ENGLISH

ZABBIX MEETUP ONLINE '22

WMI and Performance counter discovery and monitoring

Aleksandrs Petrovs-Gavrilovs
Technical Support Engineer, Zabbix

01

ZABBIX
6.0

WMI MONITORING

WHAT IS WMI?

Windows Management Instrumentation (**WMI**) is the infrastructure for data management and operations on Windows-based operating systems which can be used for:

- Managing remote computers
- Sharing management information between applications
- Accessing management data from any source in a uniform manner
- Monitoring Windows-based systems and networks
- Monitoring activities across an enterprise network as part of a user entity behaviour analytics (UEBA) system
- Monitoring anomalous events and potentially suspicious behaviours, and checking for insider threats

HOW TO RUN WMI QUERIES?

There are multiple ways to run WMI queries in Windows:

- Using WMIC (deprecated in newer Windows versions)
- Using Windows PowerShell
- Execute WMI based scripts
- Using Zabbix Agent

WMI QUERY EXAMPLES

ZABBIX 6.0

Before starting monitoring the output of the WMI queries, it is always a good idea to test them, WMI queries in Zabbix are executed using Windows Management Instrumentation Query Language (WQL) :

- So, syntax will be different, using WMIC i.e., to get information about installed software:

```
wmic:root\cli>product where "name like 'Zabbix%'" get name, version
```

- Unlike using PowerShell to get all avaiable information on running process (the WQL approach):

```
Get-WmiObject -query "SELECT * FROM Win32_Process"
```

- How to find all useful tables in Windows with information?

```
Get-WmiObject -query "Select * From Meta_Class"
```

OUTPUT EXAMPLES

ZABBIX 6.0

```
Name           : WebexHost.exe
KernelModeTime : 1562500
UserModeTime   : 625000
ProcessID      : 15792
WorkingSetSize : 22212608
PageFileUsage  : 11200
PageFaults     : 11861

Name           : chrome.exe
KernelModeTime : 557656250
UserModeTime   : 766250000
ProcessID      : 31796
WorkingSetSize : 269451264
PageFileUsage  : 165088
PageFaults     : 1386269
```

```
RefreshRateService                                      ASUSTeK COMPUTER INC.      2.1.0
Orca                                                    Microsoft Corporation      3.1.3790.0000
Microsoft Visual C++ 2012 x86 Additional Runtime - 11.0.61030  Microsoft Corporation  11.0.61030
Microsoft Visual C++ 2012 x64 Minimum Runtime - 11.0.61030     Microsoft Corporation  11.0.61030
paint.net                                               dotPDN LLC                 4.3.7
Microsoft Update Health Tools                           Microsoft Corporation      4.66.0.0
ASUS AURA Headset Component                             ASUSTek COMPUTER INC.      1.3.26.0
Microsoft Visual C++ 2013 x86 Additional Runtime - 12.0.40664  Microsoft Corporation  12.0.40664
Microsoft Visual C++ 2012 x86 Minimum Runtime - 11.0.61030     Microsoft Corporation  11.0.61030
Windows PC Health Check                                 Microsoft Corporation      3.2.2110.14001
```

02

ZABBIX 6.0

USING ZABBIX AGENT TO MONITOR WMI OUTPUT

HOW TO RUN WMI QUERIES IN ZABBIX?

We can use built-in agent keys to execute WMI queries:

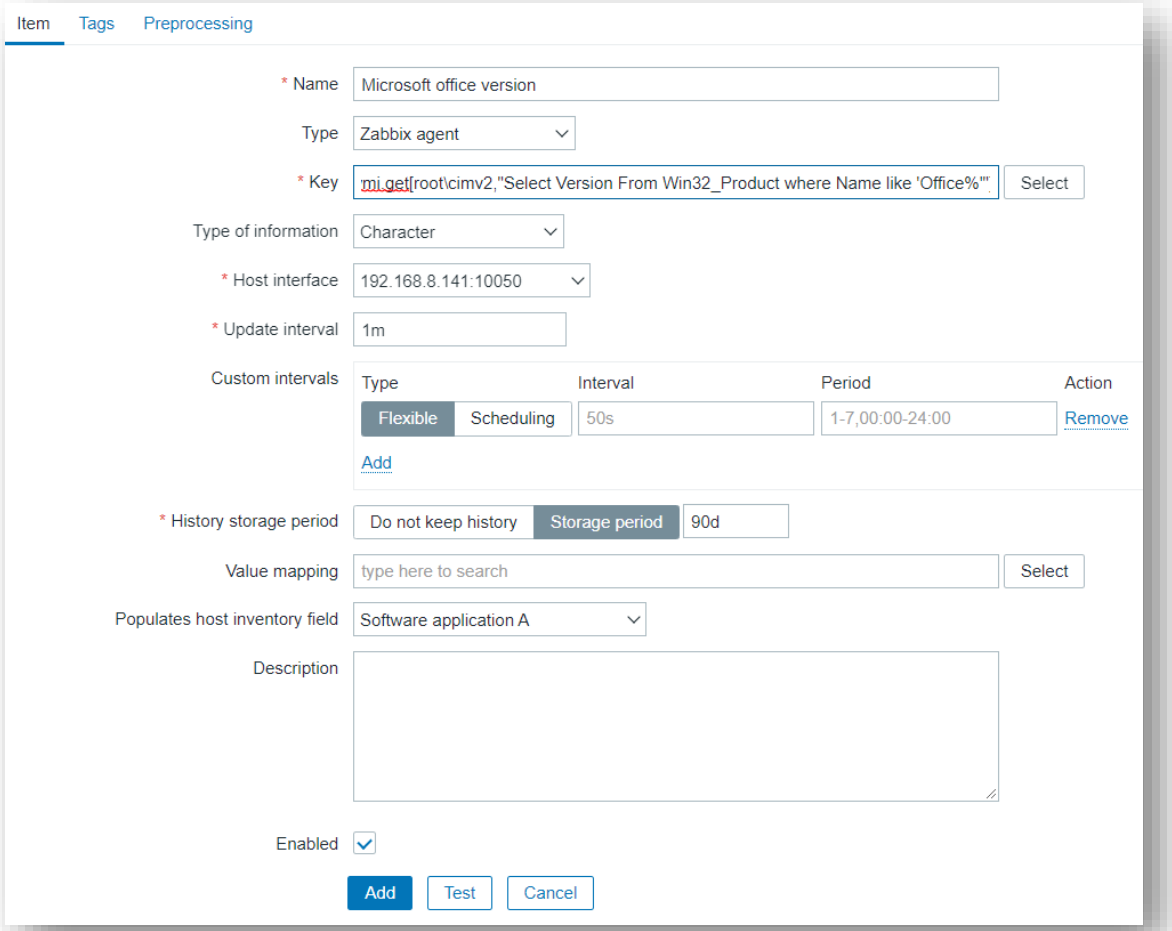
- wmi.get[<namespace>,<query>]
  - Execute WMI query and return the first selected object.
  - namespace - WMI namespace
  - query - WMI query returning a single object

```
Get-WmiObject -query "Select Version From Win32_Product where Name like 'Office%'"
```

USING THE WMI.GET

To start using WMI monitoring in Zabbix you need to

- Install Zabbix agent on the Windows machine
- Configure it for passive or active checks
- Create an wmi.get item to collect the data



USING THE WMI.GETALL TO GET MORE DATA

ZABBIX 6.0

We can use built-in agent keys to execute WMI queries not only to return single point of data, but to retrieve all information available:

- wmi.getall[<namespace>,<query>]
  - Execute WMI query and return everything.
  - namespace - WMI namespace
  - query - WMI query

```
Get-WmiObject -query "SELECT * FROM Win32_Process"
```

In this case the query will result into a JSON array

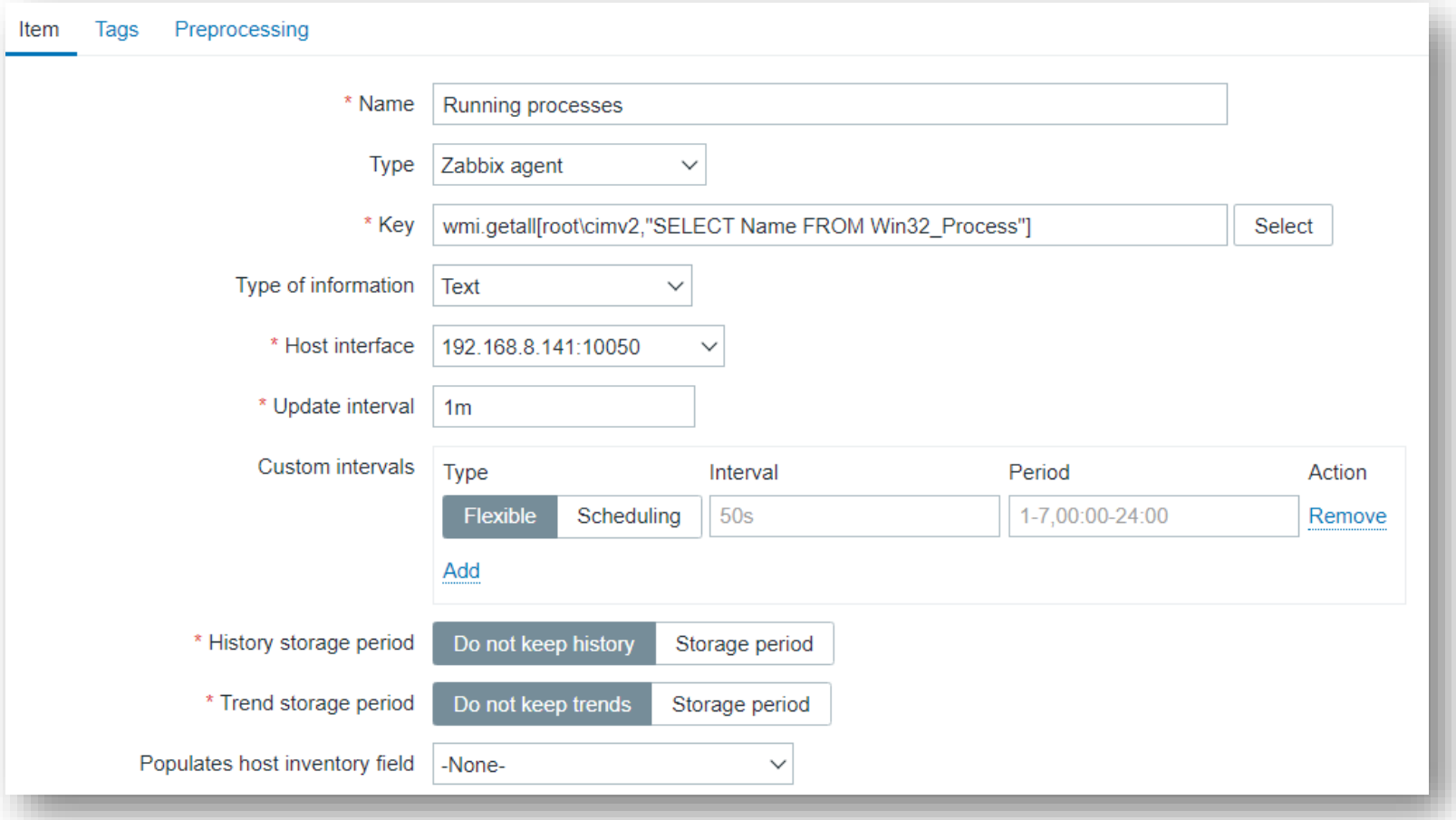
```
{"Handle": "25056",    "Name": "Teams.exe"},
{"Handle": "17720",    "Name": "AutoConnectHelper.exe"},
{"Handle": "15792",    "Name": "WebexHost.exe"},
{"Handle": "17796",    "Name": "atmgr.exe"},
{"Handle": "31796",    "Name": "chrome.exe"},
```

USING THE WMI.GETALL TO DISCOVER

ZABBIX 6.0

To start using WMI monitoring as discovery, you will need a Zabbix agent just as with wmi.get

- First thing you will need to do is create an item, to return information you are interested in. In this example those are names of all running processes:



USING THE WMI.GETALL TO DISCOVER

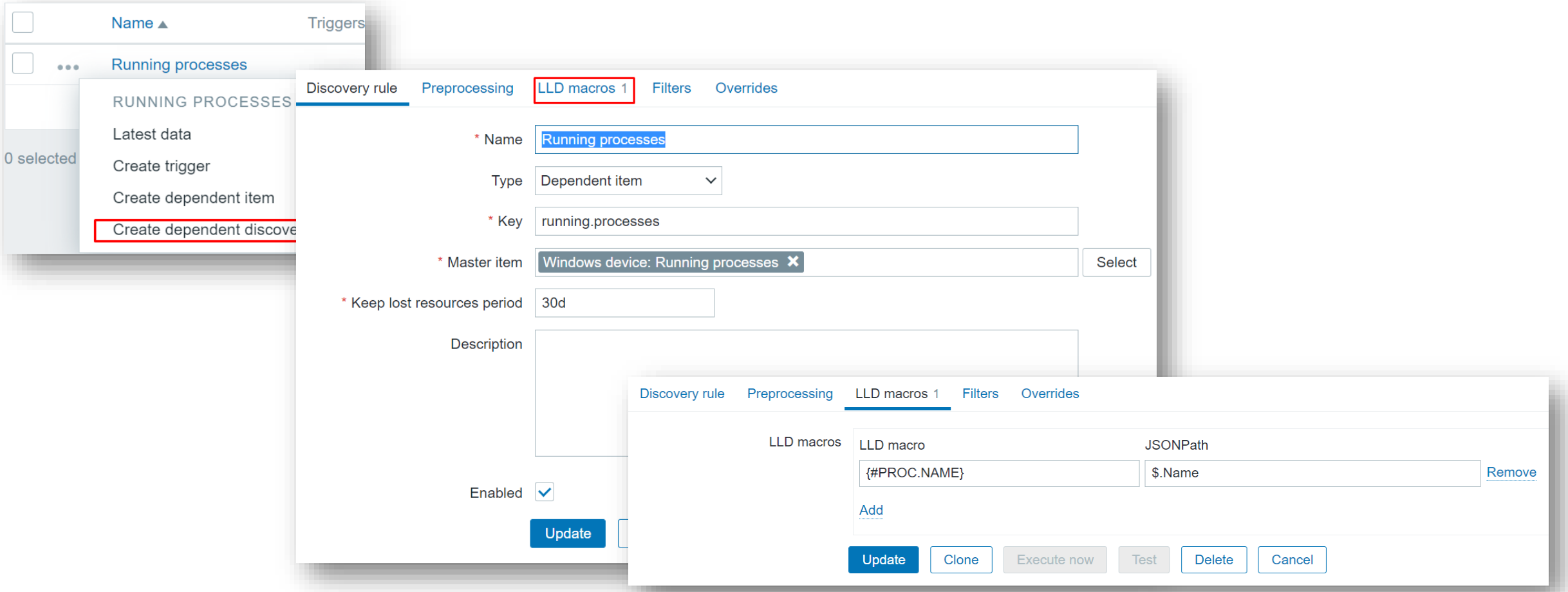
ZABBIX 6.0

This will give us output like this, which we can use to build a dependent discovery rule:


```
[
  {
    "Handle": "0",
    "Name": "System Idle Process"
  },
  {
    "Handle": "4",
    "Name": "System"
  },
  {
    "Handle": "1204",
    "Name": "csrss.exe"
  },
  {
    "Handle": "1352",
    "Name": "wininit.exe"
  },
```

USING THE WMI.GETALL TO DISCOVER

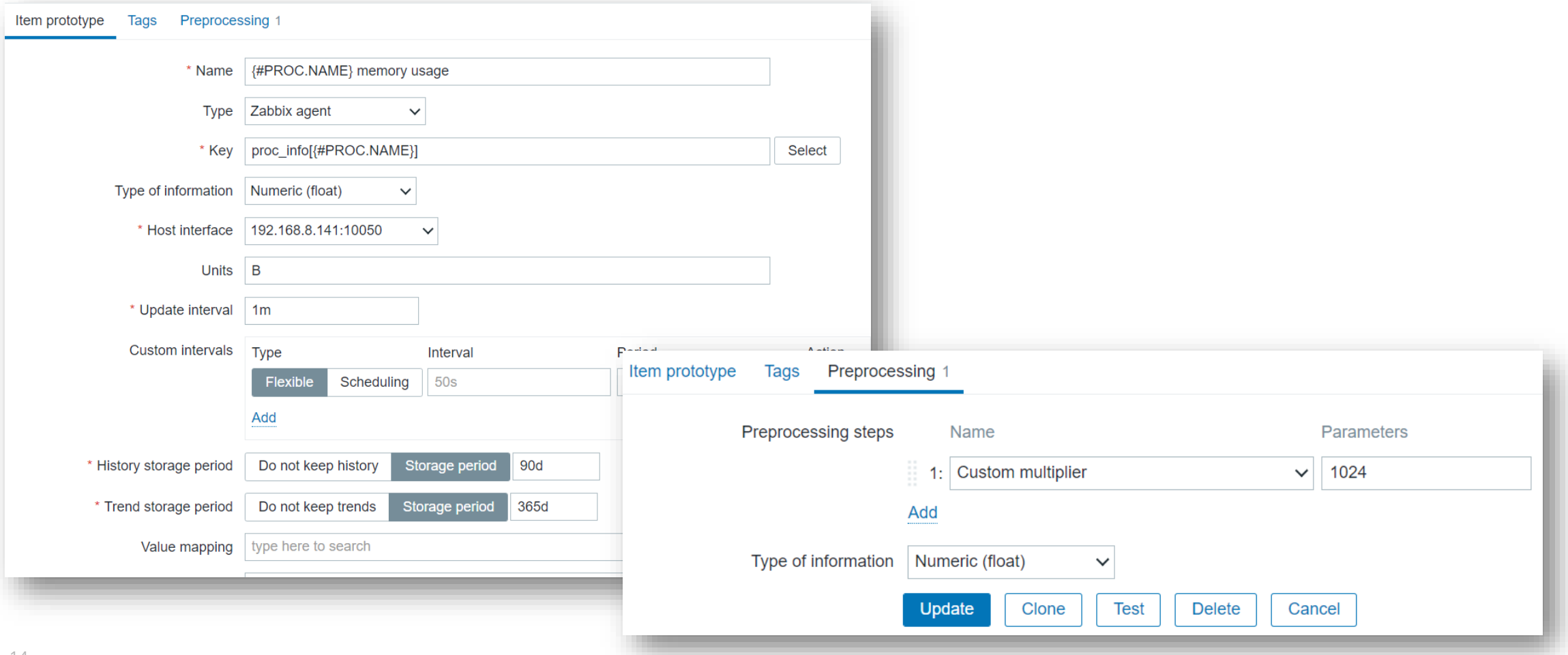
From the previous item, we create the dependent discovery rule, which with the help of LLD macro allows us to easily extract process name to use in item prototypes:



USING THE WMI.GETALL TO DISCOVER

ZABBIX 6.0

And for the item prototype, we now can use proc_info[process,<attribute>,<type>] key with our LLD macro to monitor memory usage (and a lot more) of all the currently running processes in Windows:



USING THE WMI.GETALL TO DISCOVER

ZABBIX 6.0

And this allows us to monitor all the processes memory usage in mere seconds:

Running processes: LightingService.exe memory usage	proc_info[LightingService.exe]	1m	90d	365d	Zabbix agent	Enabled
Running processes: Lightshot.exe memory usage	proc_info[Lightshot.exe]	1m	90d	365d	Zabbix agent	Enabled
Running processes: LockApp.exe memory usage	proc_info[LockApp.exe]	1m	90d	365d	Zabbix	Enabled
Running processes: lsass.exe memory usage						
Running processes: mcafee-security-ft.exe memory usage						
Running processes: mcafee-security.exe memory usage						
Running processes: Memory Compression memory usage						
Running processes: Microsoft.Photos.exe memory usage						
Running processes: mmc.exe memory usage						
Running processes: MpCopyAccelerator.exe memory usage						
Running processes: msedgewebview2.exe memory usage						

Windows device	chrome.exe memory usage	33s	78.75 MB	+993.3 KB
Windows device	cmd.exe memory usage	32s	2.28 MB	
Windows device	conhost.exe memory usage	49s	5.51 MB	-8.57 KB
Windows device	Cortana.exe memory usage	28s	30.64 MB	
Windows device	csrss.exe memory usage	4s	6.58 MB	
Windows device	ctfmon.exe memory usage	53s	4.47 MB	
Windows device	dasHost.exe memory usage	57s	10.27 MB	
Windows device	DAX3API.exe memory usage	39s	7.93 MB	+2 KB
Windows device	DCIService.exe memory usage	45s	59.7 MB	-836 KB
Windows device	dllhost.exe memory usage	25s	4.99 MB	
Windows device	dwm.exe memory usage	10s	254.66 MB	+1.02 MB
Windows device	EABackgroundService.exe memory usage	31s	9.59 MB	
Windows device	explorer.exe memory usage	1m	175.03 MB	-2.59 MB

03

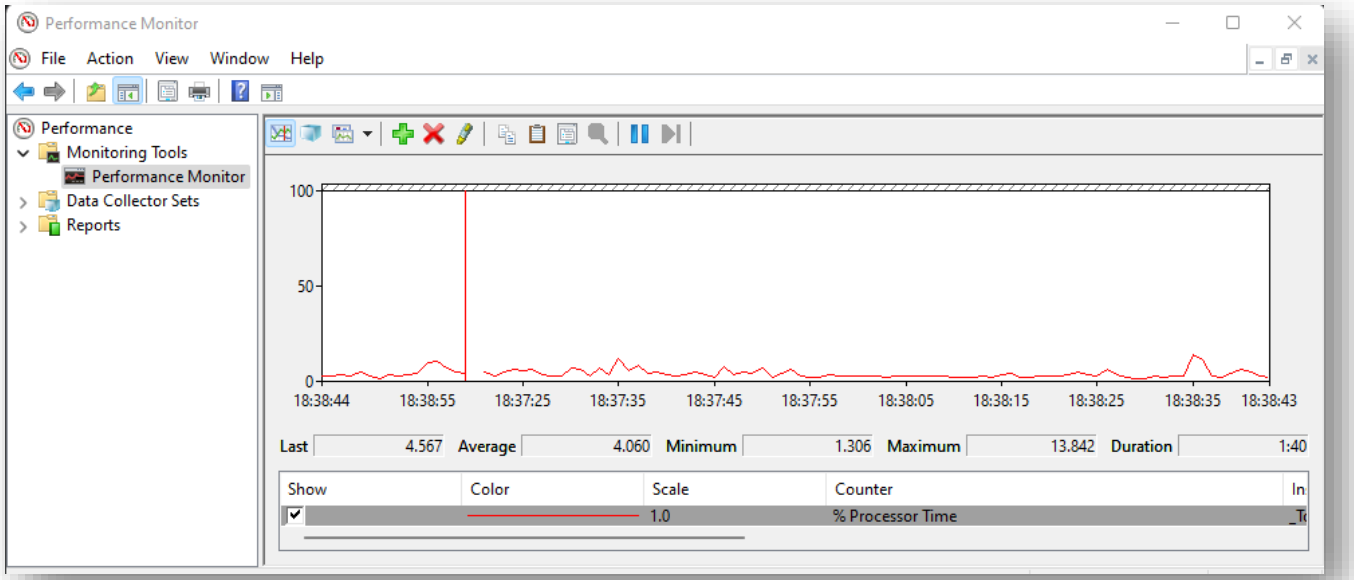
ZABBIX
6.0

PERFORMANCE COUNTER DISCOVERY AND MONITORING

WHAT ARE PERFORMANCE COUNTERS?

Windows Performance Counters provide a high-level abstraction layer that provides a consistent interface for collecting various kinds of system data:

- CPU, memory, and disk usage
- Overall system performance
- Behaviour problems
- Resource usage of programs



PERFORMANCE COUNTER MONITORING

ZABBIX 6.0

We can use built-in agent keys to gather information from performance counters:

- perf_counter[counter,<interval>]
- perf_counter_en[counter,<interval>] - performance counter in English
  - counter - path to the counter
  - interval - last N seconds for storing the average value

In order to get a full list of performance counters available for monitoring, you may run:

```
typeperf  -qx
```

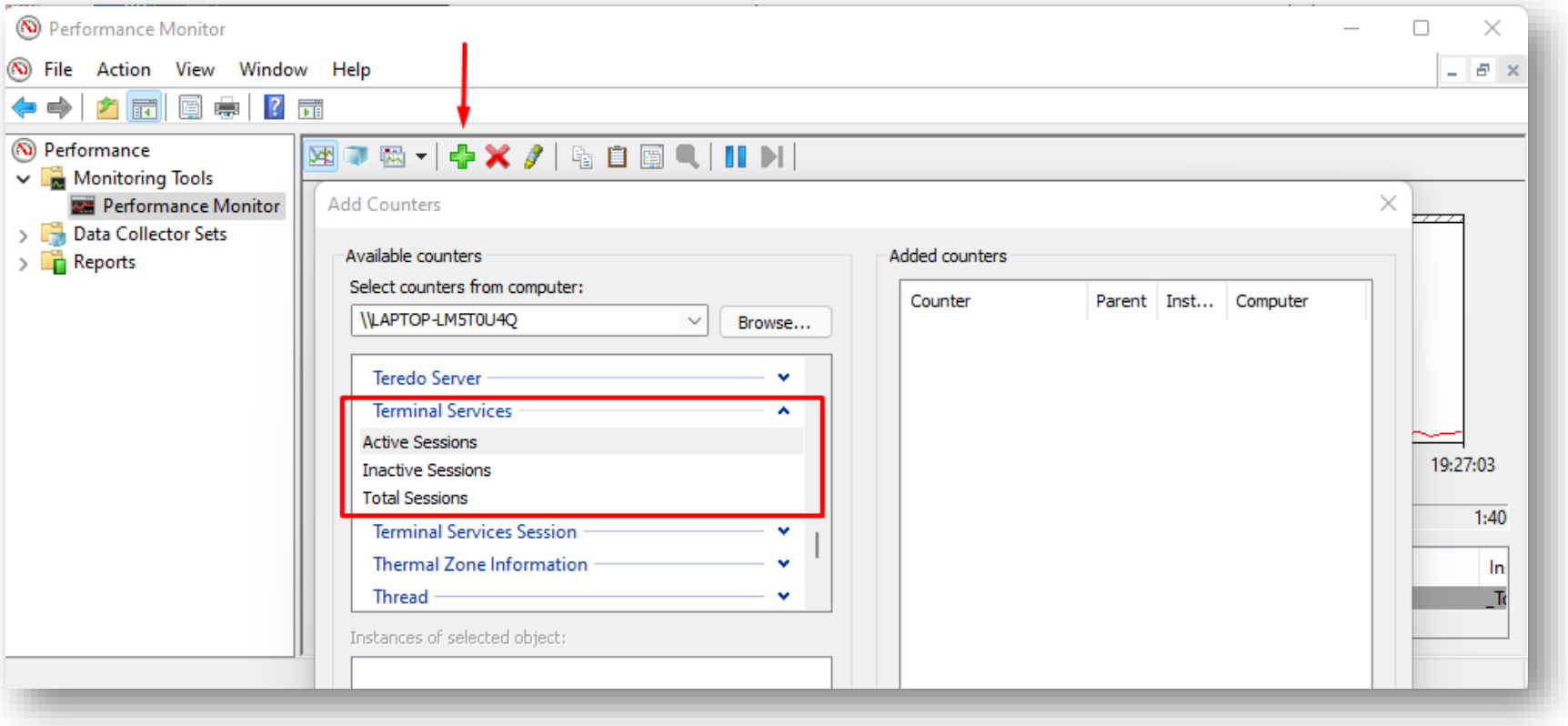
Or use performance monitor as mentioned previously

MONITORING PERFORMANCE COUNTERS

ZABBIX 6.0

To monitor performance counter in Zabbix, you will need a Zabbix agent

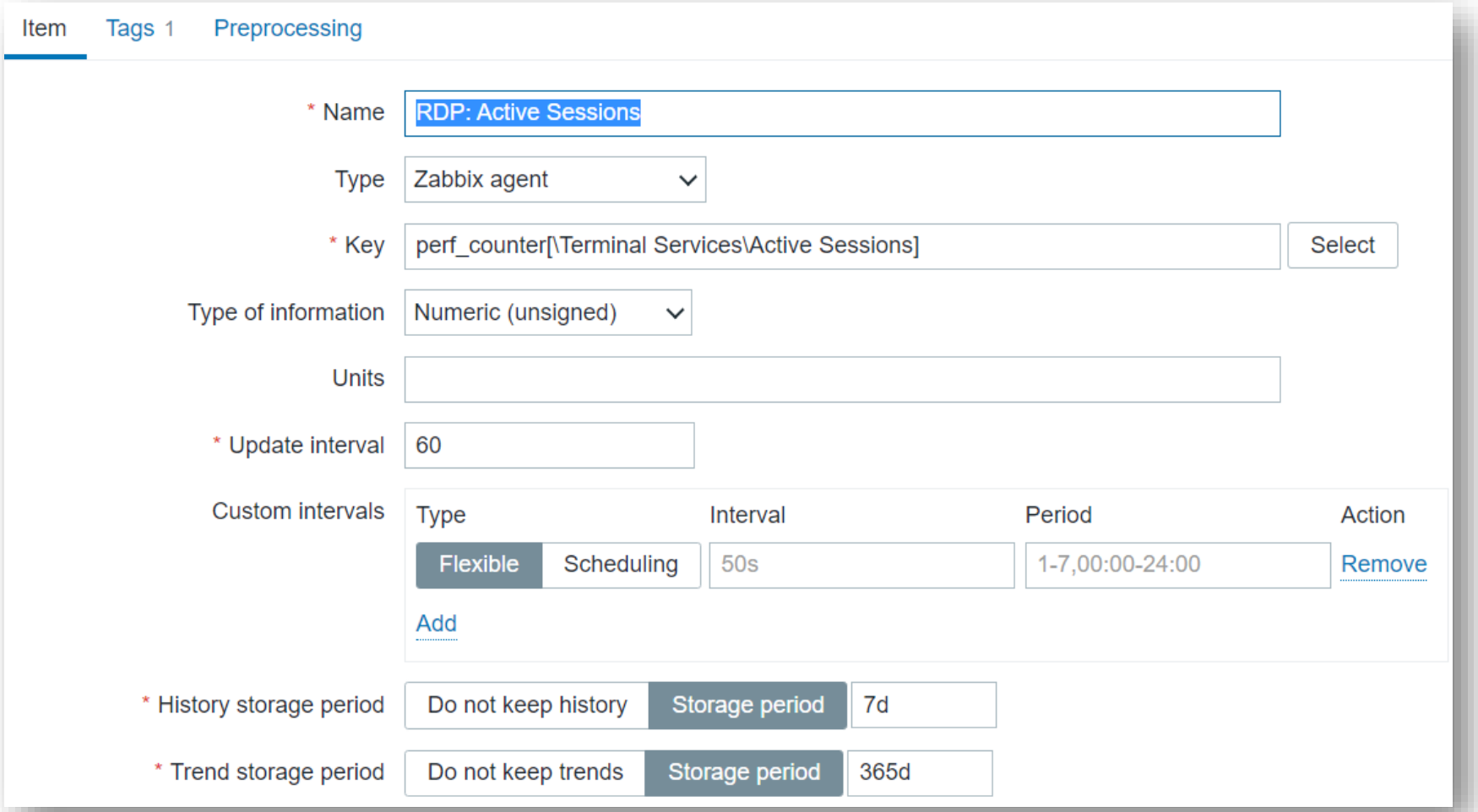
- First thing you will need to do is find the performance counter you are interested in using Parformance Monitor or CMD, let's say we want to monitor Terminal sessions:



MONITORING PERFORMANCE COUNTERS

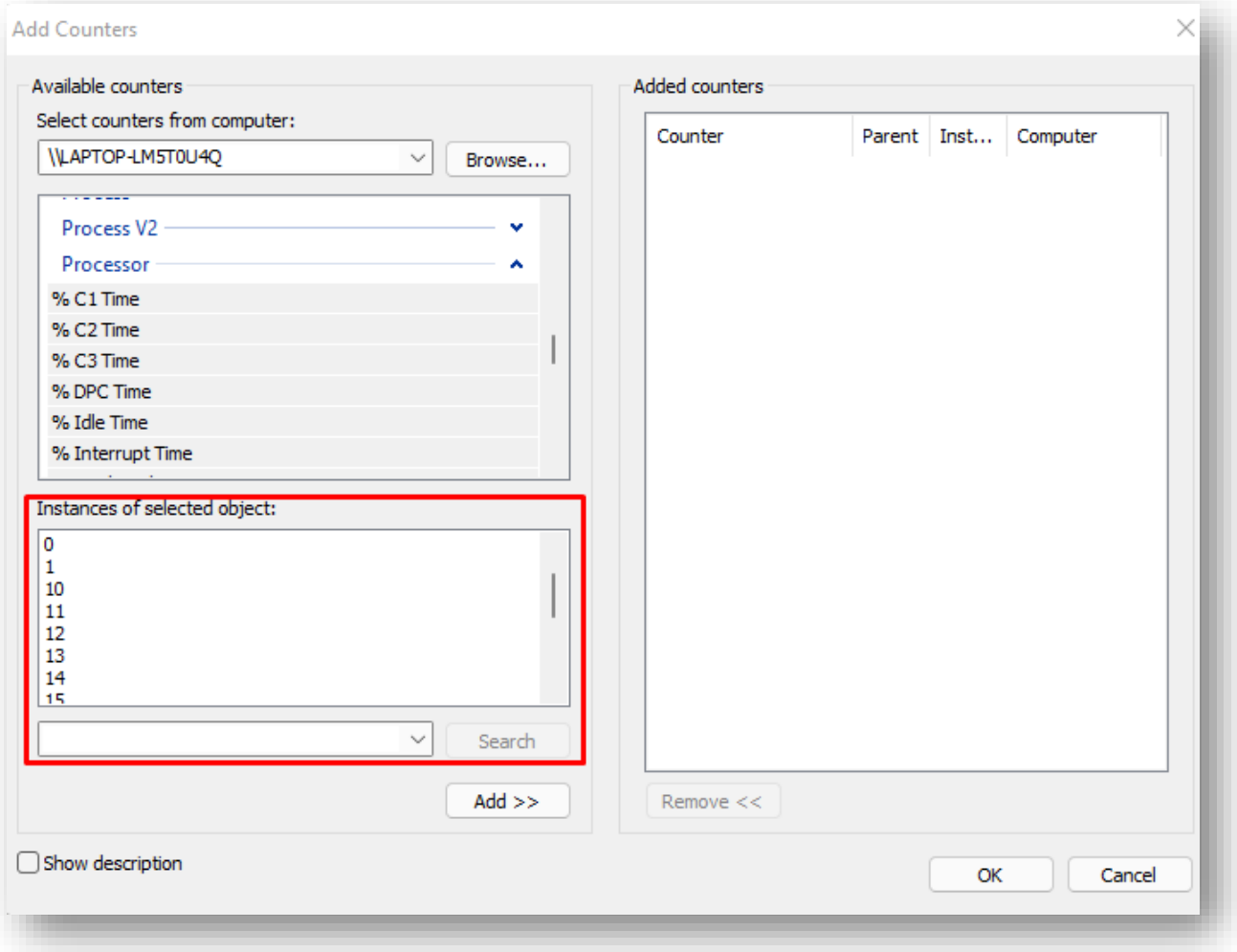
ZABBIX 6.0

Now the found the counter, we just add it to Zabbix using the perf_counter keys:



DISCOVERING COUNTER INSTANCES

Performance counter instance is an entity about which performance data is reported. An instance has a name (string) and one or more counter values :



PERFORMANCE COUNTER MONITORING

ZABBIX 6.0

It's possible to discover object instances of Windows performance counters:

- perf_instance.discovery[object]
- perf_instance_en.discovery[object] - in English
  - object – name of the object

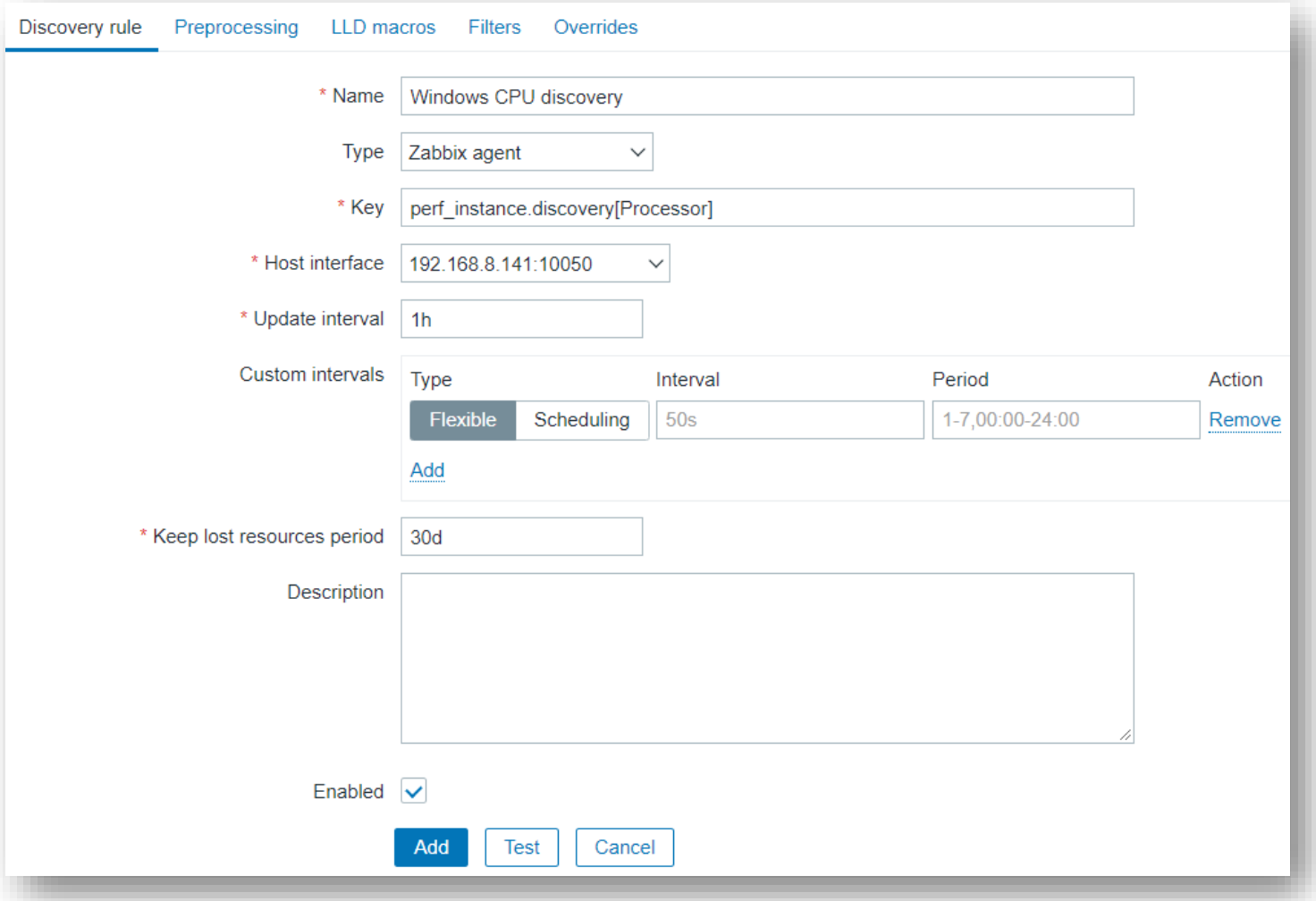
Meaning the using the key perf_instance.discovery[Processor], will give us output like:

```
[{    "{#INSTANCE}": "0"  },
 {    "{#INSTANCE}": "1"  },
 {    "{#INSTANCE}": "2"  },
 {    "{#INSTANCE}": "3"  },
 {    "{#INSTANCE}": "4"  },
 {    "{#INSTANCE}": "5"  },
 {    "{#INSTANCE}": "6"  },
 {    "{#INSTANCE}": "7"  },
 {    "{#INSTANCE}": "8"  },
 {    "{#INSTANCE}": "9"  },
 {    "{#INSTANCE}": "10"  },
 {    "{#INSTANCE}": "11"  },
 {    "{#INSTANCE}": "12"  },
 {    "{#INSTANCE}": "13"  },
 {    "{#INSTANCE}": "14"  },
 {    "{#INSTANCE}": "15"  },
 {    "{#INSTANCE}": "_Total"  }]
```

PERFORMANCE COUNTER MONITORING

ZABBIX 6.0

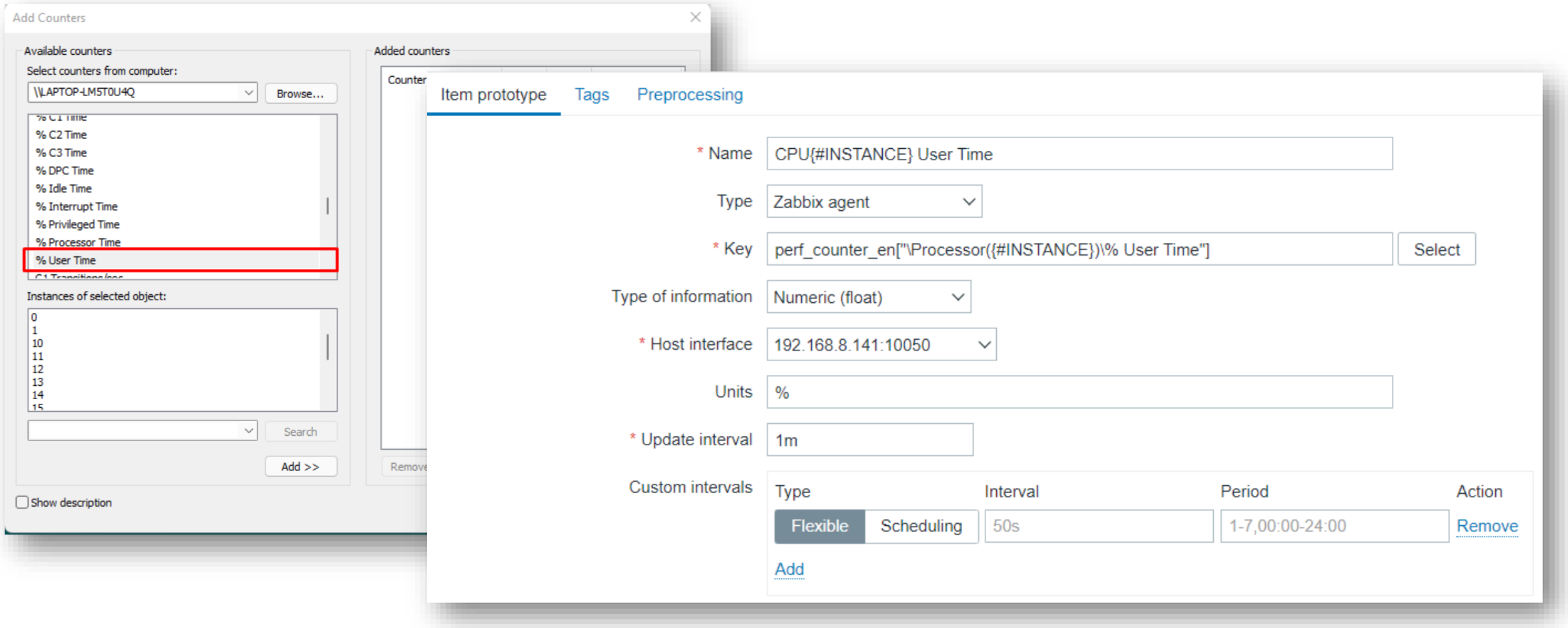
Meaning we can not create a discovery rule like:



PERFORMANCE COUNTER MONITORING

ZABBIX 6.0

And add an item prototype to monitor how busy each CPU is with user applications based on information from Performance Monitor:



PERFORMANCE COUNTER MONITORING

ZABBIX 6.0

Which will result in monitoring each CPU business with User Applications:

Windows CPU discovery: CPU0 User Time	perf_counter_en["\Processor(0)\% User Time"]	
Windows CPU discovery: CPU1 User Time	perf_counter_en["\Processor(1)\% User Time"]	
Windows CPU discovery: CPU2 User Time		
Windows CPU discovery: CPU3 User Time		
Windows CPU discovery: CPU4 User Time		
Windows CPU discovery: CPU5 User Time		
Windows CPU discovery: CPU6 User Time		
Windows CPU discovery: CPU7 User Time		
Windows CPU discovery: CPU8 User Time		
Windows CPU discovery: CPU9 User Time		
Windows CPU discovery: CPU10 User Time		
Windows CPU discovery: CPU11 User Time		
Windows CPU discovery: CPU12 User Time		
Windows CPU discovery: CPU13 User Time		
Windows CPU discovery: CPU14 User Time		
Windows CPU discovery: CPU15 User Time		

Windows device	CPU0 User Time	8s	0 %	
Windows device	CPU1 User Time	7s	0 %	
Windows device	CPU2 User Time	6s	14.0504 %	+14.0504 %
Windows device	CPU3 User Time	5s	1.5647 %	+1.5647 %
Windows device	CPU4 User Time	4s	31.414 %	+31.414 %
Windows device	CPU5 User Time	3s	1.5612 %	+1.5612 %
Windows device	CPU6 User Time	2s	0 %	
Windows device	CPU7 User Time	1s	1.5702 %	+0.007076 %
Windows device	CPU8 User Time	1m	1.5542 %	+1.5542 %
Windows device	CPU9 User Time	59s	0 %	
Windows device	CPU10 User Time	58s	4.6793 %	-12.5278 %
Windows device	CPU11 User Time	57s	1.5709 %	+0.0101 %
Windows device	CPU12 User Time	56s	0 %	-1.5643 %
Windows device	CPU13 User Time	55s	0 %	
Windows device	CPU14 User Time	54s	0 %	
Windows device	CPU15 User Time	53s	26.7157 %	-6.2254 %

ZABBIX

6.0

Questions?

www.zabbix.com

ZABBIX

6.0

Thank you

www.zabbix.com