PrometheusInterface

v. 0.5.0

Holger Queckenstedt

30.05.2024

Contents

Chapter 1

Introduction

The interface library **PrometheusInterface** provides **Robot Framework** keywords to communicate with the monitoring system **Prometheus**. With the help of this interface it is possible to use **Prometheus** to monitor data provided by **Robot Framework** tests.

Currently this library is under development. We work on it. But what is available up to now, is not ready for being used in productive systems (but hopefully will be in near future).

Informations about how to install the **PrometheusInterface** can be found in the README.

(later)

T.B.C.

Chapter 2

Description

2.1 Test setup

How many tests are passed? How many tests are failed? On which test benches do these tests run? Did any resets occur during test execution? How about the temporary unavailability of required test external components?

To monitor all these informations, a test system setup is necessary consisting of at least the following components:

1. A test framework that executes the test and provides the data to be monitored (here: the **Robot Framework**). This includes the possibility that more than one test framework runs in parallel.
2. A component that collects and stores data from all executed test frameworks (here: the monitoring system **Prometheus**).
3. A component that visualizes all data that have been collected (here: **Grafana**).

This is not the only way to establish such a test system, but in this document we concentrate on **Robot Framework**, **Prometheus** and **Grafana**.

To be able to collect values, **Prometheus** requires an http based counterpart. And the **Robot Framework** must be enabled to support this counterpart. In pure Python this is realized by a **Prometheus Python client library**. On **Robot Framework** level this is done by this component **PrometheusInterface** that is a mapping between the interface of the **Prometheus Python client library** and **Robot Framework** keywords.

Or in other words: **PrometheusInterface** uses the **Prometheus Python client library** to provide values to **Prometheus** to enable the data visualization in **Grafana**.

The **Prometheus Python client library** is part of the installation dependencies of **PrometheusInterface**. **Prometheus** and **Grafana** are components that have to be downloaded, installed and configured separately.

2.2 Installations

1. Prometheus

To install **Prometheus** please visit the homepage. This homepage also contains a *getting started* section containing useful hints about how to configure.the application.

Further informations can also be found here.

2. Grafana

The advantage of using **Grafana** for data visualization is to have a *ready to use* interface to **Prometheus** available. Other possible solutions are not in focus here.

To install **Grafana** please visit the homepage.

3. Prometheus Python client library

This library belongs to the installation dependencies of **PrometheusInterface**. A separate installation is not required. In case you want to learn more about this client library please visit the following web pages: [1], [2], [3].

2.3 Configuration

t.b.c.

Chapter 3

prometheus_interface.py

3.1 Class: prometheus_interface

Imported by:

```
from PrometheusInterface.prometheus_interface import prometheus_interface
```

The class 'prometheus_interface' provides to communicate with the monitoring system Prometheus. For this purpose the 'Prometheus Python client library' is used.

3.1.1 Keyword: get_version

Returns the version of this interface library

3.1.2 Keyword: who_am_i

Returns the full name of this interface library

3.1.3 Keyword: where_am_i

Returns path and file name of this interface library

3.1.4 Keyword: get_port_number

Returns the port number assigned to this instance of the library

3.1.5 Keyword: add_info

add_info (experimental only)

3.1.6 Keyword: add_lighting

add_lighting (experimental only)

3.1.7 Keyword: set_daylight

set_daylight (experimental only)

3.1.8 Keyword: set_nightlight

set_nightlight (experimental only)

3.1.9 Keyword: add_counter

This keyword adds a new counter. The values of existing counters can be changed with `inc_counter`.

Arguments:

- `name`
  The name of the new counter
  / *Condition*: required / *Type*: str /
- `description`
  The description of the new counter
  / *Condition*: required / *Type*: str /
- `labels`
  A semicolon separated list of label names assigned to the new counter
  / *Condition*: optional / *Type*: str / *Default*: None /

Returns:

- `success`
  / *Type*: bool /
  Indicates if the computation of the keyword was successful or not
- `result`
  / *Type*: str /
  The result of the computation of the keyword

3.1.10 Keyword: inc_counter

This keyword increments a counter. The counter has to be added with '`add_counter`' before.

Arguments:

- `name`
  The name of the counter
  / *Condition*: required / *Type*: str /
- `value`
  The value of increment. If not given, the value of the counter is incremented by value 1.
  / *Condition*: optional / *Type*: int / *Default*: None /
- `labels`
  A semicolon separated list of labels assigned to the counter. The order of labels must fit to the order of label names like defined in `add_counter`.
  / *Condition*: optional / *Type*: str / *Default*: None /

Returns:

- `success`
  / *Type*: bool /
  Indicates if the computation of the keyword was successful or not
- `result`
  / *Type*: str /
  The result of the computation of the keyword

3.1.11 Keyword: add_gauge

This keyword adds a new gauge. The values of existing gauges can be changed with `set_gauge`, `inc_gauge` and `dec_gauge`.

Arguments:

- `name`
  The name of the new gauge
  / *Condition*: required / *Type*: str /
- `description`
  The description of the new gauge
  / *Condition*: required / *Type*: str /
- `labels`
  A semicolon separated list of label names assigned to the new gauge
  / *Condition*: optional / *Type*: str / *Default*: None /

Returns:

- `success`
  / *Type*: bool /
  Indicates if the computation of the keyword was successful or not
- `result`
  / *Type*: str /
  The result of the computation of the keyword

3.1.12 Keyword: set_gauge

This keyword sets the value for a gauge. The gauge has to be added with '`add_gauge`' before.

Arguments:

- `name`
  The name of the gauge
  / *Condition*: required / *Type*: str /
- `value`
  The new value of the gauge.
  / *Condition*: optional / *Type*: int / *Default*: None /
- `labels`
  A semicolon separated list of labels assigned to the gauge. The order of labels must fit to the order of label names like defined in `add_gauge`.
  / *Condition*: optional / *Type*: str / *Default*: None /

Returns:

- `success`
  / *Type*: bool /
  Indicates if the computation of the keyword was successful or not
- `result`
  / *Type*: str /
  The result of the computation of the keyword

3.1.13 Keyword: inc_gauge

This keyword increments a gauge. The gauge has to be added with 'add_gauge' before.

Arguments:

- name

  The name of the gauge

  / *Condition*: required / *Type*: str /

- value

  The value of increment. If not given, the value of the gauge is incremented by value 1.

  / *Condition*: optional / *Type*: int / *Default*: None /

- labels

  A semicolon separated list of labels assigned to the gauge. The order of labels must fit to the order of label names like defined in add_gauge.

  / *Condition*: optional / *Type*: str / *Default*: None /

Returns:

- success

  / *Type*: bool /

  Indicates if the computation of the keyword was successful or not

- result

  / *Type*: str /

  The result of the computation of the keyword

3.1.14 Keyword: dec_gauge

This keyword decrements a gauge. The gauge has to be added with 'add_gauge' before.

Arguments:

- name

  The name of the gauge

  / *Condition*: required / *Type*: str /

- value

  The value of decrement. If not given, the value of the gauge is decremented by value 1.

  / *Condition*: optional / *Type*: int / *Default*: None /

- labels

  A semicolon separated list of labels assigned to the gauge. The order of labels must fit to the order of label names like defined in add_gauge.

  / *Condition*: optional / *Type*: str / *Default*: None /

Returns:

- success

  / *Type*: bool /

  Indicates if the computation of the keyword was successful or not

- result

  / *Type*: str /

  The result of the computation of the keyword

Chapter 4

Appendix

About this package:

Table 4.1: Package setup

Setup parameter	Value
Name	PrometheusInterface
Version	0.5.0
Date	30.05.2024
Description	Additional Robot Framework keywords
Package URL	robotframework-prometheus
Author	Holger Queckenstedt
Email	Holger.Queckenstedt@de.bosch.com
Language	Programming Language :: Python :: 3
License	License :: OSI Approved :: Apache Software License
OS	Operating System :: OS Independent
Python required	>=3.0
Development status	Development Status :: 4 - Beta
Intended audience	Intended Audience :: Developers
Topic	Topic :: Software Development

Chapter 5

History

0.1.0	05/2024
Initial version	
0.2.0	05/2024
Added metric type 'Gauge'; code maintenance	
0.3.0	05/2024
Added keywords 'inc_gauge' and 'dec_gauge'	
0.4.0	05/2024
Added interface description	
0.5.0	05/2024
Adapted handling of library version and date	

PrometheusInterface.pdf
Created at 30.05.2024 - 17:07:06
by GenPackageDoc v. 0.41.1