP server. This is reserved for the case when HTTP requests are handled by some third party.

The default corresponds to '`listen "127.0.0.1:8080"`'.

`access-log` *bool* [Configuration]

Enable HTTP access logging. E.g.:

```
access-log yes;
```

The logs are produced on the `mailfromd` error stream. Each log line describes a single access and contains the following fields:

- IP address and port of the requesting machine.
- Client IP address. If the request contains the `X-Forwarded-For` header, it is used to determine that value (see the `trusted-ip` statement, below). If not, this field has the same value as the IP in the first field.
- Two dashes.
- Time when the request was replied to. It is formatted as follows (in the `strftime` notation):

  ```
  [%d/%b/%Y:%H:%M:%S %z]
  ```

- Request method.
- Request URL with parameters.
- Response status code.
- Response size in bytes.
- Value of the `Referer` header or '`-`' if no such header.
- Value of the `User-Agent` header or '`-`' if no such header.

`trusted-ip` *cidr-list* [Configuration]

List of trusted proxy IP addresses. This is used to determine source IP address for logging (a functionality, similar to `mod_remoteip` module in `Apache`). Any number of arguments are allowed. Each argument is parsed as a CIDR. E.g.:

```
trusted-ip 127.0.0.1 10.0.1.0/16;
```

Default is '`127.0.0.0/8`'.

7 Downloads and Other Links

The program can be downloaded from `https://download.gnu.org.ua/release/mfmod_openmetrics`.

The source repository is available at `https://git.gnu.org.ua/mfmod_openmetrics.git/`.

The package development page is at `https://puszcza.gnu.org.ua/projects/mfmod_openmetrics`.

Mailfromd home page is: `https://www.gnu.org.ua/software/mailfromd`.

8 Bug Reports

If you think you found a bug in `mfmod_openmetrics` or in its documentation, please send a mail to `gray@gnu.org` (Sergey Poznyakoff) or use the bug tracker at `https://puszcza.gnu.org.ua/bugs/?group=mfmod_openmetrics` (requires authorization).

Index

/

/tmp/mfmetrics.db 8

A

`access-log` 9

C

configuration file 3

D

`database-name` 8
`database-retry` 8
`database-wait` 8

L

`listen` 8

M

mailfromd 1
metric database file 3
`mfmod_openmetrics.conf` 3, 7

N

`namespace` 8

O

OpenMetrics 1
`openmetrics_add` 6
`openmetrics_declare` 5
`openmetrics_http_address()` 7
`openmetrics_incr` 6
`openmetrics_reset` 6
`openmetrics_set` 7

P

`persistent` 8
Prometheus 1

T

`trusted-ip` 9