ACC: Prometheus integration

Summary

There are tons of collectors/applications that support Prometheus line protocol. It includes such things as Docker(experimental https://docs.docker.com/config/thirdparty/prometheus/), Kubernetes, GItlab, Ceph, Collectd, etcd, InfluxDB, InfluxDB Telegraf and more. As exposition format is quite simple and there is an HTTP agent in Zabbix (4.0+) it could be relatively straightforward to implement its support in Zabbix.

Zabbix Acceptance

Functional:

1. Add new preprocessing step **Prometheus pattern:** preprocessing to find specific single metric from exposition format using PromQL-like filters
    1. Supported in item preprocessing
    2. With two parameters: **pattern** and **output**
        1. **pattern parameter**
            1. Possible options

Filter metrics where	Pattern parameter
metric name equals to string	`<metric name>` or `{__name__="<metric name>"}`
metric name matches regex	`{__name__=~"<regex>"}`
<label name> value equals to some string	`{<label name>="<label value>", ...}`
<label name> value matches to some string	`{<label name>=~"<regex>", ...}`
value equals to some string	`{__name__=~".*"} == <value>`
any combination of above	`<metric name>{<label1 name>="<label1 value>", <label2 name>=~"<regex>", ...} == <value>`

            2. It must be possible to find the metrics if there are other labels present in the metric

        3. Order of labels in the line should not matter
        4. Use input placeholder attribute for pattern parameter:

           ```
           <metric name>{<label name>="<label value>", ...} ==
           <value>
           ```

    2. **Output parameter**
        1. You can choose what should be returned: metric's value (**\value**) or **specific label**'s value
        2. If parameter is left empty then \value is used by default
        3. Use placeholder for output parameter:

           ```
           \value|<label name>
           ```

3. Should return error if there are multiple lines after filtering. Error message must contain up to 10 first lines returned.
4. Example:

   to get metric of cpu-total:

   ```
   # HELP cpu_usage_system Telegraf collected metric
   # TYPE cpu_usage_system gauge
   cpu_usage_system{cpu="cpu-total",host="host1"} 1.1940298507220641
   cpu_usage_system{cpu="cpu0",host="host1"} 1.1940298507220641
   cpu_usage_system{cpu="cpu1",host="host1"} 1.1340298507220641
   ```

   use this pattern:

   ```
   cpu_usage_system{cpu="cpu-total"}
   ```

   or this:

   ```
   cpu_usage_system{cpu=~"cpu-tot.+"}
   ```

2. Add new preprocessing step **Prometheus to JSON** preprocessing to find a list of specific metrics from exposition format using PromQL-like filters and convert them to JSON array
    1. Supported in LLD preprocessing
    2. Returns JSON containing attributes
        1. Metric name
        2. Metric value
        3. Labels (if present)
        4. Type (if present)
        5. Help (if present)
        6. Raw line
    3. With parameter: pattern. See pattern parameter requirements in **Prometheus pattern**
    4. Example:

1. to discover disks **C:**, **D:** from these lines:

```
# HELP wmi_logical_disk_free_bytes Free space in bytes
(LogicalDisk.PercentFreeSpace)
# TYPE wmi_logical_disk_free_bytes gauge
wmi_logical_disk_free_bytes{volume="C:"} 3.5180249088e+11
wmi_logical_disk_free_bytes{volume="D:"} 2.627731456e+09
wmi_logical_disk_free_bytes{volume="HarddiskVolume4"}
4.59276288e+08
```

use this pattern:

```
wmi_logical_disk_free_bytes{volume=~"[A-Z]:"}
```

or just this pattern for all disks:

```
wmi_logical_disk_free_bytes
```

and you will get output:

```
 [
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes
(LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "C:"
         },
        "value": "3.5180249088e+11",
        "line_raw":
"wmi_logical_disk_free_bytes{volume=\"C:\"}
3.5180249088e+11"
    },
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes
(LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "D:"
         },
        "value": "2.627731456e+09",
        "line_raw":
"wmi_logical_disk_free_bytes{volume=\"D:\"}
2.627731456e+09"
    },
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes
(LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "HarddiskVolume4"
        },
        "value": "4.59276288e+08",
        "line_raw":
"wmi_logical_disk_free_bytes{volume=\"HarddiskVolume4\"}
4.59276288e+08"
    }
]
```

2. to discover windows services from these lines:

```
wmi_service_state{name="devicesflowusersvc_7b100",state="runn
ing"} 1
wmi_service_state{name="devicesflowusersvc_7b100",state="star
t pending"} 0
wmi_service_state{name="devicesflowusersvc_7b100",state="stop
pending"} 0
wmi_service_state{name="devicesflowusersvc_7b100",state="stop
ped"} 0
wmi_service_state{name="devicesflowusersvc_7b100",state="unkn
own"} 0
wmi_service_state{name="dhcp",state="continue pending"} 0
wmi_service_state{name="dhcp",state="pause pending"} 0
wmi_service_state{name="dhcp",state="paused"} 0
wmi_service_state{name="dhcp",state="running"} 1
wmi_service_state{name="dhcp",state="start pending"} 0
wmi_service_state{name="dhcp",state="stop pending"} 0
wmi_service_state{name="dhcp",state="stopped"}
wmi_service_state{name="dhcp",state="unknown"} 0
```

use this pattern:

```
wmi_service_state == 1
```

3. All examples for **Prometheus pattern and Prometheus to JSON** should work as described below in "Examples" section.
4. The following must be documented
   1. Differences between PromQL and our syntax used in expressions:

	PromQL instant vector selector	**Zabbix Prometheus preprocessing**
Differences		
Query target	Prometheus server	plain text in Prometheus exposition format
Returns	Instant vector	single value or label value **Prometheus pattern** array of metrics with single value in JSON for **Prometheus to JSON**
Label matching operators	=, !=, =~, !~	=, =~

Regex used in label or metric name matching	RE2	PCRE
Comparison operators	See supported here	Only **== (equal)** is supported for value filtering
Similarities		
Selecting by metric name equals to string	<metric name> or `{__name__="<metric name>"}`	<metric name> or `{__name__="<metric name>"}`
Selecting by metric name matches regex	{__name__=~"<regex>"}	{__name__=~"<regex>"}
Selecting by <label name> value equals to some string	`{<label name>="<label value>", ...}`	`{<label name>="<label value>", ...}`
Selecting by <label name> value matches to some string	`{<label name>=~"<regex>", ...}`	`{<label name>=~"<regex>", ...}`
Selecting by value equals to some string	`{__name__=~".*"} == <value>`	`{__name__=~".*"} == <value>`

2. How to do LLD discovery with preprocessing
3. How to retrieve label value instead of value
4. How to use filtering by value

Examples and format overview

Format overview

https://prometheus.io/docs/instrumenting/exposition_formats/

```
metric_name [
  "{" label_name "=" `"` label_value `"` { "," label_name "=" `"` label_value
`"` } [ "," ] "}"
] value [ timestamp ]
```

see also https://prometheus.io/docs/concepts/data_model/ for restrictions applied on metric_name or label_name

about PromQL queries: https://prometheus.io/docs/prometheus/latest/querying/basics/#instant-vector-selectors

some thoughts on value filtering: https://github.com/prometheus/prometheus/issues/1216

Sample output

```
# HELP http_requests_total The total number of HTTP requests.
# TYPE http_requests_total counter
http_requests_total{method="post",code="200"} 1027 1395066363000
http_requests_total{method="post",code="400"}    3 1395066363000

# Escaping in label values:
msdos_file_access_time_seconds{path="C:\\DIR\\FILE.TXT",error="Cannot find
file:\n\"FILE.TXT\""} 1.458255915e9

# Minimalistic line:
metric_without_timestamp_and_labels 12.47

# A weird metric from before the epoch:
something_weird{problem="division by zero"} +Inf -3982045

# A histogram, which has a pretty complex representation in the text format:
# HELP http_request_duration_seconds A histogram of the request duration.
# TYPE http_request_duration_seconds histogram
http_request_duration_seconds_bucket{le="0.05"} 24054
http_request_duration_seconds_bucket{le="0.1"} 33444
http_request_duration_seconds_bucket{le="0.2"} 100392
http_request_duration_seconds_bucket{le="0.5"} 129389
http_request_duration_seconds_bucket{le="1"} 133988
http_request_duration_seconds_bucket{le="+Inf"} 144320
http_request_duration_seconds_sum 53423
http_request_duration_seconds_count 144320

# Finally a summary, which has a complex representation, too:
# HELP rpc_duration_seconds A summary of the RPC duration in seconds.
# TYPE rpc_duration_seconds summary
rpc_duration_seconds{quantile="0.01"} 3102
rpc_duration_seconds{quantile="0.05"} 3272
```

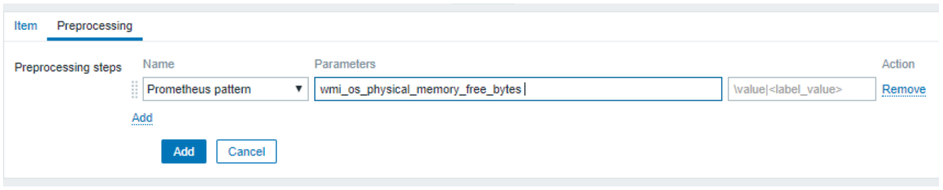
```
rpc_duration_seconds{quantile="0.5"} 4773
rpc_duration_seconds{quantile="0.9"} 9001
rpc_duration_seconds{quantile="0.99"} 76656
rpc_duration_seconds_sum 1.7560473e+07
rpc_duration_seconds_count 2693
```

More examples

Prometheus pattern

Example 1:

```
wmi_os_physical_memory_free_bytes 8.492331008e+09
```

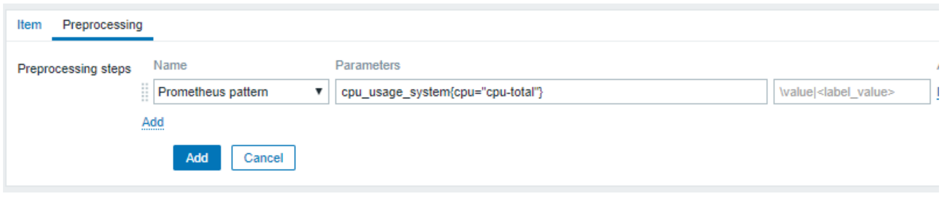


Preprocessing steps
Name
Parameters
Action
Prometheus pattern
wmi_os_physical_memory_free_bytes
\value|<label_value>

Example 2:

Get CPU total from Telegraf agent.

```
# HELP cpu_usage_system Telegraf collected metric
# TYPE cpu_usage_system gauge
cpu_usage_system{cpu="cpu-total"} 1.1940298507220641
cpu_usage_system{cpu="cpu0"} 1.1940298507220641
cpu_usage_system{cpu="cpu1"} 1.1340298507220641
```

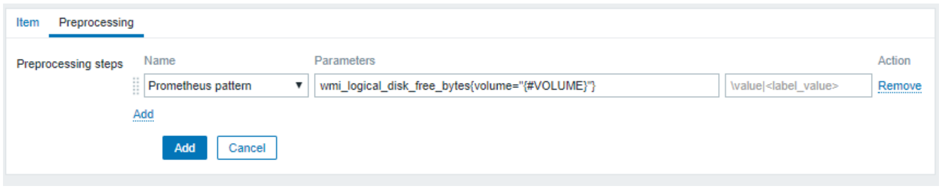


Preprocessing steps
Name
Parameters
Prometheus pattern
cpu_usage_system{cpu="cpu-total"}
\value|<label_value>

Example 3:

In item prototype (see more about LLD below)

```
# HELP wmi_logical_disk_free_bytes Free space in bytes
(LogicalDisk.PercentFreeSpace)
# TYPE wmi_logical_disk_free_bytes gauge
wmi_logical_disk_free_bytes{volume="C:"} 3.5180249088e+11
wmi_logical_disk_free_bytes{volume="D:"} 2.627731456e+09
wmi_logical_disk_free_bytes{volume="HarddiskVolume4"} 4.59276288e+08
```



Item | Preprocessing

Preprocessing steps

Name	Parameters		Action
Prometheus pattern	wmi_logical_disk_free_bytes{volume="{#VOLUME}"}	\value|<label_value>	Remove

Some one line examples. Metrics given from Prometheus:

```
# HELP cpu_usage_system Telegraf collected metric
# TYPE cpu_usage_system gauge
cpu_usage_system{cpu="cpu-total",host="host1"} 1.1940298507220641
cpu_usage_system{cpu="cpu0",host="host1"} 1.1940298507220641
cpu_usage_system{cpu="cpu1",host="host1"} 1.1340298507220641
```

All the following should match metric related to cpu="cpu-total" (even though some of them doesn't make sense):

```
cpu_usage_system{cpu="cpu-total",host=~".*"}
cpu_usage_system{cpu="cpu-total",host=~"*",\value=~".*"}
cpu_usage_system{cpu="cpu-total",host=~"*"}
cpu_usage_system{cpu="cpu-total"}
cpu_usage_system{cpu=~"cpu-tot.+"}
{__name__=~"cpu_usage_syst.+",cpu=~"cpu-tot.+"}
```

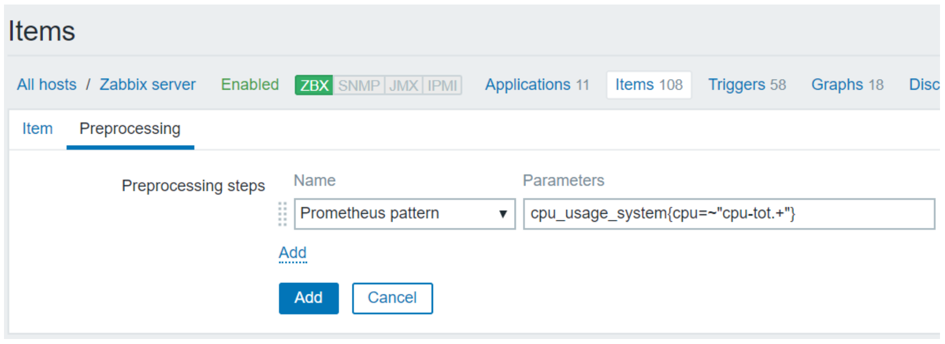
And these filters would also match cpu0, cpu1:

```
cpu_usage_system{cpu=~".*"}
```

Example 4:

Get CPU total from Telegraf agent with host label added on telegraf side.

```
# HELP cpu_usage_system Telegraf collected metric
# TYPE cpu_usage_system gauge
cpu_usage_system{cpu="cpu-total",host="host1"} 1.1940298507220641
cpu_usage_system{cpu="cpu0",host="host1"} 1.1940298507220641
cpu_usage_system{cpu="cpu1",host="host1"} 1.1340298507220641
```



Preprocessing steps

Name	Parameters
Prometheus pattern	cpu_usage_system{cpu=~"cpu-tot.+"}

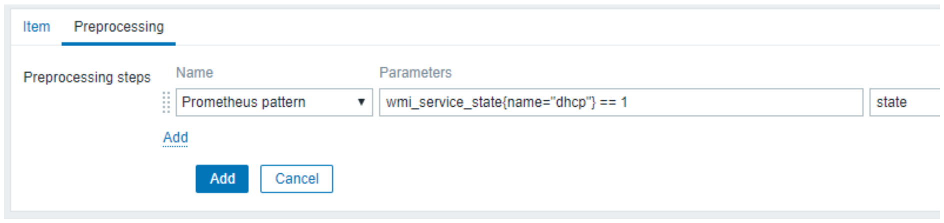
Add

Still matches, even though additional label `host` added.

Example 5:

Match Windows Service DHCP actual state:

```
wmi_service_state{name="devicesflowusersvc_7b100",state="running"} 1
wmi_service_state{name="devicesflowusersvc_7b100",state="start pending"} 0
wmi_service_state{name="devicesflowusersvc_7b100",state="stop pending"} 0
wmi_service_state{name="devicesflowusersvc_7b100",state="stopped"} 0
wmi_service_state{name="devicesflowusersvc_7b100",state="unknown"} 0
wmi_service_state{name="dhcp",state="continue pending"} 0
wmi_service_state{name="dhcp",state="pause pending"} 0
wmi_service_state{name="dhcp",state="paused"} 0
wmi_service_state{name="dhcp",state="running"} 1
wmi_service_state{name="dhcp",state="start pending"} 0
wmi_service_state{name="dhcp",state="stop pending"} 0
wmi_service_state{name="dhcp",state="stopped"} 0
wmi_service_state{name="dhcp",state="unknown"} 0
wmi_service_state{name="diagnosticshub.standardcollector.service",state="cont
inue pending"} 0
wmi_service_state{name="diagnosticshub.standardcollector.service",state="paus
e pending"} 0
wmi_service_state{name="diagnosticshub.standardcollector.service",state="paus
ed"} 0
wmi_service_state{name="diagnosticshub.standardcollector.service",state="runn
ing"} 0
wmi_service_state{name="diagnosticshub.standardcollector.service",state="star
t pending"} 0
wmi_service_state{name="diagnosticshub.standardcollector.service",state="stop
pending"} 0
wmi_service_state{name="diagnosticshub.standardcollector.service",state="stop
ped"} 1
wmi_service_state{name="diagnosticshub.standardcollector.service",state="unkn
own"} 0
.....
```



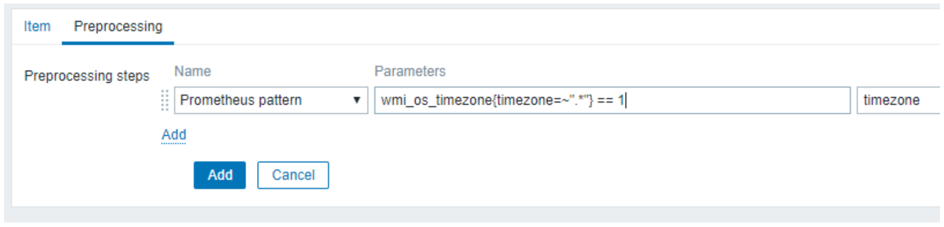
Preprocessing steps

Name	Parameters	
Prometheus pattern	wmi_service_state{name="dhcp"} == 1	state

Example 6:

Extract label value as item value:

```
# HELP wmi_os_timezone OperatingSystem.LocalDateTime
# TYPE wmi_os_timezone gauge
wmi_os_timezone{timezone="MSK"} 1
```

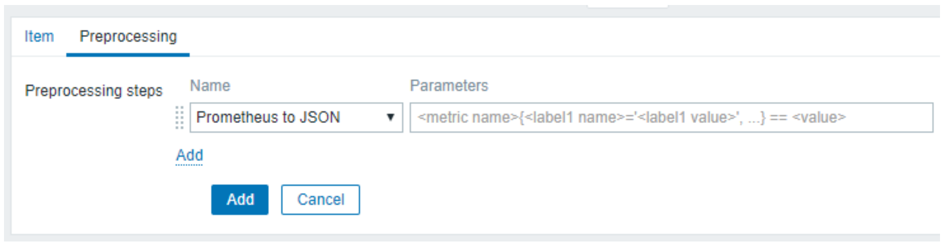


Preprocessing steps

Name	Parameters	
Prometheus pattern	wmi_os_timezone{timezone=~".*"} == 1	timezone

Prometheus to JSON

Prometheus to JSON preprocessing step works the same way as 'Prometheus pattern' but it outputs LLD-ready JSON instead.



Preprocessing steps

Name	Parameters
Prometheus to JSON	<metric name>{<label1 name>='<label1 value>', ...} == <value>

Example 1:

Find similar metrics and output to LLD format.

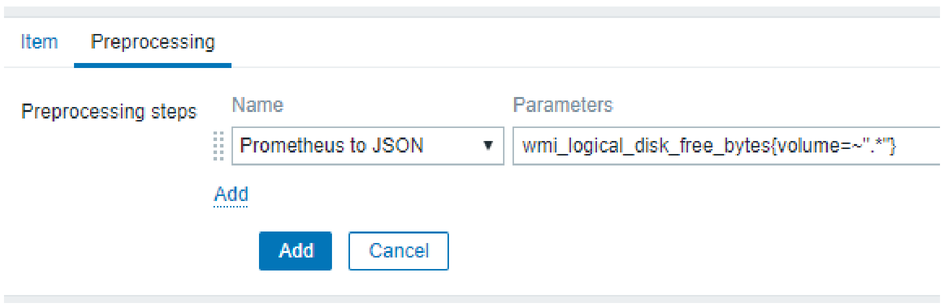
To be used in Discovery rules.

Discovery item type should be: **HTTP agent**

```
# HELP wmi_logical_disk_free_bytes Free space in bytes
(LogicalDisk.PercentFreeSpace)
# TYPE wmi_logical_disk_free_bytes gauge
```

```
wmi_logical_disk_free_bytes{volume="C:"} 3.5180249088e+11
wmi_logical_disk_free_bytes{volume="D:"} 2.627731456e+09
wmi_logical_disk_free_bytes{volume="HarddiskVolume4"} 4.59276288e+08
```

after preprocessing like this applied:



Preprocessing steps

Name	Parameters
Prometheus to JSON	wmi_logical_disk_free_bytes{volume=~".*"}

should become:

```
[
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "C:"
         },
        "value": "3.5180249088e+11",
        "line_raw": "wmi_logical_disk_free_bytes{volume=\"C:\"}
3.5180249088e+11"
    },
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "D:"
         },
        "value": "2.627731456e+09",
        "line_raw": "wmi_logical_disk_free_bytes{volume=\"D:\"}
2.627731456e+09"
    },
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "HarddiskVolume4"
         },
        "value": "4.59276288e+08",
        "line_raw": "wmi_logical_disk_free_bytes{volume=\"HarddiskVolume4\"}
4.59276288e+08"
    }
]
```

You then need to map values to LLD json:

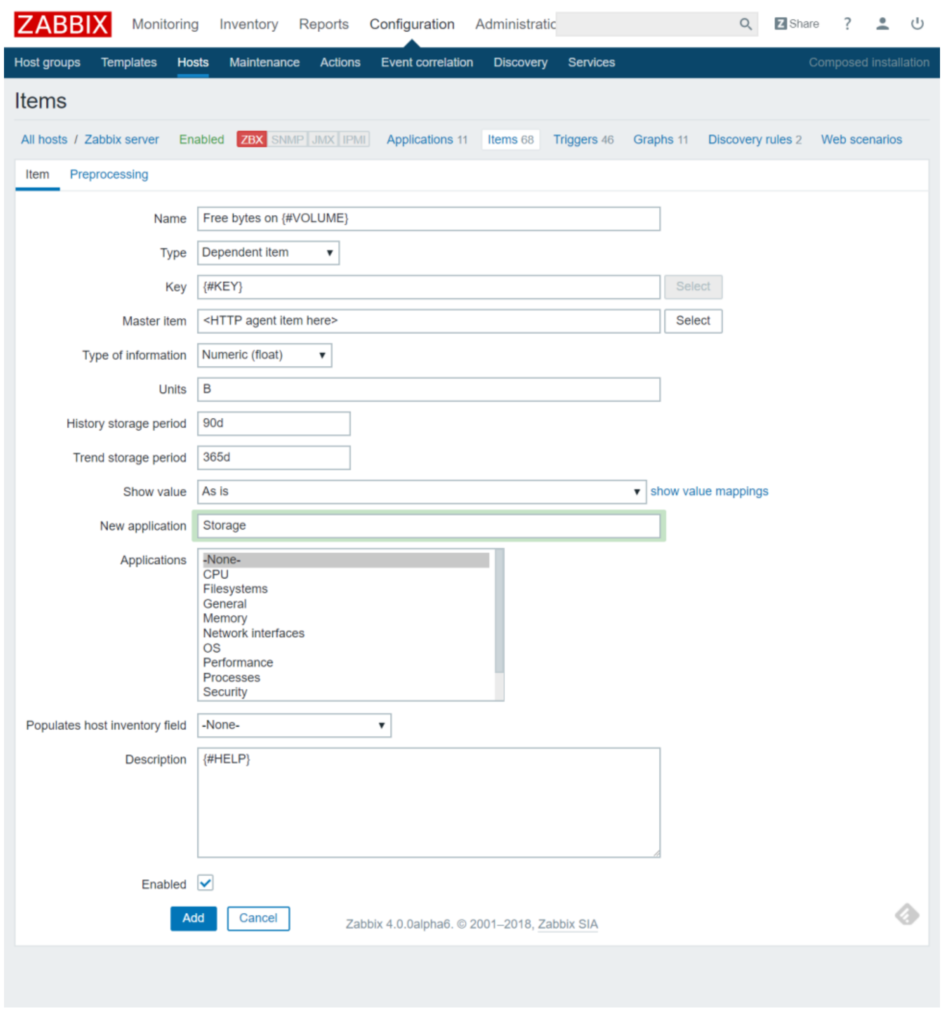
{#VOLUME} = $.labels['volume']

{#KEY} = $['name']

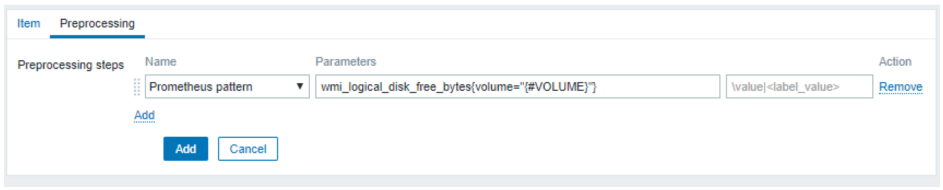
{#help} = $['help']

Once you have LLD ready, you can proceed to create item prototypes like so:

In item prototype:



Name	Free bytes on {#VOLUME}
Type	Dependent item
Key	{#KEY}
Master item	<HTTP agent item here>
Type of information	Numeric (float)
Units	B
History storage period	90d
Trend storage period	365d
Show value	As is
New application	Storage
Applications	-None-, CPU, Filesystems, General, Memory, Network interfaces, OS, Performance, Processes, Security
Populates host inventory field	-None-
Description	{#HELP}
Enabled	✓



Preprocessing steps	Name	Parameters		Action
	Prometheus pattern	wmi_logical_disk_free_bytes{volume="{#VOLUME}"}	\value|<label_value>	Remove

Business Use Cases

Ability to monitor any object that exposes metrics in Prometheus line format including Kubernetes

Out of scope

Support for Prometheus Histogram and summary type of metrics is currently out of scope

Timestamp parsing(probably ignore it first time)

Parsing of NaN , +Inf, -Inf values

Changes log

v0.1 initial draft

v0.2

- Adapted Prometheus pattern and Prometheus discovery to resemble https://prometheus.io/docs/prometheus/latest/querying/basics/#instant-vector-selectors
- Removed Prometheus match

v0.3

- Renamed Prometheus discovery to Prometheus to JSON. It might be possible to use this function in the future in item preprocessing as well
- Added comparison between PromQL and Zabbix preprocessing

v1.0

- No changes