ZABBIX SUMMIT 2024

DNS improvements in 7.0

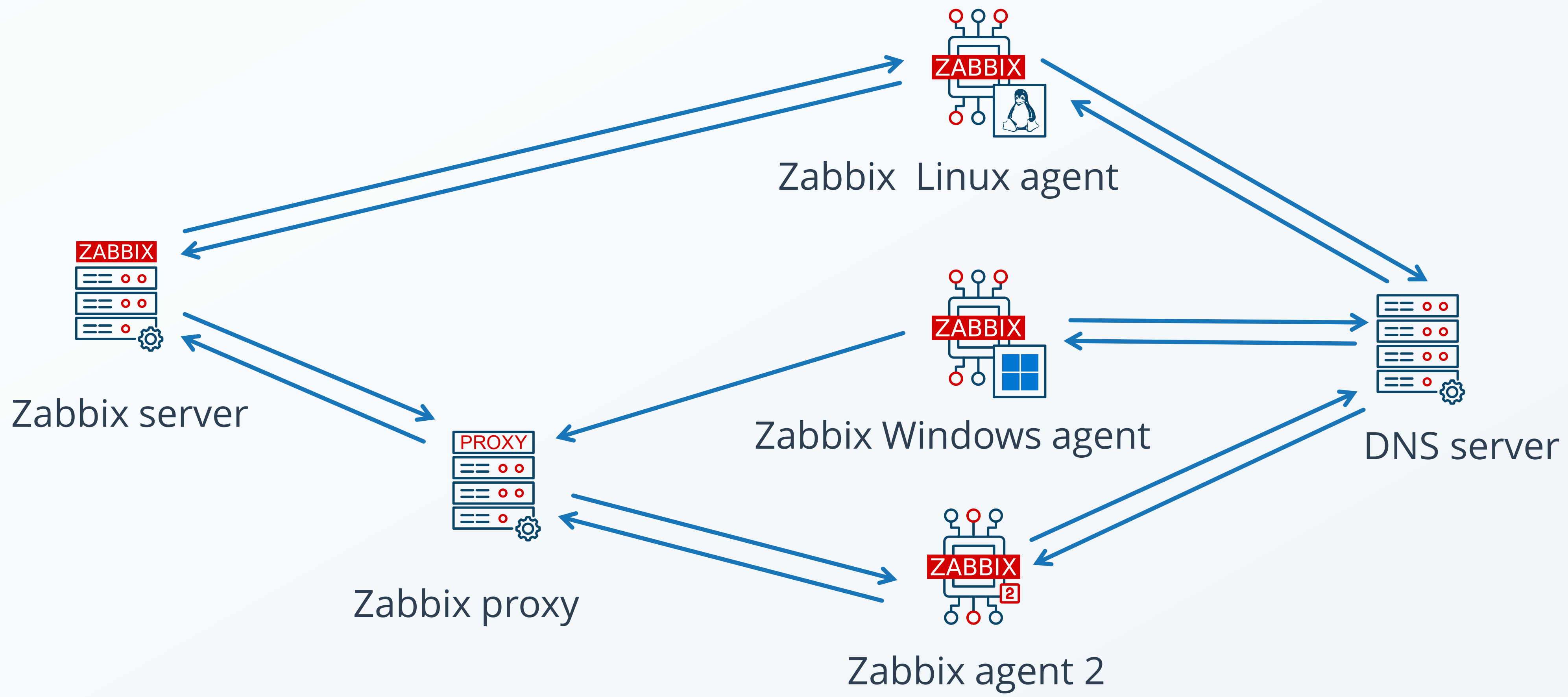
Artjoms Rimdjonoks

C developer

DNS improvements in 7.0

- **Reverse PTR lookups** for all existing and new DNS items
- New item **net.dns.perf**
- New item **net.dns.get**

DNS items are Zabbix agent items



Situation before 7.0

net.dns[<ip>,name,<type>,<timeout>,<count>,<protocol>]

Checks if the DNS service is up.

Return values:

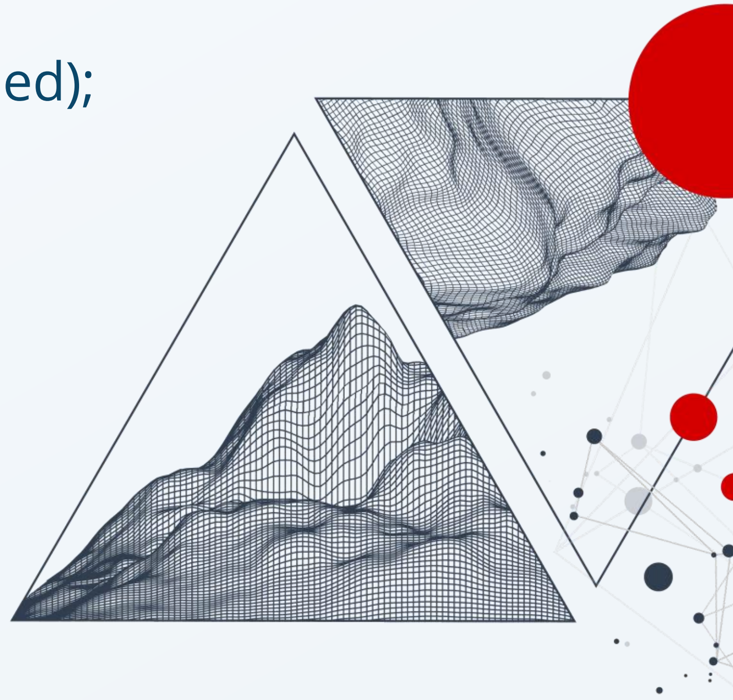
0 - DNS is down (server did not respond or DNS resolution failed);

1 - DNS is up.

net.dns[,example.com,A]

->

1



net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]

Performs a DNS query.

Return value: a character string with the required type of information.

Type – record type to be queried, possible values:

ANY, A, NS, CNAME, MB, MG, MR, PTR, MD, MF, MX, SOA, NULL, HINFO, MINFO, TXT, SRV

WKS (not supported for Zabbix agent on Windows, Zabbix agent 2 on all OS)

net.dns.record[,example.com,A]

->

example.com A 93.184.215.14

Reverse PTR form

Usability improvement when querying PTR records. (ZBXNEXT-3826)

Can be used for existing and new DNS items.

Regular record (any non-PTR) query:

net.dns.record[,example.com,A] -> example.com A 93.184.215.14

PTR record query:

net.dns.record[,107.170.251.121,PTR] –

item becomes unsupported “Cannot perform DNS query.”

We cannot just supply IP to DNS server for a reverse PTR lookup…

1) **IP addresses need to inverted** (because when they are read from left to right - they get more specific)

2) need to **add “.in-addr.arpa”** domain for ipv4, (”**.ip6.arpa**" for ipv6), since reverse lookups are stored in this special domain

Examples:

```
net.dns.record[,121.251.170.107.in-addr.arpa, PTR]

net.dns.record[,c.2.b.8.b.6.f.a.a.d.0.8.0.2.8.6.7.0.b.c.f.1.2.0.0.0.8.2.6.0.6.2.ip6.arpa., PTR]
```

How does the world outside Zabbix handle it?

ZABBIX SUMMIT 2024

dig tool supports -x option to improve usability:

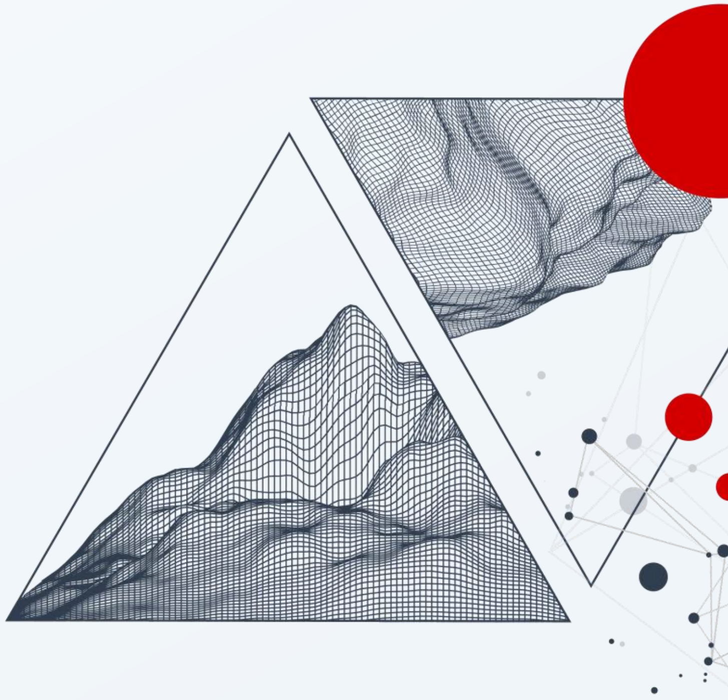
```
dig 121.251.170.107.in-addr.arpa. PTR
dig -x 107.170.251.121


dig c.2.b.8.b.6.f.a.a.d.0.8.0.2.8.6.7.0.b.c.f.1.2.0.0.0.8.2.6.0.6.2.ip6.arpa. PTR
dig -x 2606:2800:21f:cb07:6820:80da:af6b:8b2c
```

In Zabbix 7.0 in addition to the original form, new form can now also be used:

```
net.dns.record[,107.170.251.121, PTR]

net.dns[,2606:2800:21f:cb07:6820:80da:af6b:8b2c, PTR]
```



ZABBIX SUMMIT 2024

net.dns.perf

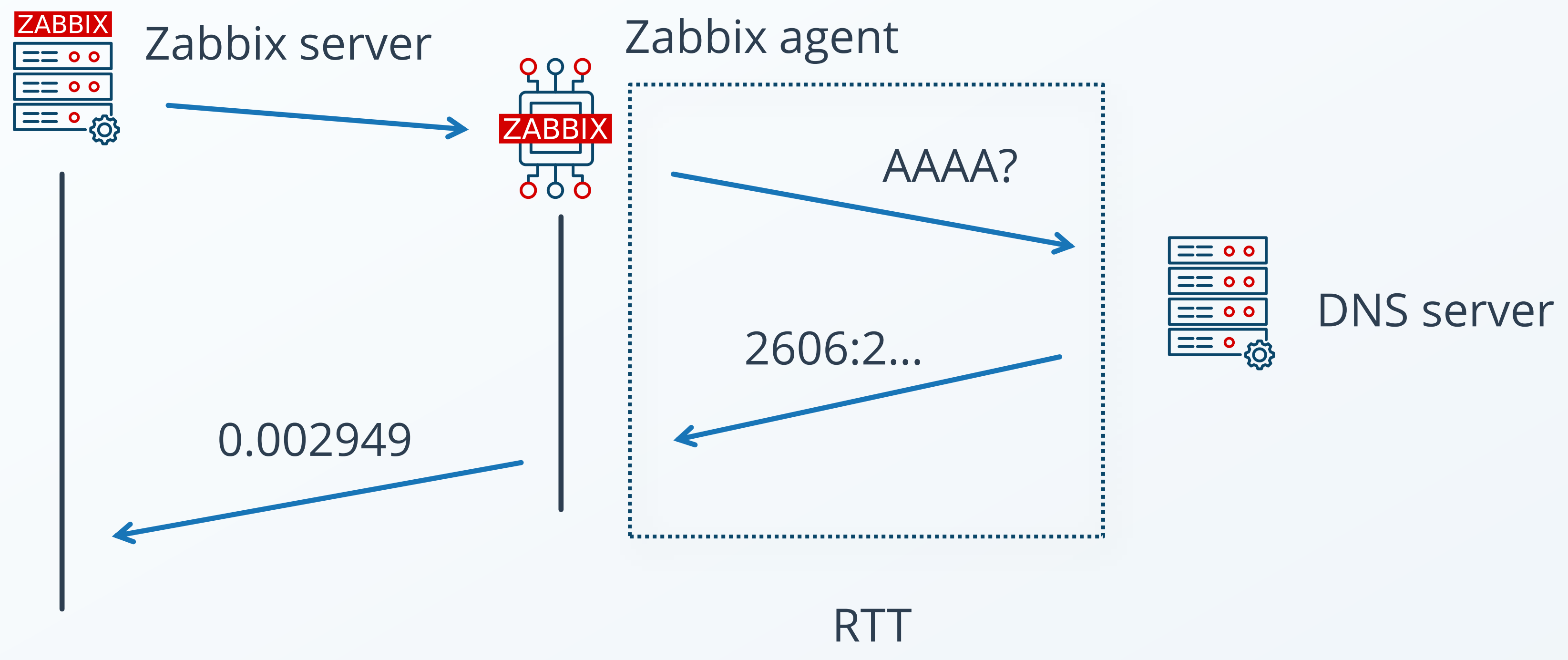
Measures response time for a DNS-query.

Added to Zabbix agent 1 and 2.

ZBXNEXT-5401 (thanks to Robert Young for the patch proposal)

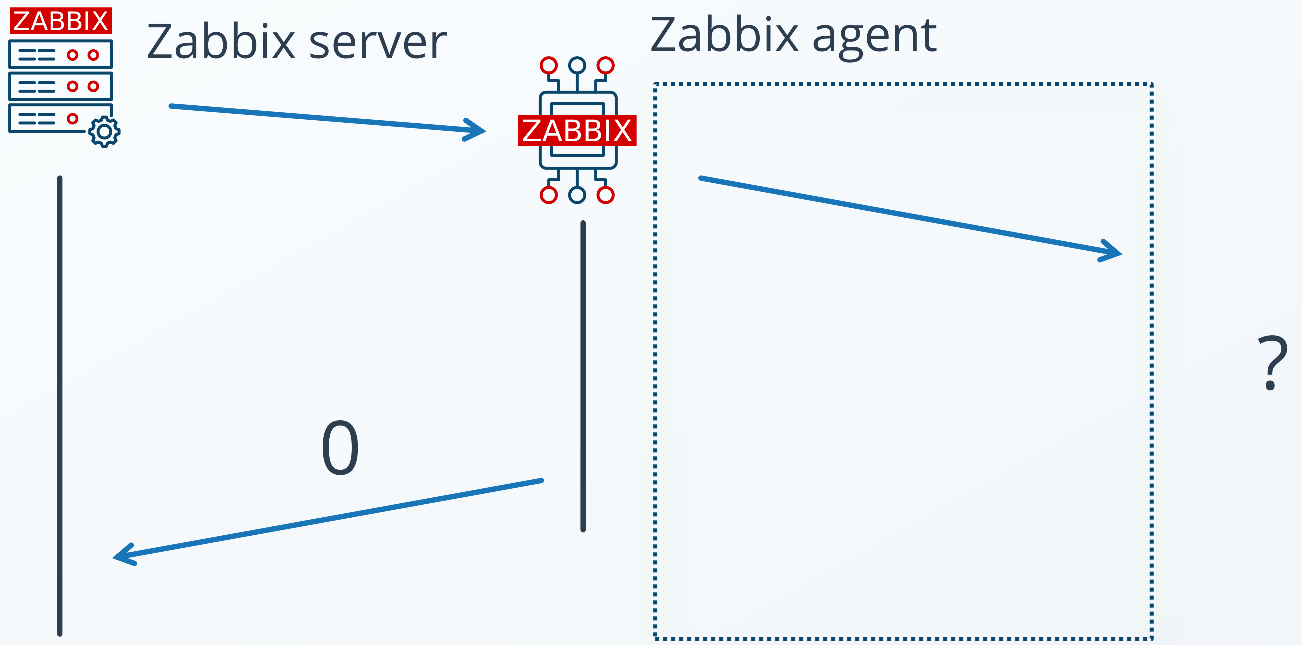
net.dns.perf[<ip>,name,<type>,<timeout>,<count>,<protocol>]

net.dns.perf measures RTT from Zabbix agent to DNS server…

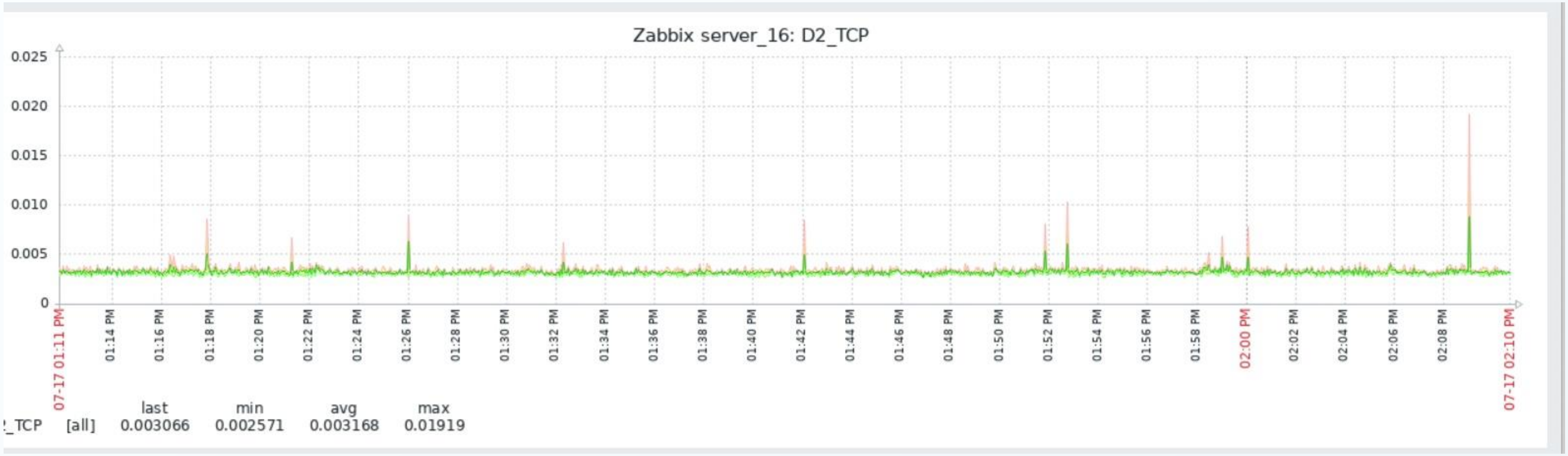


...even if query was not resolved and DNS server returned an error.
(e.g. when NXDOMAIN or SERVFAIL was received)

When connection to the DNS server could not be made (timeout) – it returns **0**.



So, we can have a nice graph:



		last	min	avg	max
_TCP	[all]	0.003066	0.002571	0.003168	0.01919

Windows note

For:

net.dns[<ip>,...

net.dns.record[<ip>,...

net.dns.perf[<ip>,...

- ip is ignored on Windows, when using Zabbix agent 1 (and also timeout and count..)
- Windows DNS C library uses Windows system DNS resolver, before connecting to other remote DNS servers.
- On Windows 0 could be returned only when Zabbix agent 1 fails to connect to the local resolver...

net.dns.get

Before 7.0 for querying record type there was only:

net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]

Return value: a character string with the required type of information.

net.dns.record[,example.com,A]

->

example.com A 93.184.215.14

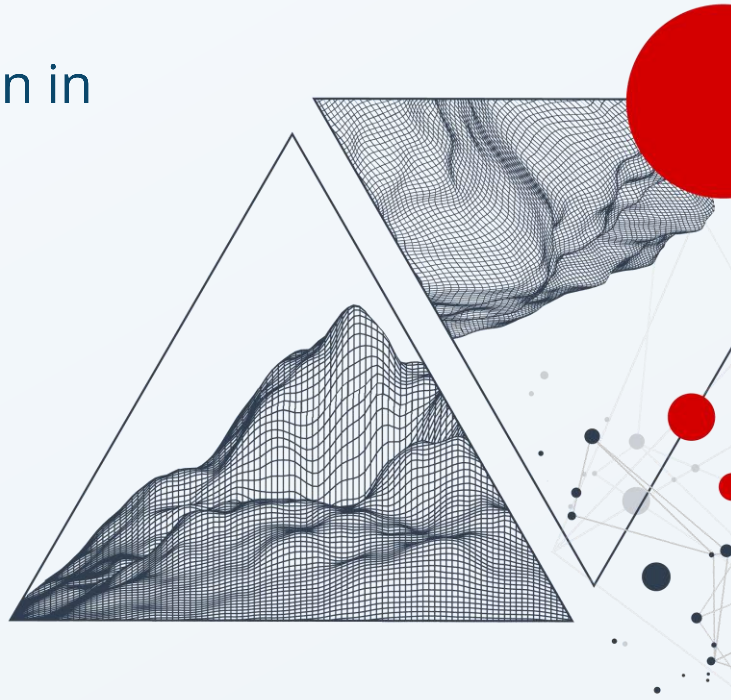
Record types: ANY, A, NS, CNAME, MB, MG, MR, PTR, MD, MF, MX, SOA, NULL, WKS, HINFO, MINFO, TXT, SRV, (not supported for Zabbix agent on Windows, Zabbix agent 2 on all OS),

net.dns.get[<ip>,name,<type>,<timeout>,<count>,<protocol>,"<flags>"]

ZABBIX SUMMIT 2024

- An extended version of the net.dns.record Zabbix agent item with more record types and customizable flags supported..

- Performs a DNS query and returns detailed DNS record information in JSON.



1. Can query more record types !

A, NS, MD, MF, CNAME, SOA,MB, MG, MR, PTR, NULL, HINFO, MINFO, MX, TXT, SRV

+

RP, AFSDB, X25, ISDN, RT, NSAPPTR, SIG, KEY, PX, GPOS, AAAA, LOC, NXT, EID, NIMLOC,, ATMA, NAPTR, KX, CERT, DNAME, OPT, APL, DS, SSHFP, IPSECKEY, RRSIG, NSEC, DNSKEY, DHCID, NSEC3, NSEC3PARAM, TLSA, SMIMEA, HIP, NINFO, RKEY, TALINK, CDS, CDNSKEY, OPENPGPKEY, CSYNC, ZONEMD, SVCB, HTTPS, SPF, UINFO, UID, GID, UNSPEC, NID, L32, L64, LP, EUI48, EUI64, URI, CAA, AVC, AMTRELAY

Note, there are no WKS and ANY record types:

1) ANY is deprecated (RFC8482) ...

2) WKS is not used in practice

Non-capital cases or mixed cases are not allowed in item key arguments.

2. Can pass Flags !

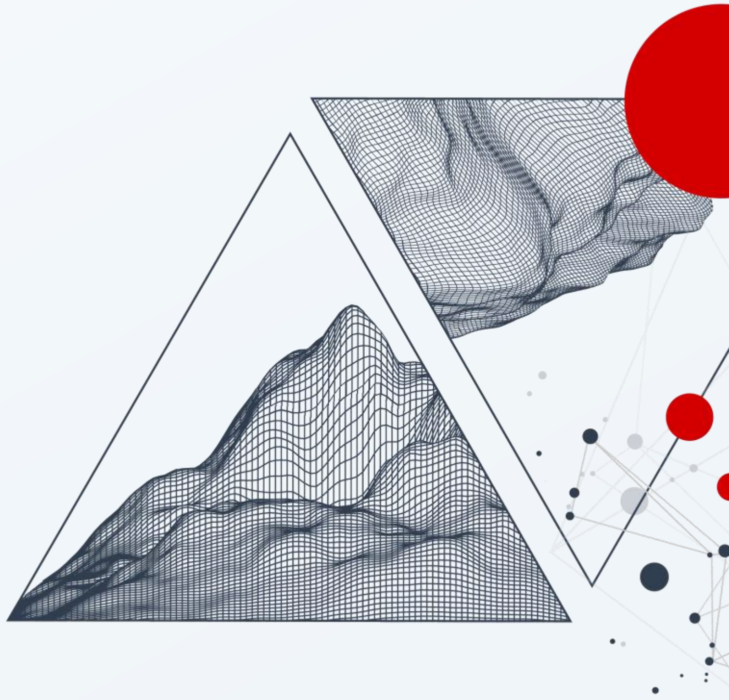
ZABBIX SUMMIT 2024

flag	opposite flag	comment
cdflag	nocdflag(default)	checking disabled (dnssec only)
rdflag(default)	nordflag	recursion desired
dnssec	nodnssec (default)	
nsid	nonsid (default)	
edns0 (default)	noedns0	
aaflag	noaaflag (default)	authoritative answer
adflag	noadflag (default)	authenticated data (dnssec only)

Other tools like dig allow the following when supplying flags:

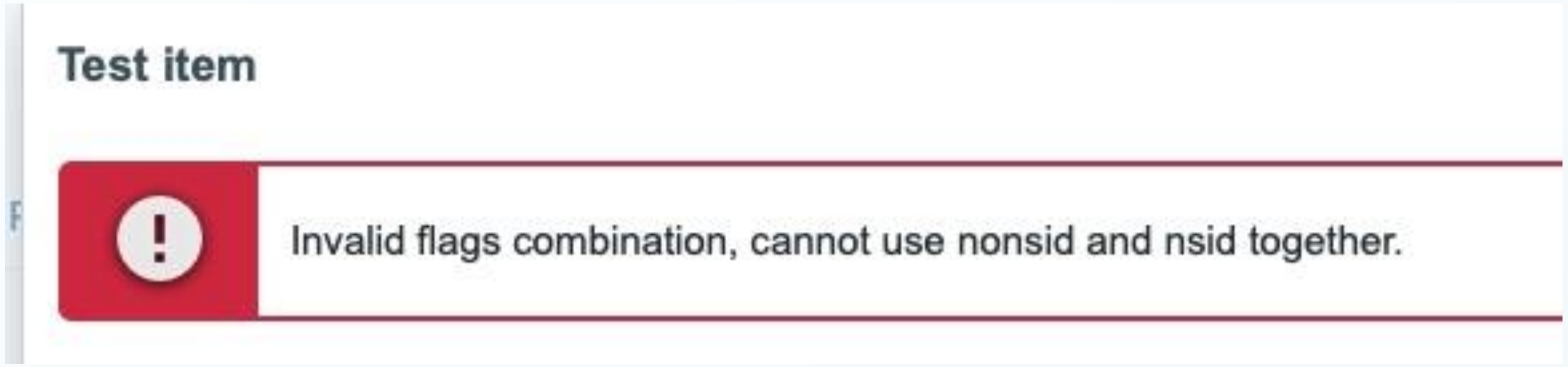
```
dig  zabbix.com +short +noshort +short
104.26.7.148
104.26.6.148
```

```
 dig  +dnssec +nodnssec zabbix.com DS
```



net.dns.get has some checks:

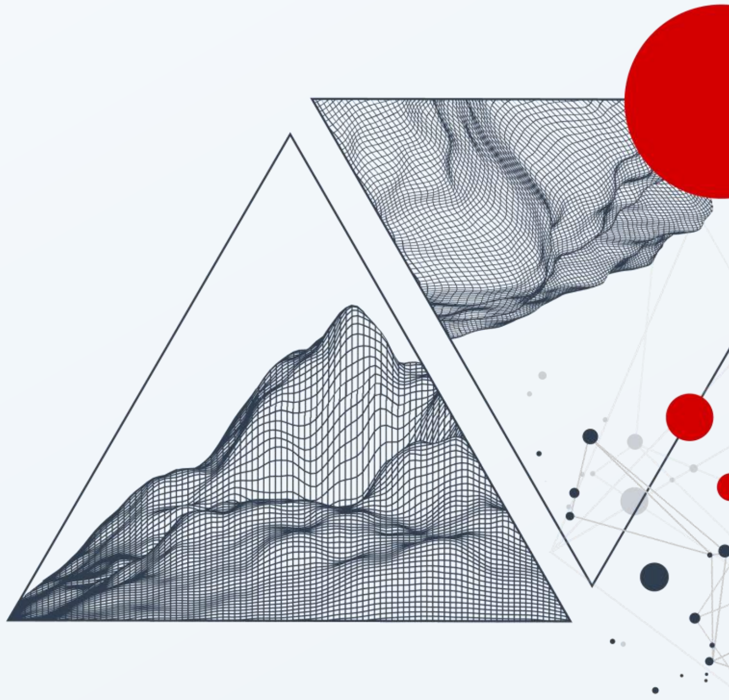
net.dns.get[,zabbix.com,A,,,,"nonsid,nsid"]


Test item

Invalid flags combination, cannot use nonsid and nsid together.

One more example:

net.dns.get[,zabbix.com,A,,,,"noedns0,nsid"]



3. return is a JSON!

net.dns.get[,zabbix.com,A,,,,"nsid"]

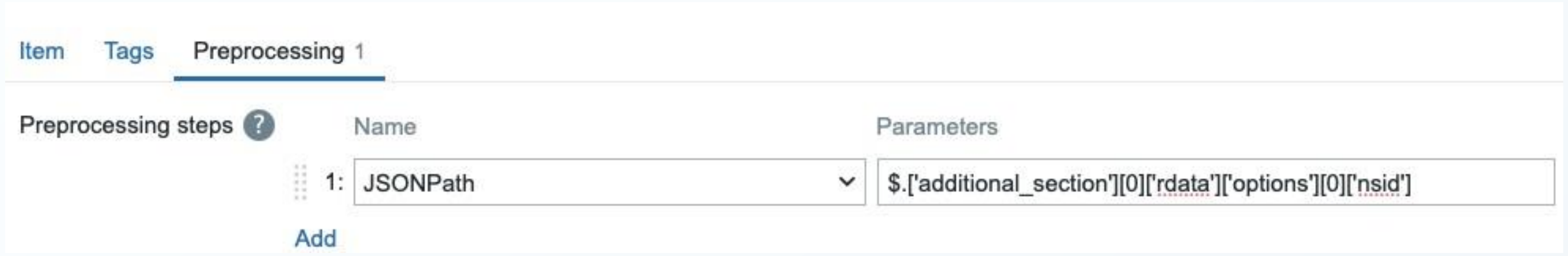
Result:

```
{"additional_section":[{"extended_rcode":0,"name":".","rdata":{"options":[{"code":0,"nsid":"33
38 66 30 62 32 37 66 39 39 34 32 34 38 31 37 39 66 37 39 63 31 35 36 64 38 31 61 36 33 30 32
2e 72 65 73 6f 6c 76 65 64 2e 73 79 73 74 65 6d 64 2e 69
6f"}]},"rdlength":56,"type":"OPT","udp_payload":65494}],"answer_section":[{"class":"IN","name"
:"zabbix.com.","rdata":{"a":"104.26.6.148"},"rdlength":4,"ttl":120,"type":"A"},{"class":"IN","name"
:"zabbix.com.","rdata":{"a":"104.26.7.148"},"rdlength":4,"ttl":120,"type":"A"},{"class":"IN","name"
:"zabbix.com.","rdata":{"a":"172.67.69.4"},"rdlength":4,"ttl":120,"type":"A"}],"flags":["RD","RA"],"
query_time":"0.02","question_section":[{"qclass":"IN","qname":"zabbix.com.","qtype":"A"}],"re
sponse_code":"NOERROR","zbx_error_code":0}
```

ZABBIX SUMMIT 2024

Result can be easily preprocessed with **JSONPath**:

$.['additional_section'][0]['rdata']['options'][0]['nsid’]



Preprocessing steps

	Name	Parameters
1:	JSONPath	$.['additional_section'][0]['rdata']['options'][0]['nsid']

Result:

33 38 66 30 62 32 37 66 39 39 34 32 34 38 31 37 39 66 37 39 63 31 35 36 64 38 31 61 36 33 30 32 2e 72 65 73 6f 6c 76 65 64 2e 73 79 73 74 65 6d 64 2e 69 6f

Return JSON always has:

1) “**query_time**” (in seconds, float type) e.g. "query_time": "0.02"

2) "**zbx_error_code**” (with optional “**zbx_error_msg**” if there is an error)

Scenario	"zbx_error_code"	"zbx_error_msg"
No errors and the DNS response was received and parsed.	0	
DNS is down.	-1	"Communication error"
Error occurs during JSON parsing	-2	"Received unexpected response"

Use-case

Let’s say we want to display:

- nsid
- response_code
- query_time

values in Dashboard widgets…

What do we do ?

Should we create 3 net.dns.get items now ?

Use-case

net.dns.get, like other *.get items that return multiple metric in bulk JSON:

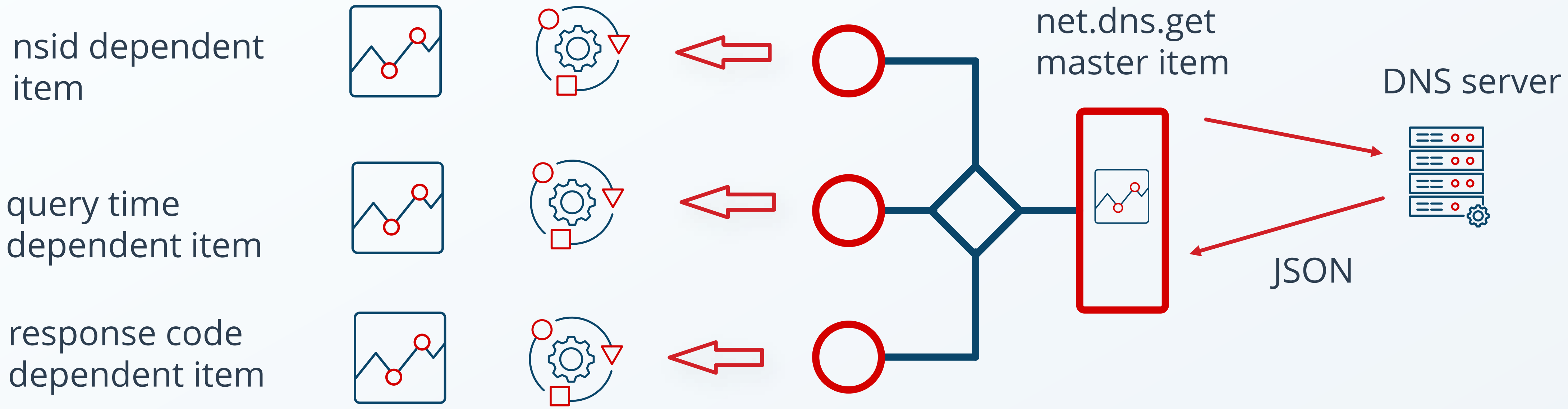
- proc.get
- system.sw.packages.get
- vfs.dir.get
- etc …

(except web.page.get (historical reasons))

are designed to work as **master items**, which provide input data for multiple dependent items for further preprocessing.

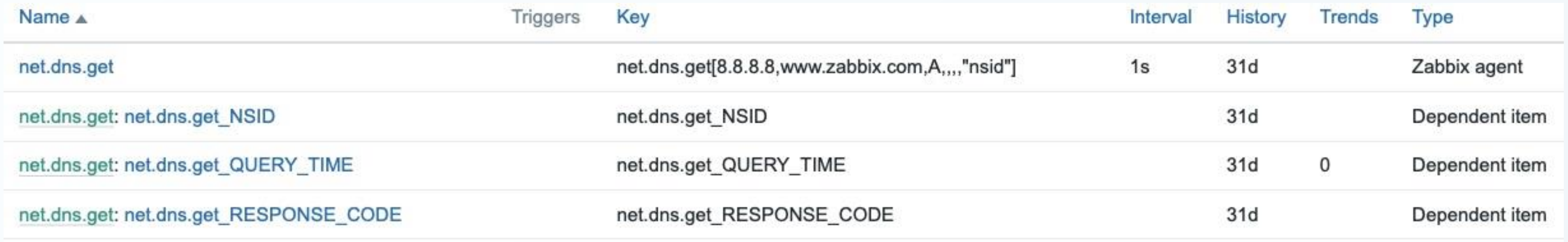
Use-case

- Single, expensive master item extracts bulk metrics.
- Several dependent items extract from master item particular metric they want.
- Any item can be a master item, even the dependent item.
- *.get items return JSON, which allows more advanced/easier preprocessing.



Dependent items

We need 3 new dependent items that will take value from net.dns.get master item.

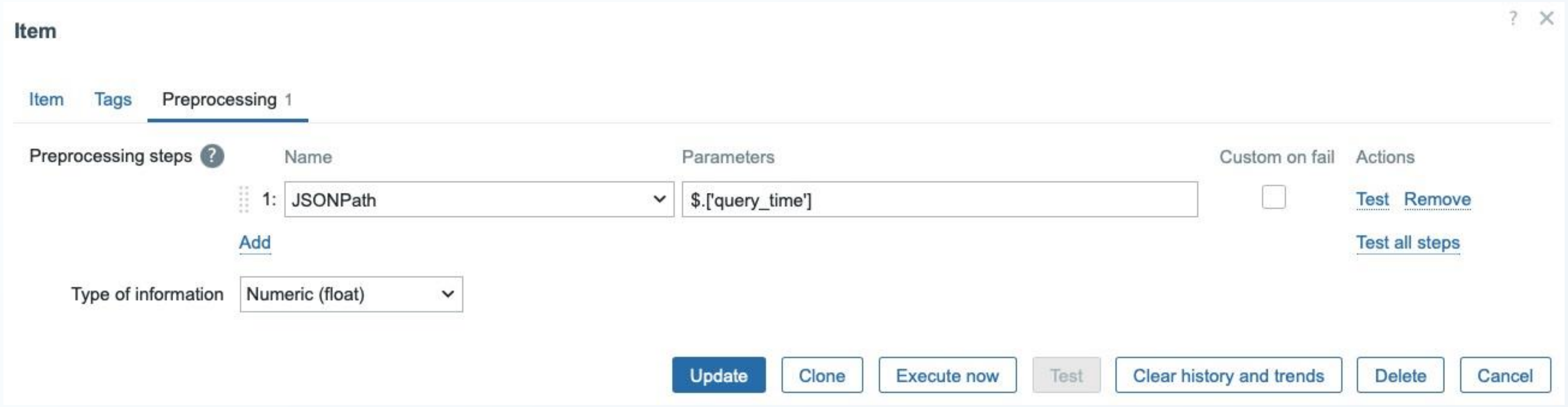


Name ▲	Triggers	Key	Interval	History	Trends	Type
net.dns.get		net.dns.get[8.8.8.8,www.zabbix.com,A,,,,"nsid"]	1s	31d		Zabbix agent
net.dns.get: net.dns.get_NSID		net.dns.get_NSID		31d		Dependent item
net.dns.get: net.dns.get_QUERY_TIME		net.dns.get_QUERY_TIME		31d	0	Dependent item
net.dns.get: net.dns.get_RESPONSE_CODE		net.dns.get_RESPONSE_CODE		31d		Dependent item

Dependent items

query time preprocessing

$.['query_time']



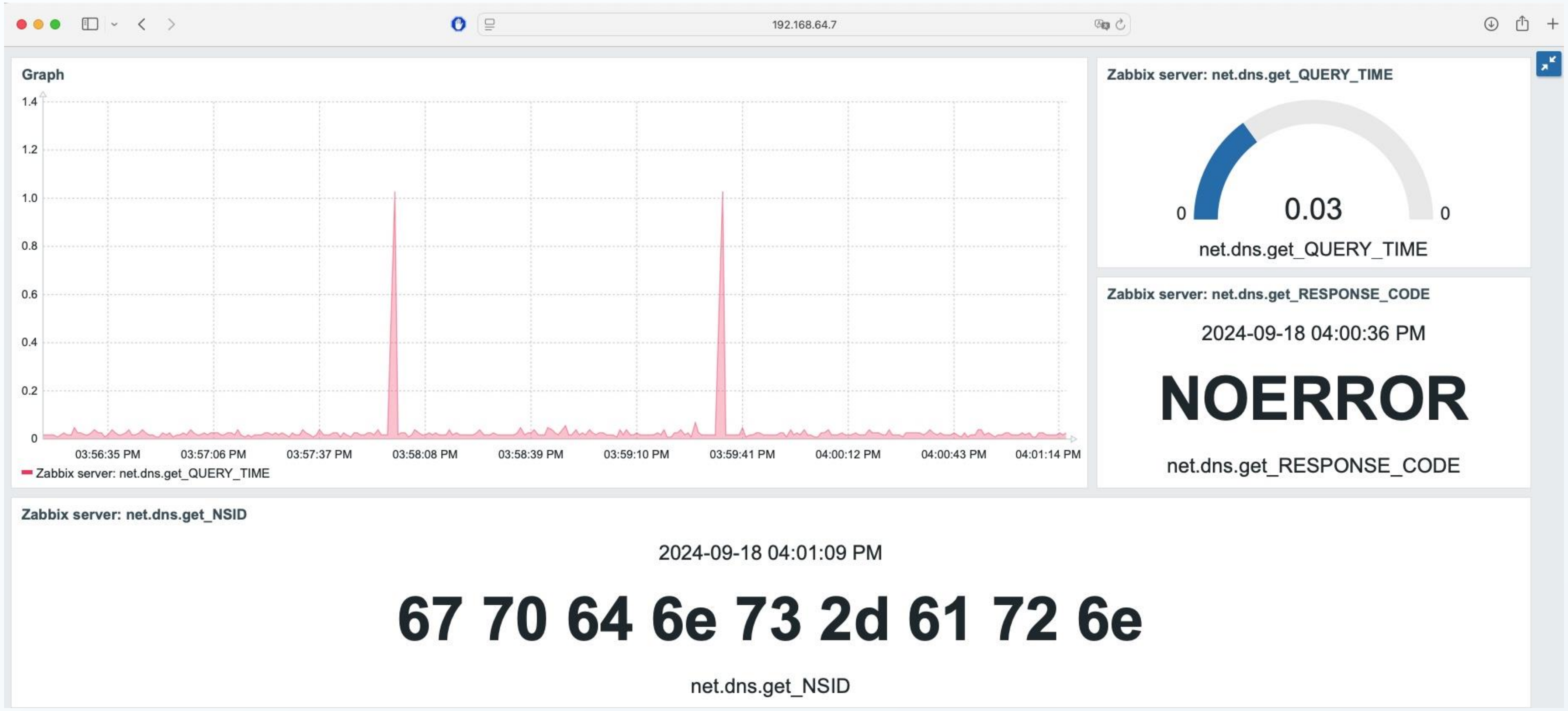
Item

Preprocessing steps

	Name	Parameters	Custom on fail	Actions
1:	JSONPath	$.['query_time']	☐	Test Remove

Type of information: Numeric (float)

Dashboard widgets



Zabbix server: net.dns.get_QUERY_TIME

0 0.03 0

net.dns.get_QUERY_TIME

Zabbix server: net.dns.get_RESPONSE_CODE

2024-09-18 04:00:36 PM

NOERROR

net.dns.get_RESPONSE_CODE

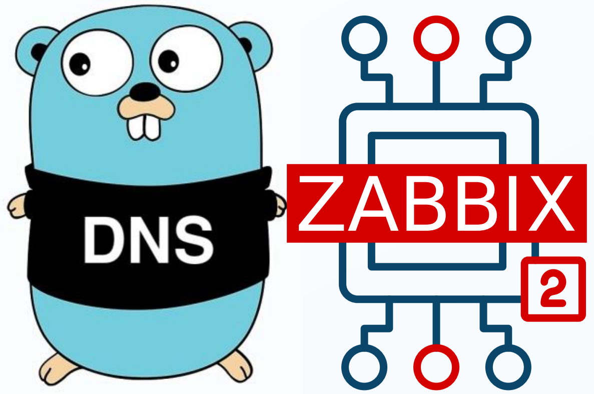
Zabbix server: net.dns.get_NSID

2024-09-18 04:01:09 PM

67 70 64 6e 73 2d 61 72 6e

net.dns.get_NSID

Available only for Zabbix agent 2



… since there is different DNS C library for every OS …

And in Go – we could use just the one (https://github.com/miekg/dns)

Artjoms Rimdjonoks

C developer