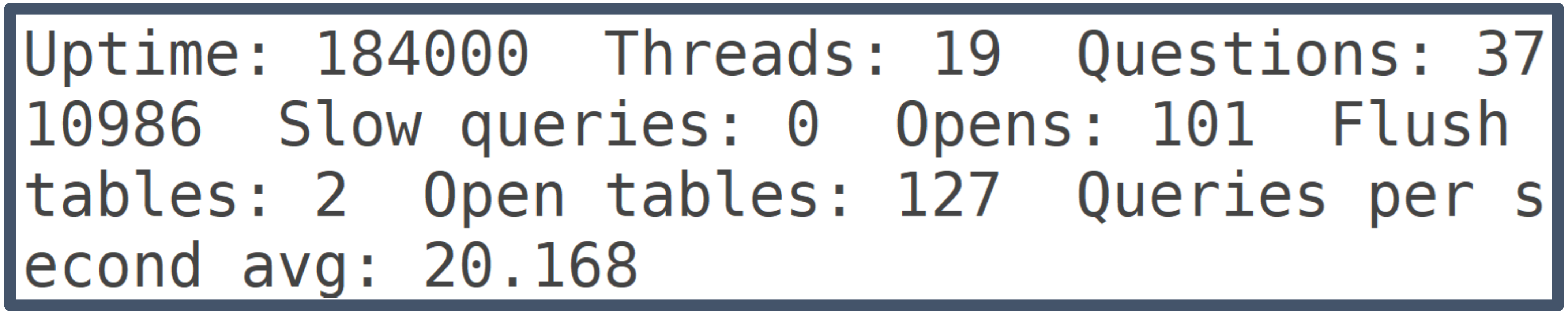
DEEP DIVE IN **ZABBIX PRE-PROCESSING**

Arturs Lontons
ZABBIX Technical Support Engineer

ZABBIX SUMMIT '19

WHY PREPROCESSING?

The retrieved data is freeform and is not fit for calculations, aggregations and/or optimal data storage:



```
Uptime: 184000  Threads: 19  Questions: 37
10986  Slow queries: 0  Opens: 101  Flush
tables: 2  Open tables: 127  Queries per s
econd avg: 20.168
```

WHY PREPROCESSING?

HOW CAN I SOLVE THIS?

USE PREPROCESSING!

● Text preprocessing ● Structured data ● Arithmetic	● Deltas ● Numeral systems ● Javascript	● Validation ● Prometheus Exporter

THE EVOLUTION OF PREPROCESSING

DEEP DIVE IN
ZABBIX PRE-PROCESSING

THE LEGACY WAY

Limited preprocessing support in versions <3.4:

- Custom multiplier
- Numeral system transformations
- Delta calculation (speed per second/simple change)

What if you needed more?

Preprocess the data by using your own scripts!

VERSIONS 3.4 AND 4.2

What has changed?

ZABBIX VERSION 3.4

Introducing - **Preprocessing!**

New ways to transform data:

- Regex
- Trim
- XML XPath
- JSON Path

ZABBIX VERSION 4.2

- Extended preprocessing
- Validation and throttling
- Custom error handling
- Preprocessing support by Zabbix Proxy
- Prometheus exporter support

ZABBIX SUMMIT '19

VERSION 4.4

What has changed?

- Preprocessing of XML data via Xpath
- JSONPath aggregation and search
- Extended Custom error handling
- Introducing CSV to JSON preprocessing
- WMI, JMX and ODBC data collection returns JSON arrays – ready to preprocess via JSONPath!

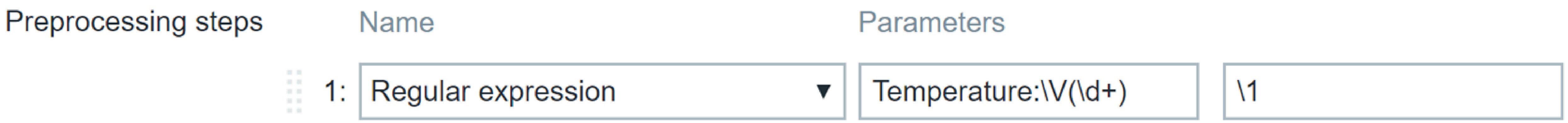
THE MANY WAYS OF PREPROCESSING

DEEP DIVE IN
ZABBIX PRE-PROCESSING

TEXT PREPROCESSING

Retrieve a value by using Regex:

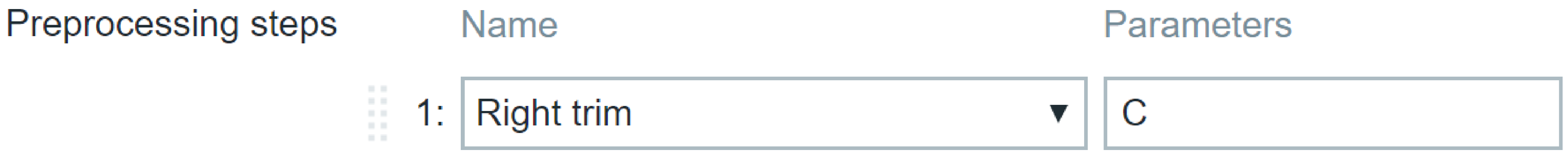
Temperature: 36C $\longrightarrow$ PCRE $\longrightarrow$ 36



Preprocessing steps	Name	Parameters	
1:	Regular expression ▼	Temperature:\V(\d+)	\1

Trim the retrieved value and store it as a number:

36C $\longrightarrow$ Right Trim $\longrightarrow$ 36



Preprocessing steps	Name	Parameters
1:	Right trim ▼	C

STRUCTURED DATA

Retrieve value from JSON or XML data:

```
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book category="cooking">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year> <price>30.00</price>
</book>

<book category="children">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="web">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
```

`sum(/bookstore/book/price)`

→ 99.94

`count(/bookstore/book)`

→ 3

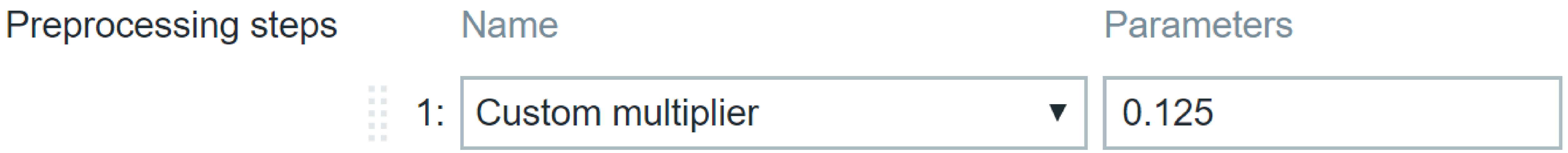
`number(/bookstore/book[price<30]/price)`

→ 29.99

ZABBIX SUMMIT '19

ARITHMETIC

Custom Multipliers to transform numeric data:



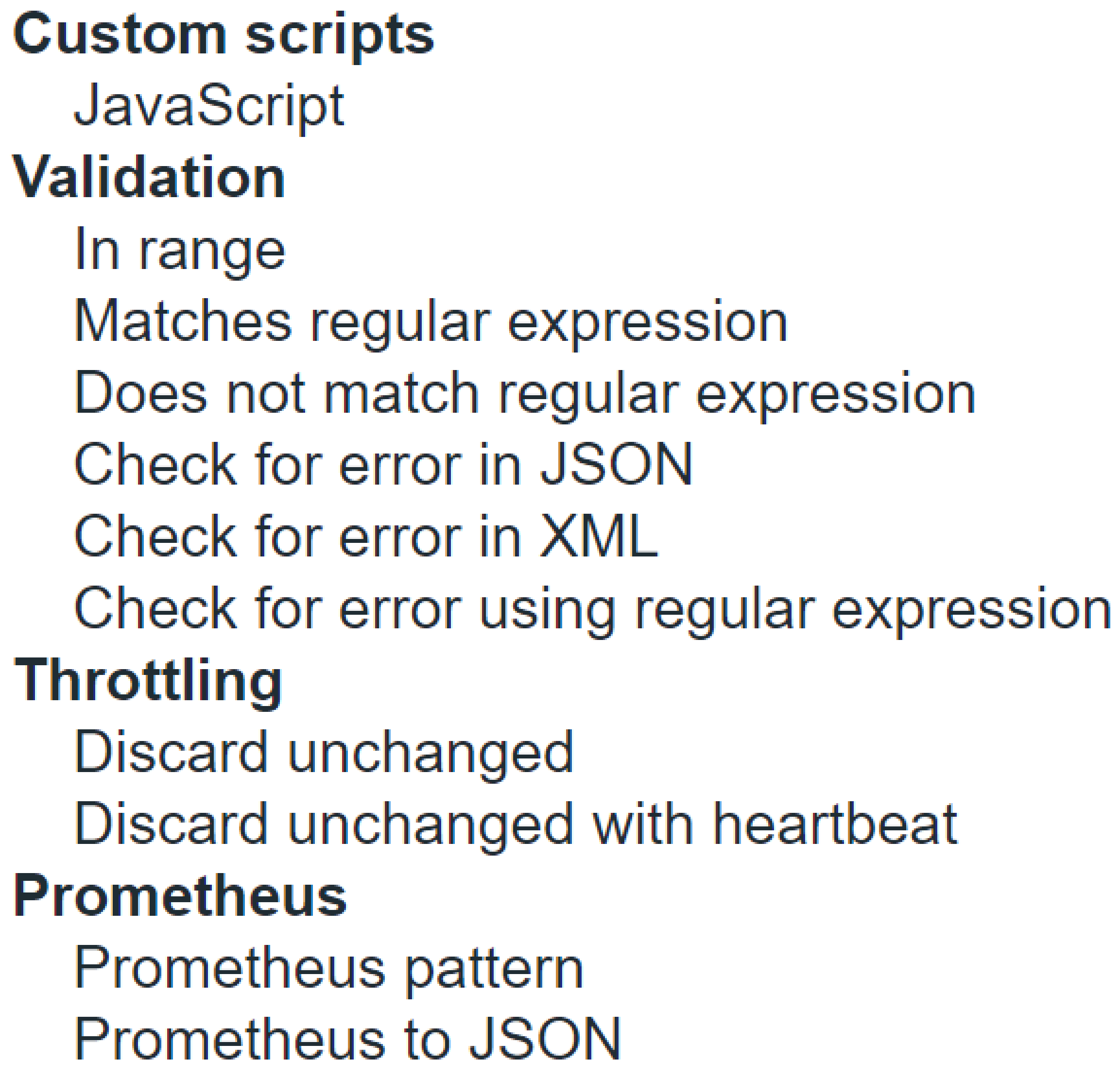
Preprocessing steps		Name	Parameters
	1:	Custom multiplier ▼	0.125

DELTA CALCULATIONS

- Difference between current and previous value
- Change per second

INTRODUCED IN VERSION 4.2

- Custom scripts
- Validation
- Throttling
- Prometheus



Custom scripts
- JavaScript

Validation
- In range
- Matches regular expression
- Does not match regular expression
- Check for error in JSON
- Check for error in XML
- Check for error using regular expression

Throttling
- Discard unchanged
- Discard unchanged with heartbeat

Prometheus
- Prometheus pattern
- Prometheus to JSON

JAVASCRIPT

Implemented with **Duktape** JavaScript engine!

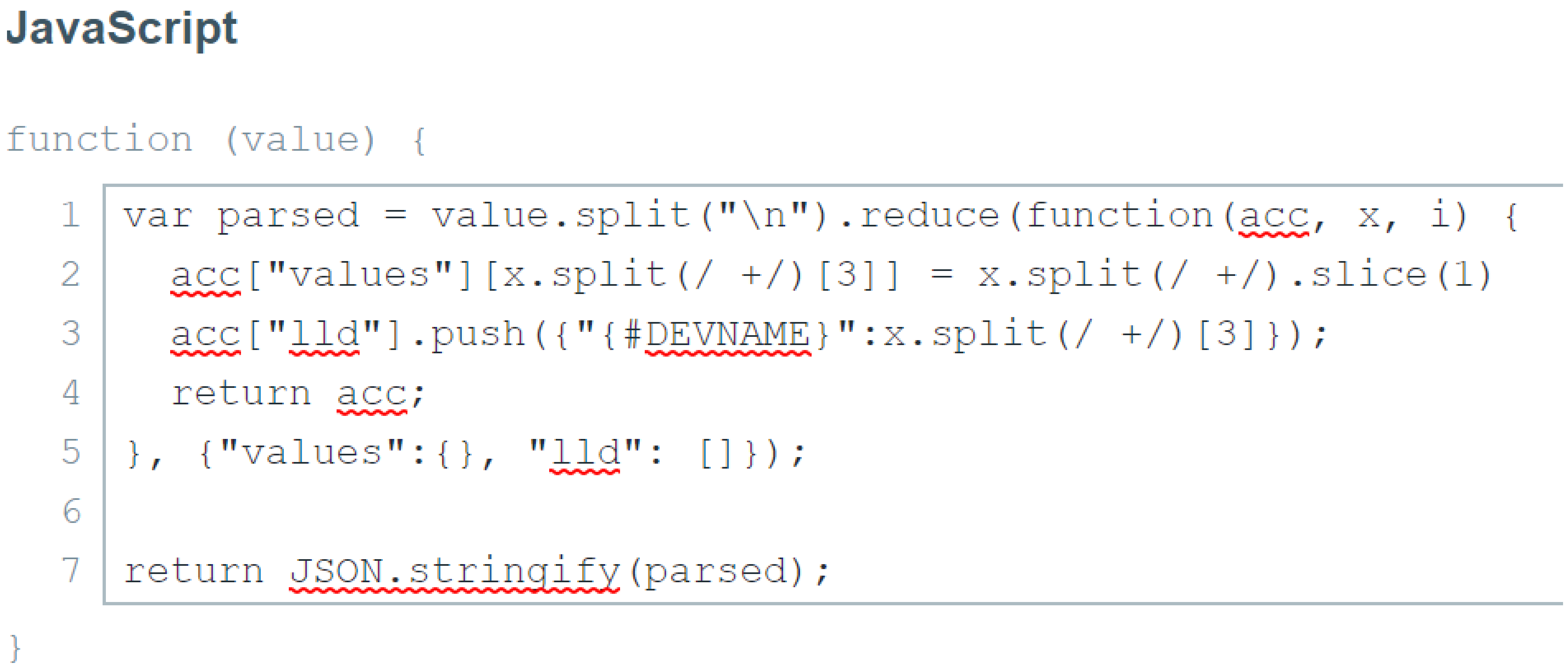
With JavaScript you can perform:

- Data transformation
- Data aggregation
- Data filtering
- Logical expressions
- etc.

...All done internally by Zabbix

JAVASCRIPT

Convert **diskstats** to JSON:



JavaScript

```
function (value) {
var parsed = value.split("\n").reduce(function(acc, x, i) {
  acc["values"][x.split(/ +/)[3]] = x.split(/ +/).slice(1)
  acc["lld"].push({"{#DEVNAME}":x.split(/ +/)[3]});
  return acc;
}, {"values":{}, "lld": []});

return JSON.stringify(parsed);
}
```

JAVASCRIPT

Initial data

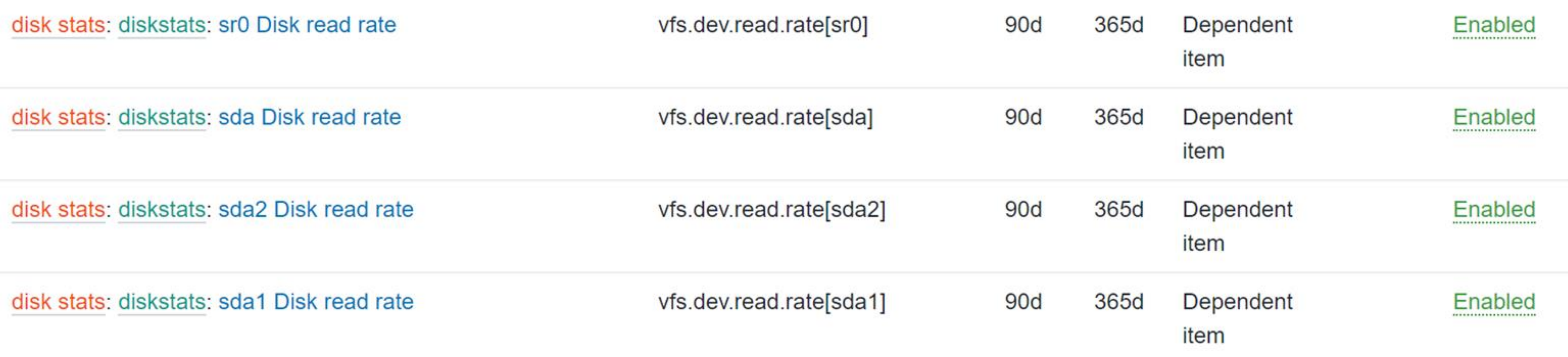
```
  8       0 sda 9224 3 606559 8510 207906 1416 7707716 275248 0 219108 283619
  8       1 sda1 969 0 12390 146 10 0 4136 14 0 153 160
  8       2 sda2 8230 3 590897 8352 157268 1416 7703580 98777 0 48122 106992
 11       0 sr0 0 0 0 0 0 0 0 0 0 0 0
253       0 dm-0 8061 0 581737 8311 203596 0 7703556 292885 0 219283 301196
253       1 dm-1 90 0 4920 32 3 0 24 5 0 21 37
```

Preprocessed data - ready for LLD!

```
{"values":{"sda":
["8","0","sda","9248","3","607351","8535","219127","1432","8108169","286348","0","228233","294732"],"sda1":
["8","1","sda1","969","0","12390","146","10","0","4136","14","0","153","160"],"sda2":
["8","2","sda2","8248","3","591617","8371","165831","1432","8104033","102626","0","50209","110848"],"sr0":
["11","0","sr0","0","0","0","0","0","0","0","0","0","0","0"],"dm-0":["253","0","dm-
0","8079","0","582457","8330","214576","0","8104009","304043","0","228416","312373"],"dm-1":["253","1","dm-
1","90","0","4920","32","3","0","24","5","0","21","37"]},"lld":[{"{#DEVNAME}":"sda"},{"{#DEVNAME}":"sda1"},{"
{#DEVNAME}":"sda2"},{"{#DEVNAME}":"sr0"},{"{#DEVNAME}":"dm-0"},{"{#DEVNAME}":"dm-1"}]}
```

JAVASCRIPT

Result:



disk stats: diskstats: sr0 Disk read rate	vfs.dev.read.rate[sr0]	90d	365d	Dependent item	Enabled
disk stats: diskstats: sda Disk read rate	vfs.dev.read.rate[sda]	90d	365d	Dependent item	Enabled
disk stats: diskstats: sda2 Disk read rate	vfs.dev.read.rate[sda2]	90d	365d	Dependent item	Enabled
disk stats: diskstats: sda1 Disk read rate	vfs.dev.read.rate[sda1]	90d	365d	Dependent item	Enabled

The items have been discovered and created by our LLD!

VALIDATION

Validate data against validation logic:

- In range
- Matches regular expression
- Does not match regular expression
- Check for errors in JSON/XML or by using regex

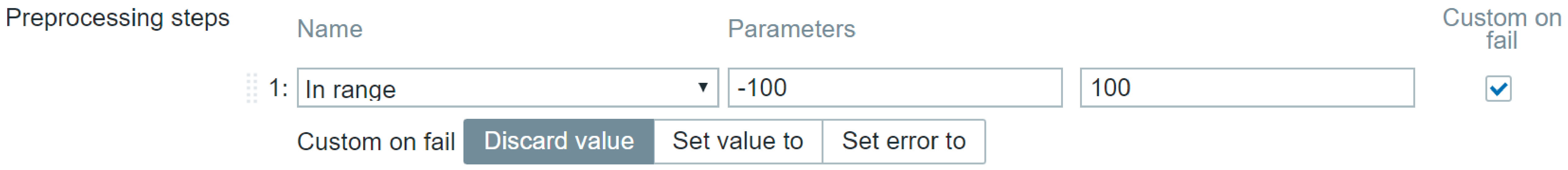
CUSTOM ON FAIL

Define the behavior in cases when the preprocessing fails:

- Discard value
- Set value to
- Set error to

VALIDATION

Discard the value outside of defined range:

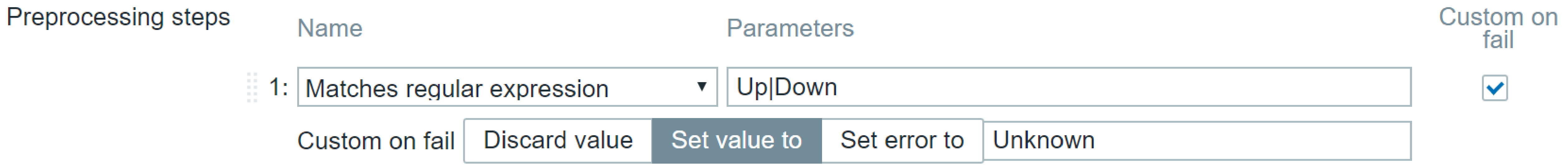


Preprocessing steps

Name	Parameters		Custom on fail
1: In range	-100	100	✓

Custom on fail: [✓ Discard value] Set value to Set error to

Match a regular expression and set the value to Unknown if no match is obtained:



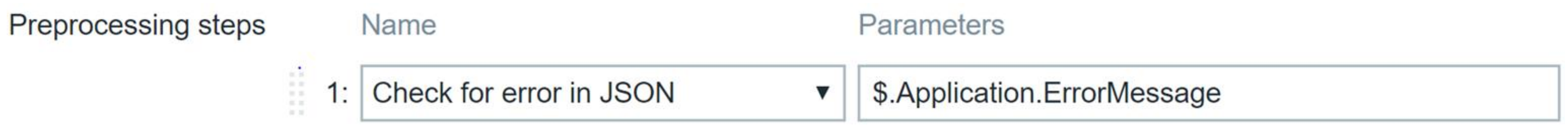
Preprocessing steps

Name	Parameters	Custom on fail
1: Matches regular expression	Up|Down	✓

Custom on fail: Discard value [✓ Set value to] Set error to Unknown

VALIDATION

Check for an application-level error message located at JSONPath/XPath:



Preprocessing steps | Name | Parameters

1: Check for error in JSON ▼ | $.Application.ErrorMessage

```
{
  "Software":"Zabbix",
  "Version":"4.2.0",
  "OS":"CentOS 7",
  "ErrorMessage":"Service Down"
}
```

Error message!

THROTTLING

Enables high frequency monitoring with minimal performance impact:

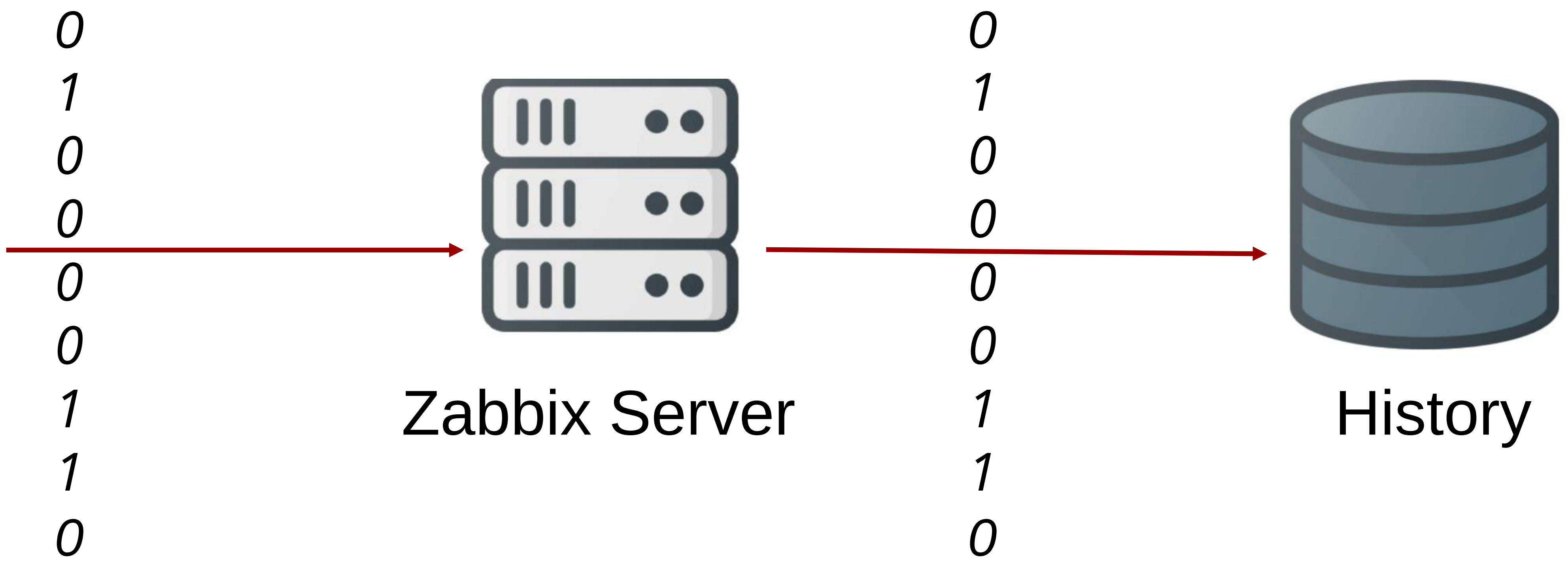
- Discard repeating values
- Discard repeating values with a heartbeat

Useful when we are receiving a lot of duplicate data at a very high frequency!

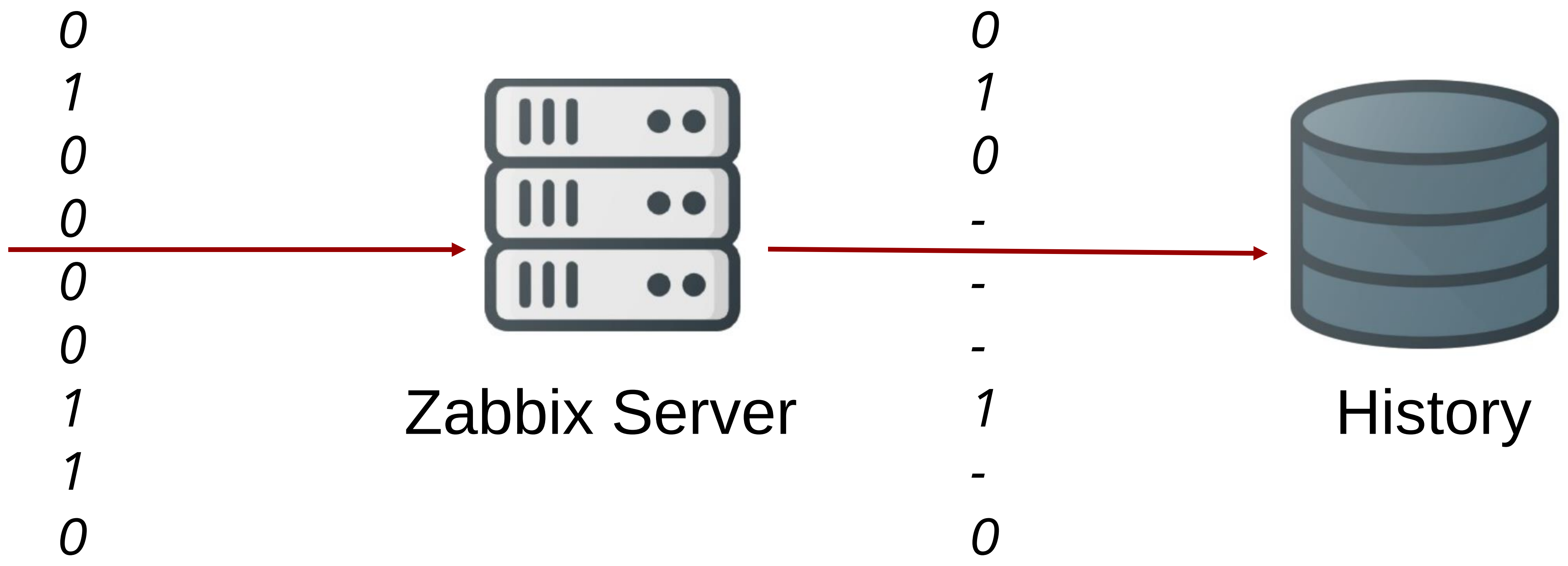
Application service monitoring:							
Time:	00:00	00:02	00:04	00:06	00:08	00:10	00:12
Data:	Up	Down	Down	Up	Up	Up	Up

ZABBIX SUMMIT '19

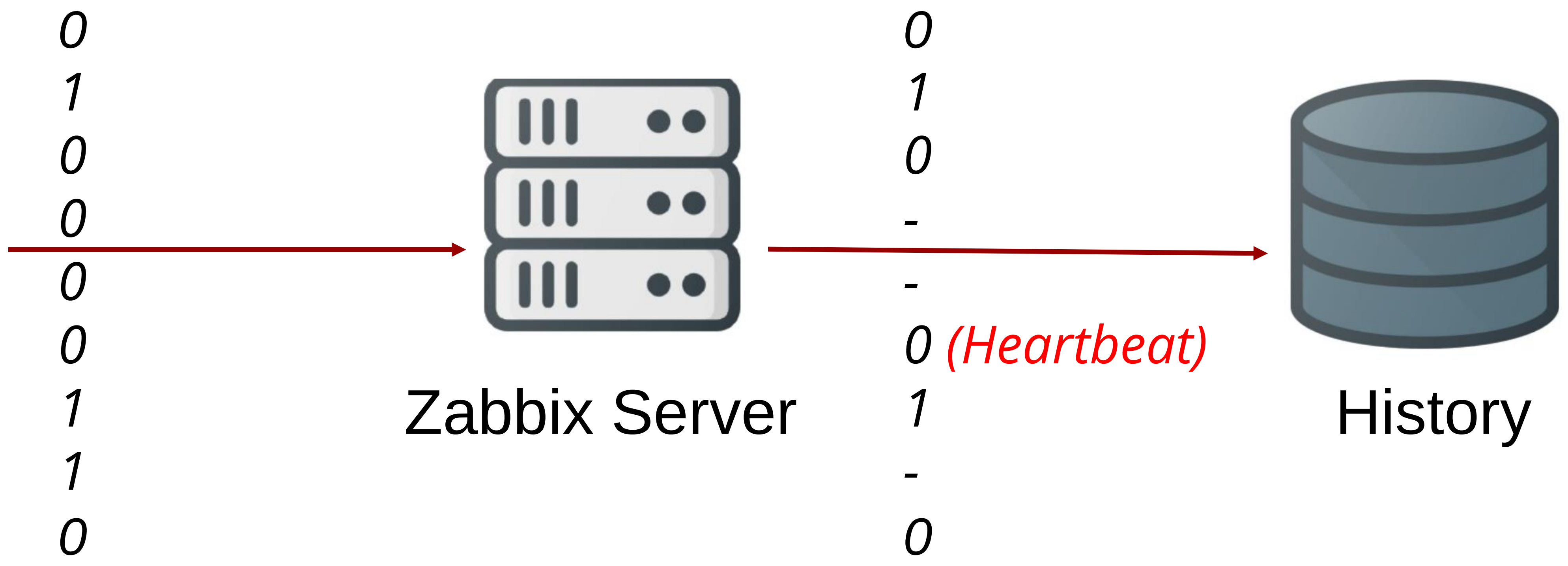
BEFORE THROTTLING



WITH THROTTLING

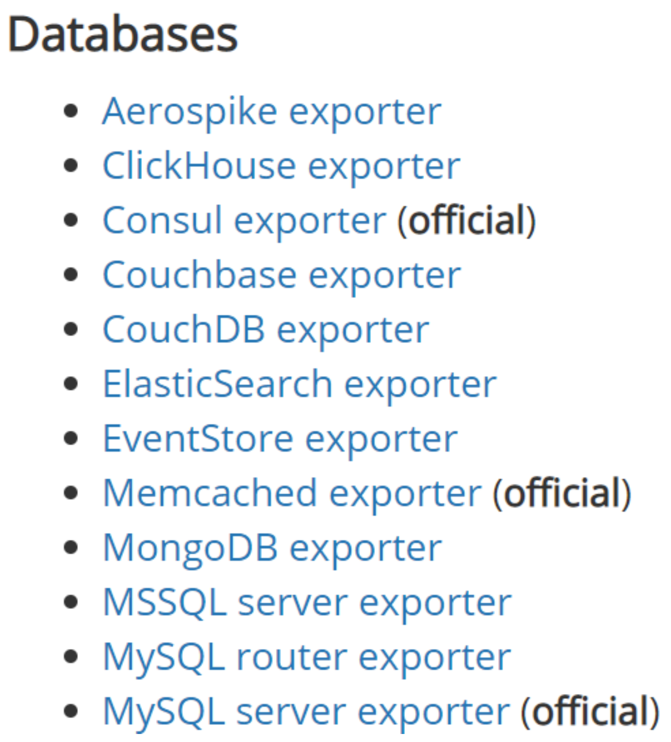


THROTTLING WITH A HEARTBEAT



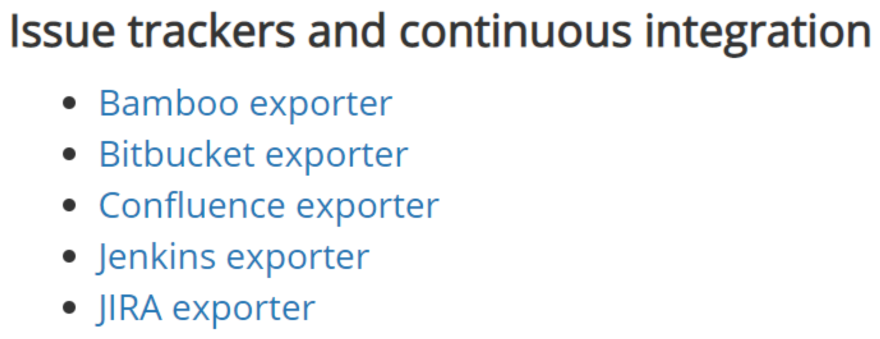
PROMETHEUS

- Query Prometheus endpoint with HTTP checks
- Use preprocessing to obtain metrics
- Use within LLD to discover components monitored by Prometheus



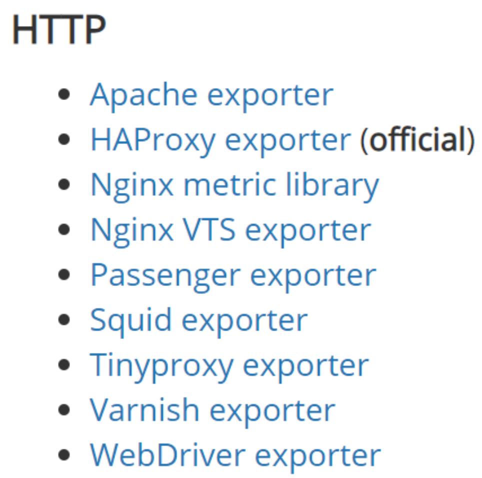
Databases

- Aerospike exporter
- ClickHouse exporter
- Consul exporter (official)
- Couchbase exporter
- CouchDB exporter
- ElasticSearch exporter
- EventStore exporter
- Memcached exporter (official)
- MongoDB exporter
- MSSQL server exporter
- MySQL router exporter
- MySQL server exporter (official)



Issue trackers and continuous integration

- Bamboo exporter
- Bitbucket exporter
- Confluence exporter
- Jenkins exporter
- JIRA exporter



HTTP

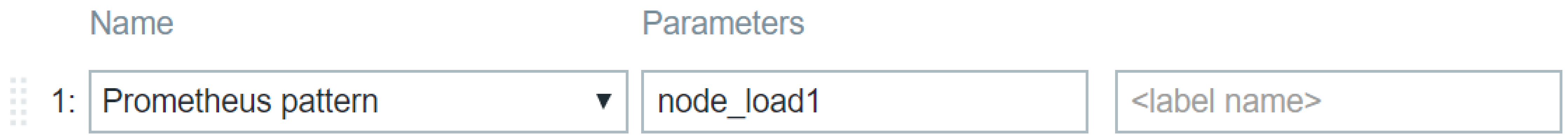
- Apache exporter
- HAProxy exporter (official)
- Nginx metric library
- Nginx VTS exporter
- Passenger exporter
- Squid exporter
- Tinyproxy exporter
- Varnish exporter
- WebDriver exporter

...And many more

PROMETHEUS PATTERN

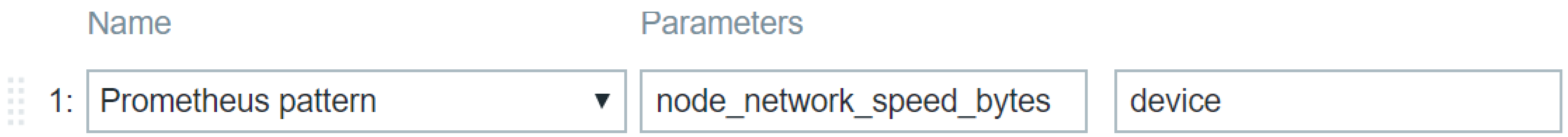
- Create master HTTP item
- Create dependent items with "Prometheus pattern"

Retrieve a metric:



	Name	Parameters	
1:	Prometheus pattern ▼	node_load1	<label name>

Retrieve a label value:

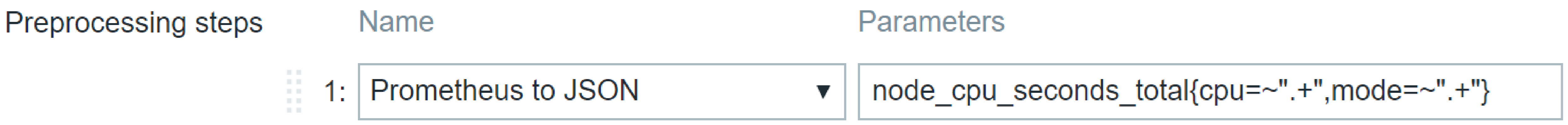


	Name	Parameters	
1:	Prometheus pattern ▼	node_network_speed_bytes	device

PROMETHEUS LLD

Create a dependent item LLD with a “Prometheus to JSON” preprocessing step.

Discover all CPU’s:

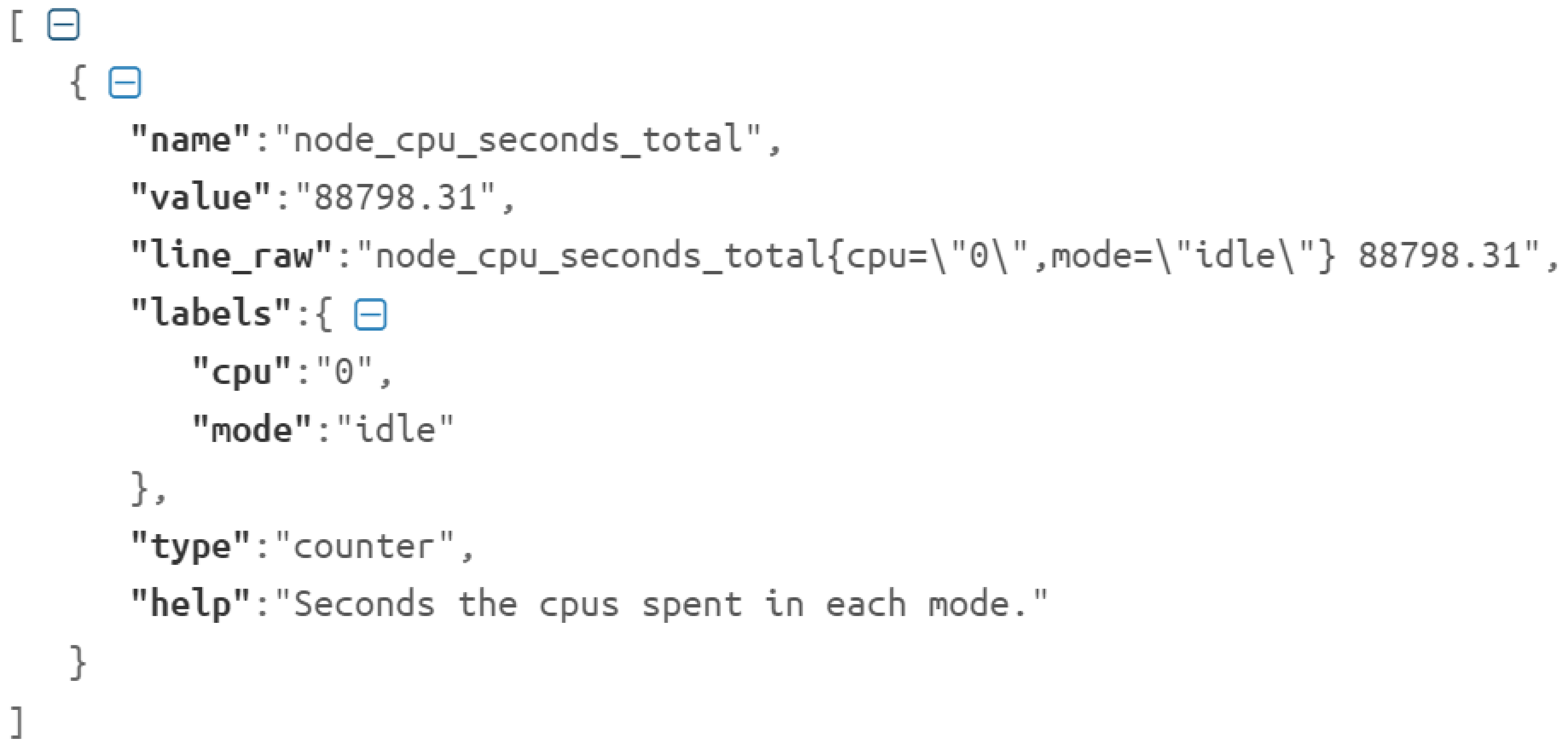


Preprocessing steps		Name	Parameters
	1:	Prometheus to JSON ▼	node_cpu_seconds_total{cpu=~".+",mode=~".+"}

ZABBIX SUMMIT '19

PROMETHEUS LLD

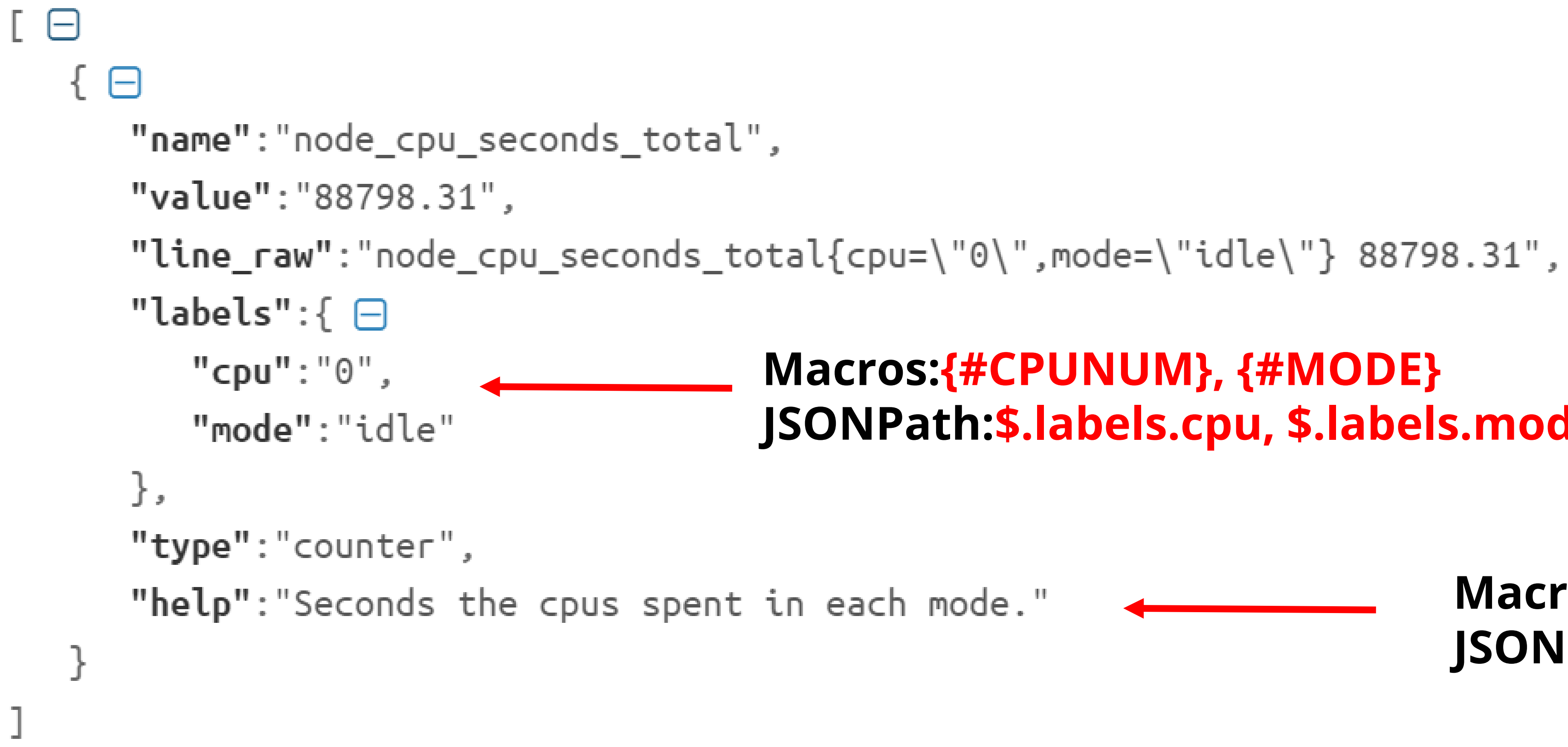
Retrieved JSON:



```
[
  {
    "name":"node_cpu_seconds_total",
    "value":"88798.31",
    "line_raw":"node_cpu_seconds_total{cpu=\"0\",mode=\"idle\"} 88798.31",
    "labels":{
      "cpu":"0",
      "mode":"idle"
    },
    "type":"counter",
    "help":"Seconds the cpus spent in each mode."
  }
]
```

PROMETHEUS LLD

Retrieved JSON:



```
[
  {
    "name":"node_cpu_seconds_total",
    "value":"88798.31",
    "line_raw":"node_cpu_seconds_total{cpu=\"0\",mode=\"idle\"} 88798.31",
    "labels":{
      "cpu":"0",
      "mode":"idle"
    },
    "type":"counter",
    "help":"Seconds the cpus spent in each mode."
  }
]
```

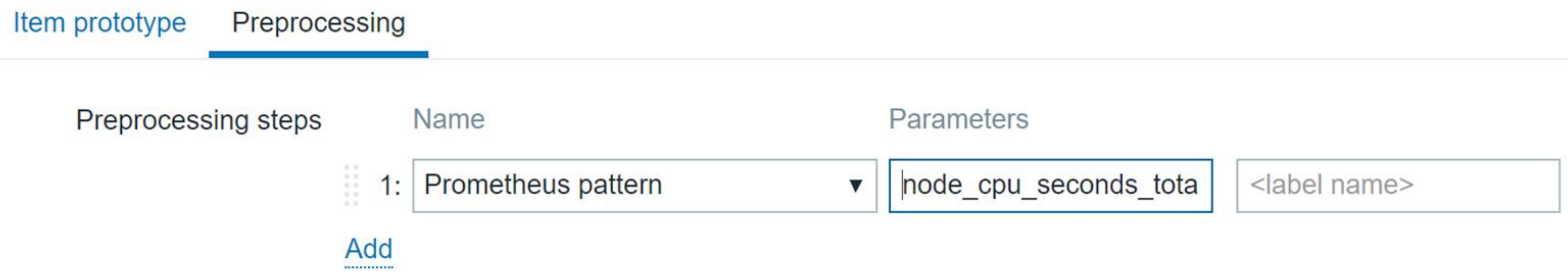
Macros:{#CPUNUM}, {#MODE}
JSONPath:$.labels.cpu, $.labels.mode

Macro:{#HELP}
JSONPath:$.help

PROMETHEUS LLD

Item prototype with “Prometheus pattern” preprocessing step:

node_cpu_seconds_total{cpu="{#CPUNUM}",mode="{#MODE}"}



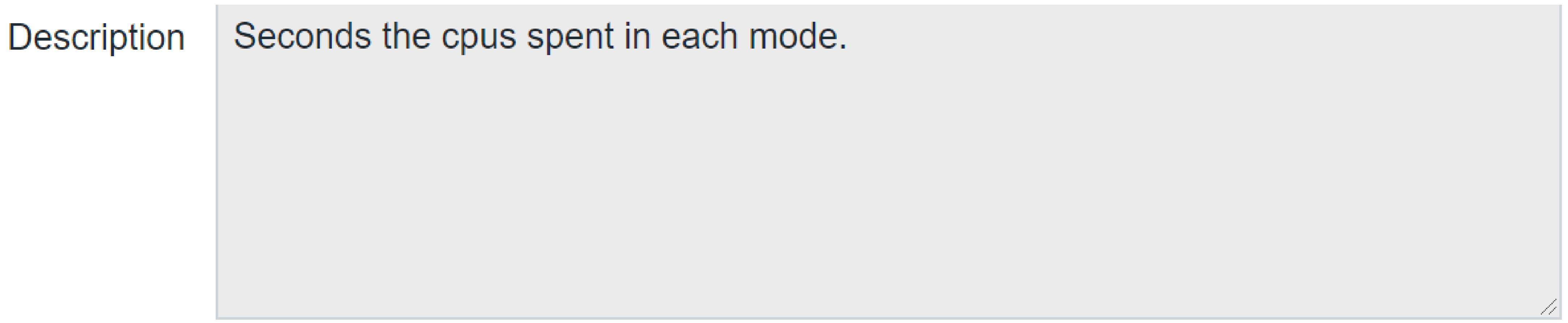
Preprocessing steps	Name	Parameters	
1:	Prometheus pattern	node_cpu_seconds_tota	<label name>

PROMETHEUS LLD

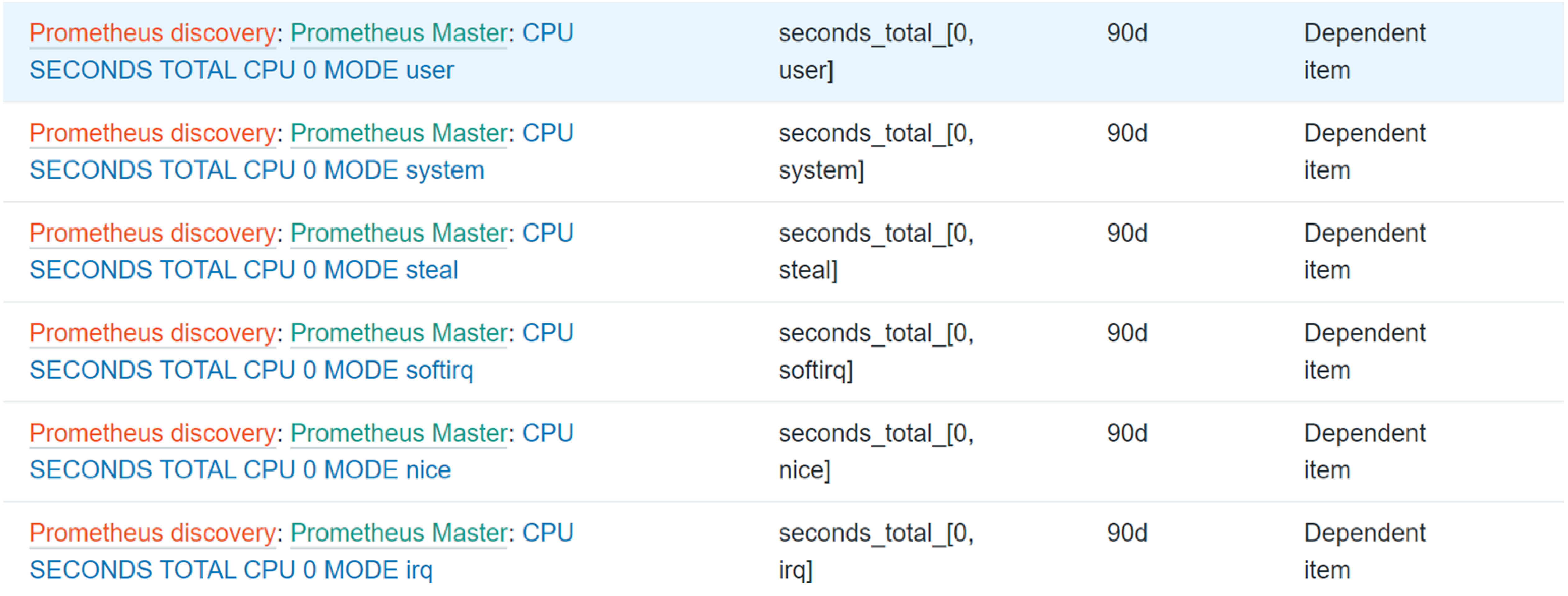
Use {#HELP} to populate the Description field:

```
"help":"Seconds the cpus spent in each mode."
```

Macro:{#HELP}
JSONPath:$.help

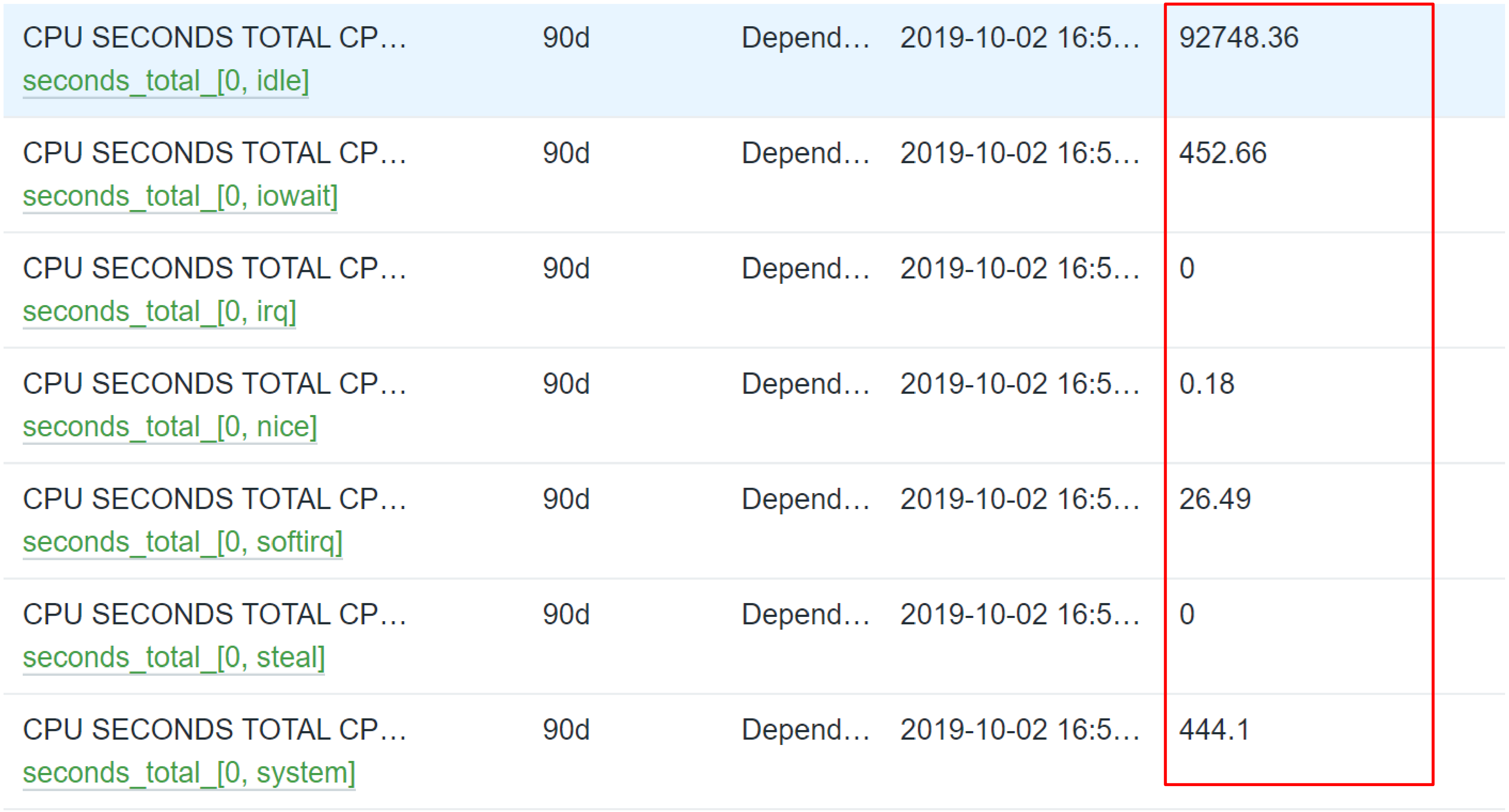

Description
Seconds the cpus spent in each mode.

PROMETHEUS LLD



Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE user	seconds_total_[0, user]	90d	Dependent item
Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE system	seconds_total_[0, system]	90d	Dependent item
Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE steal	seconds_total_[0, steal]	90d	Dependent item
Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE softirq	seconds_total_[0, softirq]	90d	Dependent item
Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE nice	seconds_total_[0, nice]	90d	Dependent item
Prometheus discovery: Prometheus Master: CPU SECONDS TOTAL CPU 0 MODE irq	seconds_total_[0, irq]	90d	Dependent item

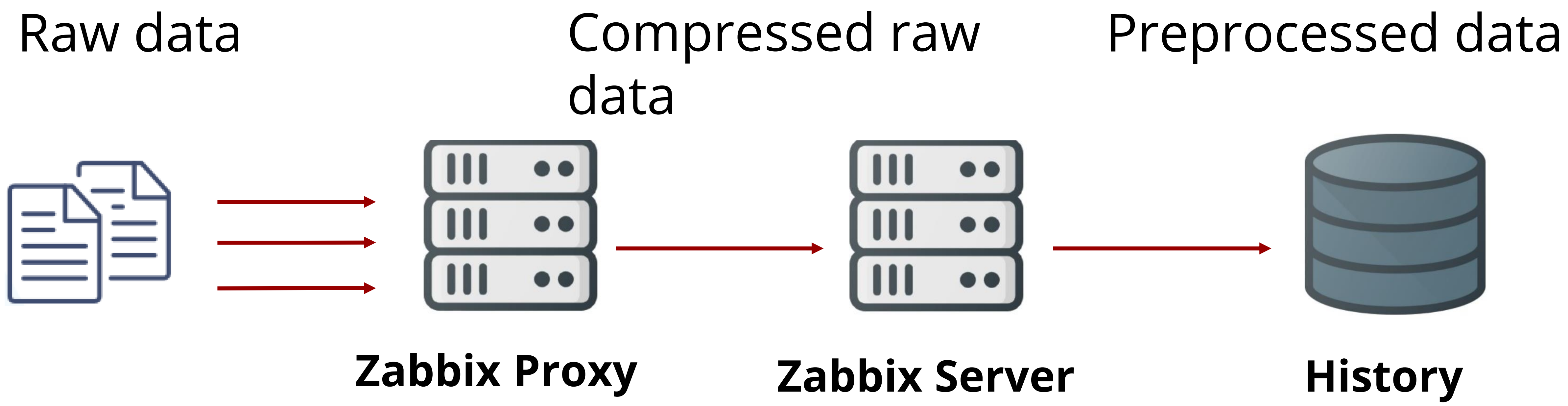
PROMETHEUS LLD



CPU SECONDS TOTAL CP… seconds_total_[0, idle]	90d	Depend…	2019-10-02 16:5…	92748.36
CPU SECONDS TOTAL CP… seconds_total_[0, iowait]	90d	Depend…	2019-10-02 16:5…	452.66
CPU SECONDS TOTAL CP… seconds_total_[0, irq]	90d	Depend…	2019-10-02 16:5…	0
CPU SECONDS TOTAL CP… seconds_total_[0, nice]	90d	Depend…	2019-10-02 16:5…	0.18
CPU SECONDS TOTAL CP… seconds_total_[0, softirq]	90d	Depend…	2019-10-02 16:5…	26.49
CPU SECONDS TOTAL CP… seconds_total_[0, steal]	90d	Depend…	2019-10-02 16:5…	0
CPU SECONDS TOTAL CP… seconds_total_[0, system]	90d	Depend…	2019-10-02 16:5…	444.1

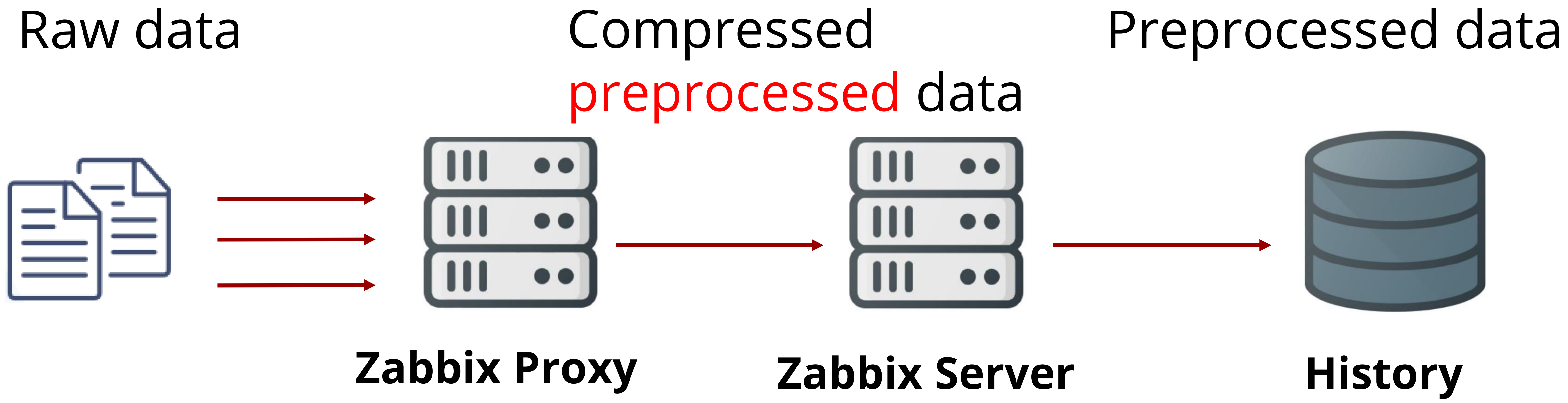
PREPROCESSING BEFORE VERSION 4.2

All of the preprocessing is being done by the server!



PREPROCESSING WITH VERSION >= 4.2

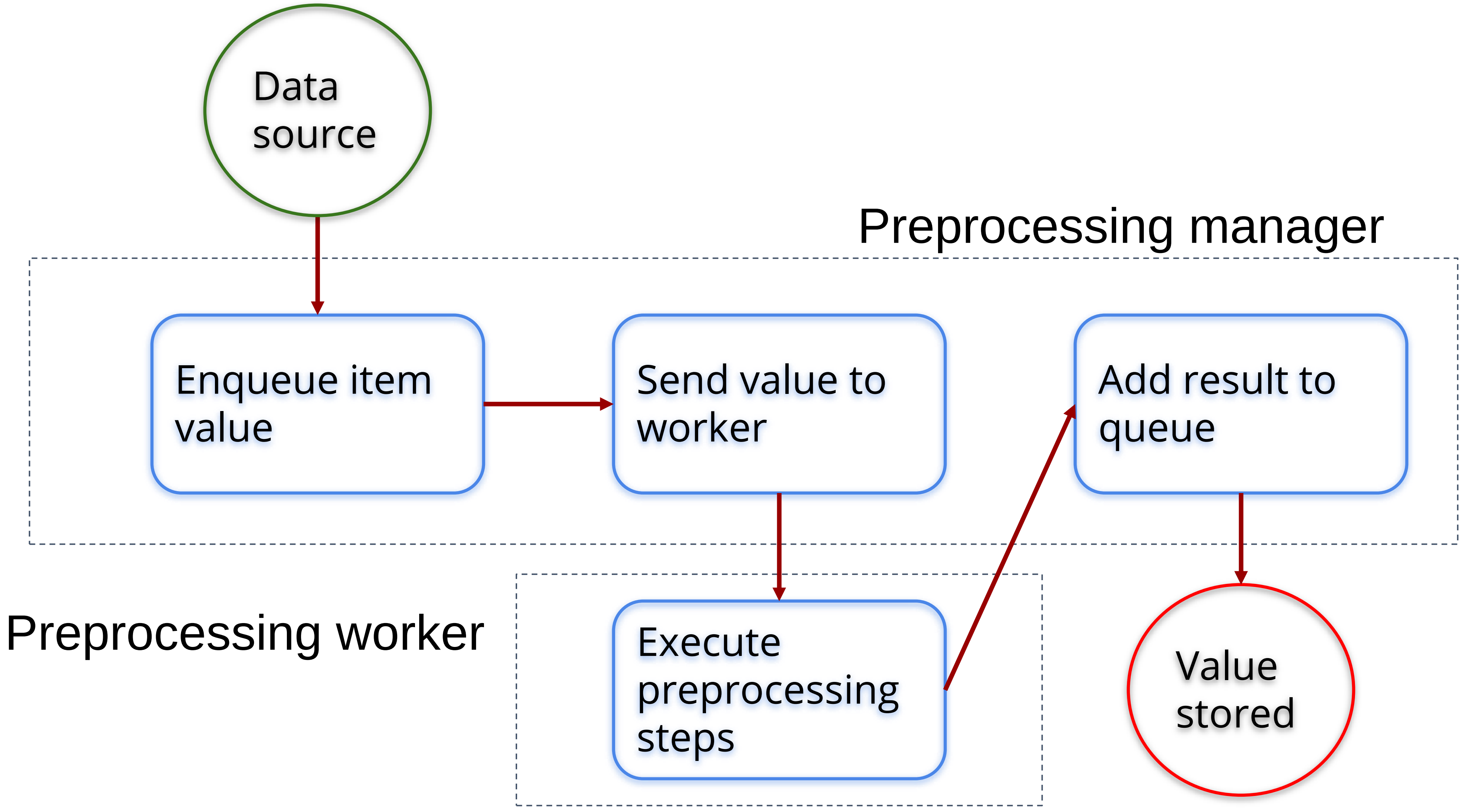
Preprocessing is performed on the proxy!



PREPROCESSING UNDER THE HOOD

DEEP DIVE IN
ZABBIX PRE-PROCESSING

PREPROCESSING WORKFLOW



PREPROCESSING MANAGER

- Added in version 3.4
- Enqueues the items in the preprocessing queue
- Assigns the preprocessing tasks to preprocessing workers
- Flushes the preprocessed values from the queue

PREPROCESSING WORKERS

- StartPreprocessors defines the value of pre-forked preprocessing workers
- Number of workers determined by:
    - Count of preprocessable items
    - Count of preprocessing steps
    - etc.

LET'S RECAP

Automate!

- Use master items with LLD!
- Discover your metrics with preprocessing!

Improve!

- Discard unnecessary data with throttling!
- Improve performance by preprocessing on the proxy!

Customize!

- Data validation!
- Custom behavior with advanced preprocessing rules!

Transform!

- Transform your data!
- Enable data aggregation and calculation with preprocessing!

ZABBIX SUMMIT '19

THANK YOU!

Arturs Lontons

ZABBIX Technical Support Engineer

ZABBIX SUMMIT '19

zabbix