

ENGLISH



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

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 available 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

```
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

The screenshot displays the Zabbix web interface for configuring a new item. The 'Preprocessing' tab is active. The configuration is as follows:

- Name:** Microsoft office version
- Type:** Zabbix agent
- Key:** wmi.get[rootcimv2,"Select Version From Win32_Product where Name like 'Office%'", Select
- Type of information:** Character
- Host interface:** 192.168.8.141:10050
- Update interval:** 1m
- Custom intervals:** A table with columns Type, Interval, Period, and Action. One entry is shown: Flexible Scheduling 50s 1-7,00:00-24:00 Remove. An Add button is below.
- History storage period:** Do not keep history Storage period 90d
- Value mapping:** type here to search Select
- Populates host inventory field:** Software application A
- Description:** (Empty text area)
- Enabled:**

Buttons at the bottom: Add, Test, Cancel.

USING THE **WMI.GETALL** TO GET MORE DATA

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

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:

The screenshot shows the Zabbix web interface for configuring a new item. The 'Item' tab is selected, and the 'Preprocessing' sub-tab is active. The configuration is as follows:

- Name:** Running processes
- Type:** Zabbix agent
- Key:** wmi.getall[root\cimv2,"SELECT Name FROM Win32_Process"] (with a 'Select' button)
- Type of information:** Text
- Host interface:** 192.168.8.141:10050
- Update interval:** 1m
- Custom intervals:** A table with one entry:

Type	Interval	Period	Action
Flexible Scheduling	50s	1-7,00:00-24:00	Remove

An 'Add' link is located below the table.
- History storage period:** Do not keep history (selected) / Storage period
- Trend storage period:** Do not keep trends (selected) / Storage period
- Populates host inventory field:** -None-

USING THE WMI.GETALL TO DISCOVER

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:

The screenshot displays the Zabbix configuration interface for a dependent discovery rule. The main window is titled 'Running processes' and has tabs for 'Discovery rule', 'Preprocessing', 'LLD macros 1', 'Filters', and 'Overrides'. The 'LLD macros 1' tab is active, showing a table with the following content:

LLD macro	JSONPath	Remove
{#PROC.NAME}	\$.Name	Remove

Below the table are buttons for 'Add', 'Update', 'Clone', 'Execute now', 'Test', 'Delete', and 'Cancel'. The main configuration form includes the following fields:

- Name:** Running processes
- Type:** Dependent item
- Key:** running.processes
- Master item:** Windows device: Running processes
- Keep lost resources period:** 30d
- Description:** (empty text area)
- Enabled:**

A context menu is open over the 'Running processes' item, with 'Create dependent discovery rule' highlighted. The 'Update' button at the bottom of the configuration form is also visible.

USING THE WMI.GETALL TO DISCOVER

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:

The image shows two overlapping screenshots from the Zabbix web interface. The background screenshot displays the 'Item prototype' configuration page for 'Preprocessing 1'. The foreground screenshot shows the 'Preprocessing steps' configuration for the same item.

Item prototype configuration (Background):

- Name: `{#PROC.NAME} memory usage`
- Type: Zabbix agent
- Key: `proc_info[{#PROC.NAME}]`
- Type of information: Numeric (float)
- Host interface: 192.168.8.141:10050
- Units: B
- Update interval: 1m
- Custom intervals: Flexible (50s)
- History storage period: Storage period (90d)
- Trend storage period: Storage period (365d)

Preprocessing steps configuration (Foreground):

Preprocessing steps	Name	Parameters
1:	Custom multiplier	1024

Type of information: Numeric (float)

Buttons: Update, Clone, Test, Delete, Cancel

USING THE WMI.GETALL TO DISCOVER

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

<input type="checkbox"/>	... Running processes: LightingService.exe memory usage	proc_info[LightingService.exe]	1m	90d	365d	Zabbix agent	Enabled
<input type="checkbox"/>	... Running processes: Lightshot.exe memory usage	proc_info[Lightshot.exe]	1m	90d	365d	Zabbix agent	Enabled
<input type="checkbox"/>	... Running processes: LockApp.exe memory usage	proc_info[LockApp.exe]	1m	90d	365d	Zabbix	Enabled
<input type="checkbox"/>	... Running processes: Isass.exe memory usage	<input type="checkbox"/> Windows device	chrome.exe memory usage	33s	78.75 MB	+993.3 KB	
<input type="checkbox"/>	... Running processes: mcafee-security-ft.exe memory usage	<input type="checkbox"/> Windows device	cmd.exe memory usage	32s	2.28 MB		
<input type="checkbox"/>	... Running processes: mcafee-security.exe memory usage	<input type="checkbox"/> Windows device	conhost.exe memory usage	49s	5.51 MB	-8.57 KB	
<input type="checkbox"/>	... Running processes: Memory Compression memory usage	<input type="checkbox"/> Windows device	Cortana.exe memory usage	28s	30.64 MB		
<input type="checkbox"/>	... Running processes: Microsoft.Photos.exe memory usage	<input type="checkbox"/> Windows device	csrss.exe memory usage	4s	6.58 MB		
<input type="checkbox"/>	... Running processes: mmc.exe memory usage	<input type="checkbox"/> Windows device	ctfmon.exe memory usage	53s	4.47 MB		
<input type="checkbox"/>	... Running processes: MpCopyAccelerator.exe memory usage	<input type="checkbox"/> Windows device	dasHost.exe memory usage	57s	10.27 MB		
<input type="checkbox"/>	... Running processes: msedgewebview2.exe memory usage	<input type="checkbox"/> Windows device	DAX3API.exe memory usage	39s	7.93 MB	+2 KB	
		<input type="checkbox"/> Windows device	DCIService.exe memory usage	45s	59.7 MB	-836 KB	
		<input type="checkbox"/> Windows device	dllhost.exe memory usage	25s	4.99 MB		
		<input type="checkbox"/> Windows device	dwm.exe memory usage	10s	254.66 MB	+1.02 MB	
		<input type="checkbox"/> Windows device	EABackgroundService.exe memory usage	31s	9.59 MB		
		<input type="checkbox"/> Windows device	explorer.exe memory usage	1m	175.03 MB	-2.59 MB	

03

ZABBIX

6.0

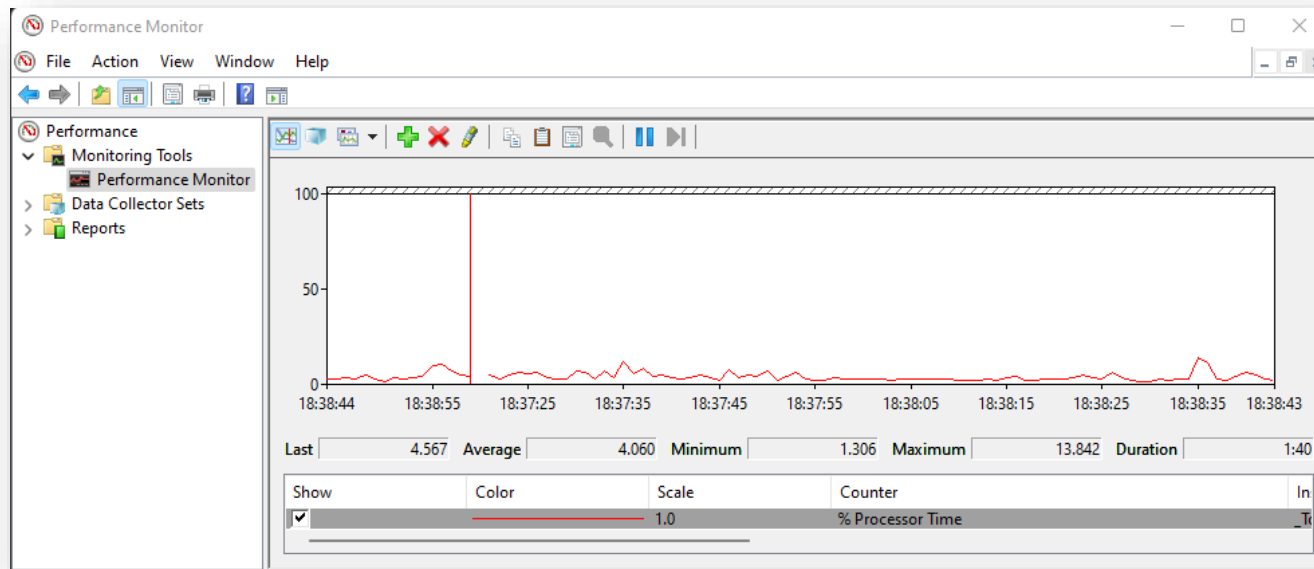
PERFORMANCE COUNTER DISCOVERY AND MONITORING



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

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:

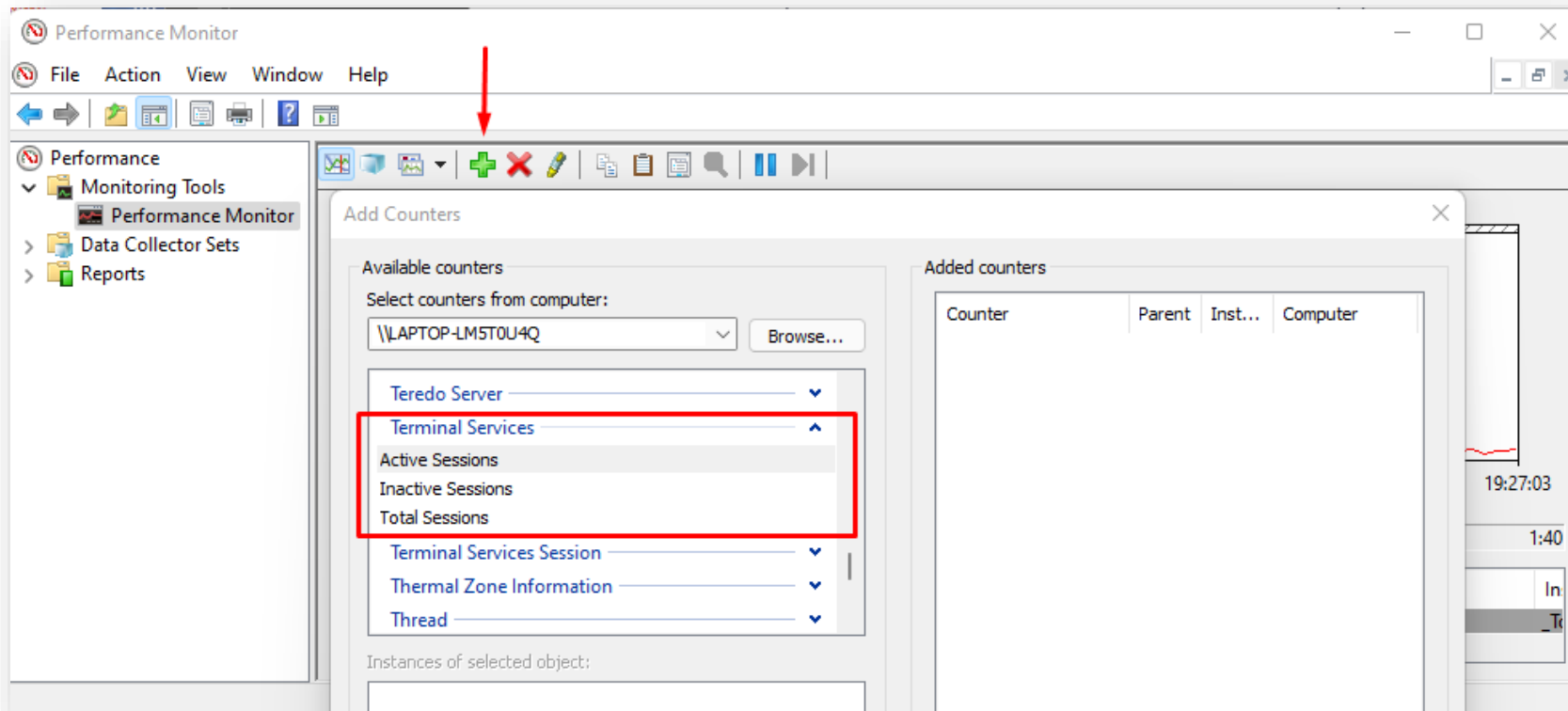
```
typeperf -qx
```

Or use performance monitor as mentioned previously

MONITORING PERFORMANCE COUNTERS

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 Performance Monitor or CMD, let's say we want to monitor Terminal sessions:



MONITORING PERFORMANCE COUNTERS

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

Item Tags 1 Preprocessing

* Name

Type

* Key

Type of information

Units

* Update interval

Custom intervals

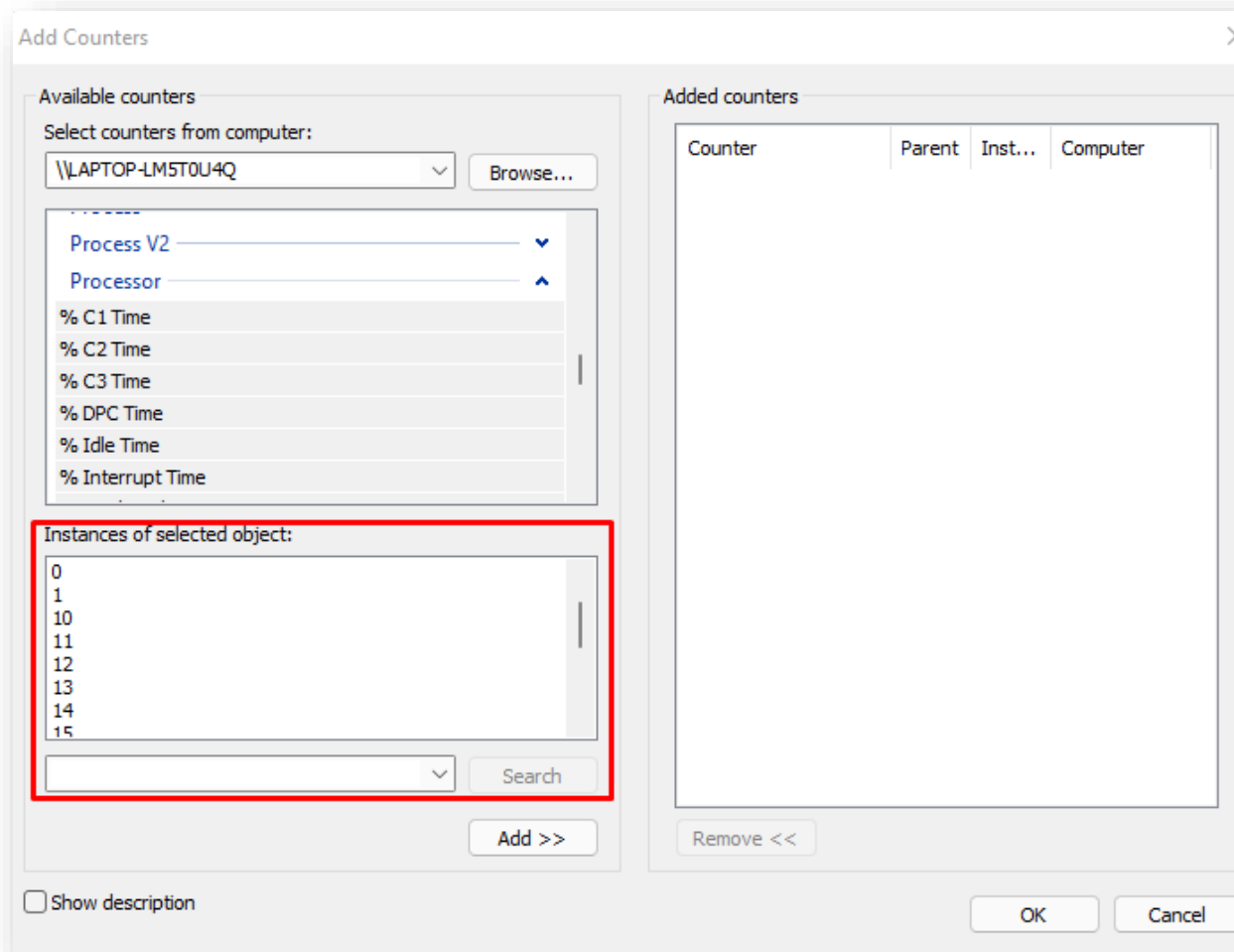
Type	Interval	Period	Action
<input checked="" type="checkbox"/> Flexible <input type="checkbox"/> Scheduling	<input type="text" value="50s"/>	<input type="text" value="1-7,00:00-24:00"/>	Remove
Add			

* History storage period

* Trend storage period

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

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

Meaning we can not create a discovery rule like:

The screenshot shows the Zabbix Discovery Rule configuration interface. The tabs at the top are 'Discovery rule', 'Preprocessing', 'LLD macros', 'Filters', and 'Overrides'. The 'Discovery rule' tab is active.

Fields and values:

- * Name: Windows CPU discovery
- Type: Zabbix agent
- * Key: perf_instance.discovery[Processor]
- * Host interface: 192.168.8.141:10050
- * Update interval: 1h
- * Keep lost resources period: 30d
- Enabled:

Custom intervals table:

Type	Interval	Period	Action
Flexible Scheduling	50s	1-7,00:00-24:00	Remove

Buttons: Add, Test, Cancel

PERFORMANCE COUNTER MONITORING

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

The image shows two overlapping windows from the Zabbix web interface. The background window is titled 'Add Counters' and shows a list of available performance counters for the computer '\LAPTOP-LM5T0U4Q'. The counter '% User Time' is highlighted with a red box. The foreground window is titled 'Item prototype' and shows the configuration for a new item. The configuration includes:

- Name:** CPU{#INSTANCE} User Time
- Type:** Zabbix agent
- Key:** perf_counter_en["\Processor({#INSTANCE})\% User Time"]
- Type of information:** Numeric (float)
- Host interface:** 192.168.8.141:10050
- Units:** %
- Update interval:** 1m
- Custom intervals:** A table with columns for Type, Interval, Period, and Action.

Type	Interval	Period	Action
Flexible	Scheduling	50s	1-7,00:00-24:00

There is an 'Add' link below the custom intervals table and a 'Remove' link next to the last row.

PERFORMANCE COUNTER MONITORING

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

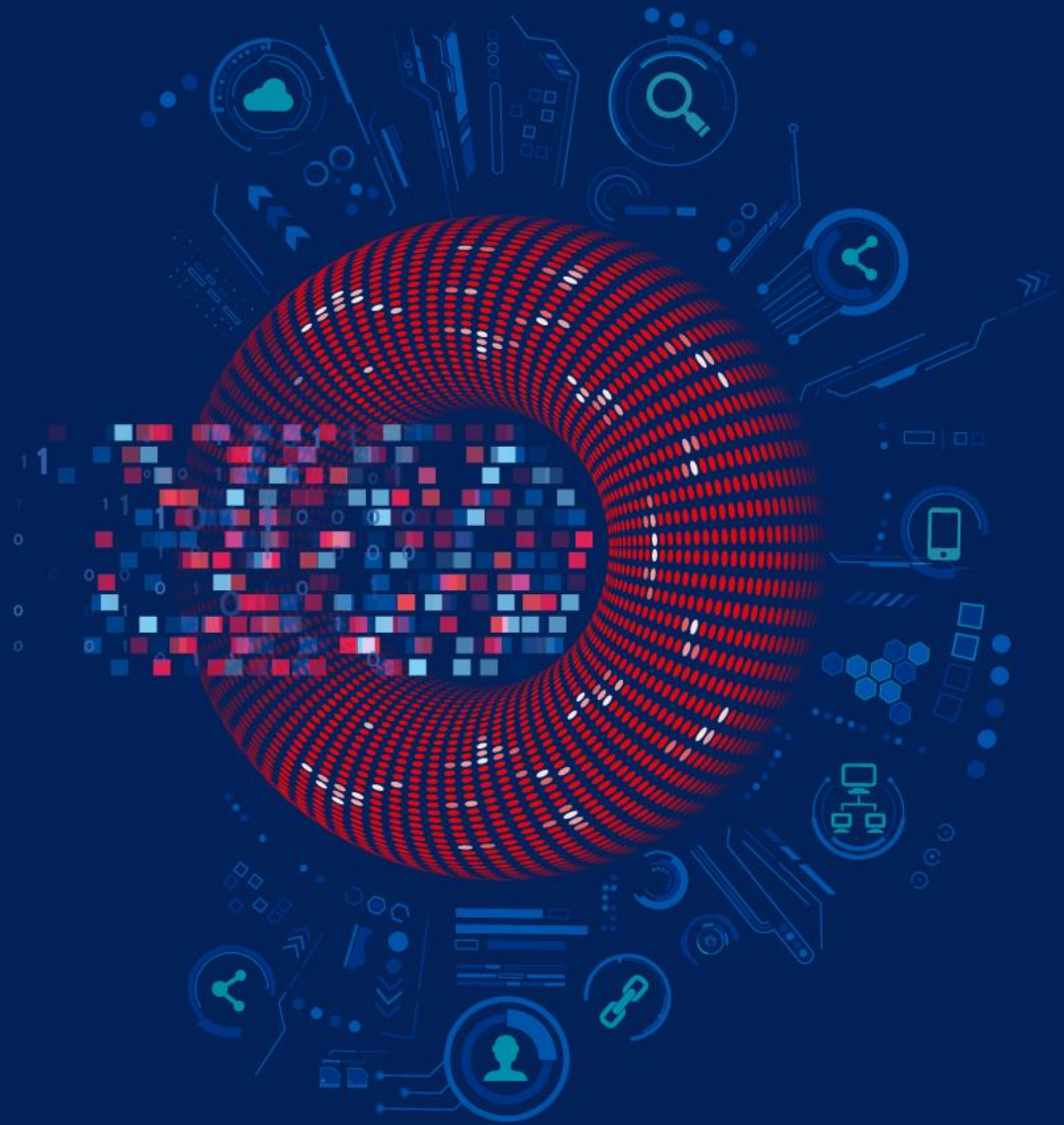
<input type="checkbox"/>	... Windows CPU discovery: CPU0 User Time		perf_counter_en["\Processor(0)\% User Time"]				
<input type="checkbox"/>	... Windows CPU discovery: CPU1 User Time		perf_counter_en["\Processor(1)\% User Time"]				
<input type="checkbox"/>	... Windows CPU discovery: CPU2 User Time	<input type="checkbox"/>	Windows device	CPU0 User Time	8s	0 %	
<input type="checkbox"/>	... Windows CPU discovery: CPU3 User Time	<input type="checkbox"/>	Windows device	CPU1 User Time	7s	0 %	
<input type="checkbox"/>	... Windows CPU discovery: CPU4 User Time	<input type="checkbox"/>	Windows device	CPU2 User Time	6s	14.0504 %	+14.0504 %
<input type="checkbox"/>	... Windows CPU discovery: CPU5 User Time	<input type="checkbox"/>	Windows device	CPU3 User Time	5s	1.5647 %	+1.5647 %
<input type="checkbox"/>	... Windows CPU discovery: CPU6 User Time	<input type="checkbox"/>	Windows device	CPU4 User Time	4s	31.414 %	+31.414 %
<input type="checkbox"/>	... Windows CPU discovery: CPU7 User Time	<input type="checkbox"/>	Windows device	CPU5 User Time	3s	1.5612 %	+1.5612 %
<input type="checkbox"/>	... Windows CPU discovery: CPU8 User Time	<input type="checkbox"/>	Windows device	CPU6 User Time	2s	0 %	
<input type="checkbox"/>	... Windows CPU discovery: CPU9 User Time	<input type="checkbox"/>	Windows device	CPU7 User Time	1s	1.5702 %	+0.007076 %
<input type="checkbox"/>	... Windows CPU discovery: CPU10 User Time	<input type="checkbox"/>	Windows device	CPU8 User Time	1m	1.5542 %	+1.5542 %
<input type="checkbox"/>	... Windows CPU discovery: CPU11 User Time	<input type="checkbox"/>	Windows device	CPU9 User Time	59s	0 %	
<input type="checkbox"/>	... Windows CPU discovery: CPU12 User Time	<input type="checkbox"/>	Windows device	CPU10 User Time	58s	4.6793 %	-12.5278 %
<input type="checkbox"/>	... Windows CPU discovery: CPU13 User Time	<input type="checkbox"/>	Windows device	CPU11 User Time	57s	1.5709 %	+0.0101 %
<input type="checkbox"/>	... Windows CPU discovery: CPU14 User Time	<input type="checkbox"/>	Windows device	CPU12 User Time	56s	0 %	-1.5643 %
<input type="checkbox"/>	... Windows CPU discovery: CPU15 User Time	<input type="checkbox"/>	Windows device	CPU13 User Time	55s	0 %	
<input type="checkbox"/>	... Windows CPU discovery: CPU14 User Time	<input type="checkbox"/>	Windows device	CPU14 User Time	54s	0 %	
<input type="checkbox"/>	... Windows CPU discovery: CPU15 User Time	<input type="checkbox"/>	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

