

IBM i Monitoring in 2026

Experience of monitoring IBM iSeries systems with Zabbix



Agenda



Agenda

Introduction

A few words about me, about our bank

Some words about my project

Its necessity, its history, main idea and implementation, state at September 2017

New story

Project's history since September 2017

Introduction



About me



Constantin Oshmyan

Konstantīns Ošmjans, Константин ОШМЯН

COshmyan@gmail.com

My personal e-mail

Senior analyst-programmer

In Rietumu Bank since 2003

About Rietumu Bank

Leading private bank in the Baltic states

**Over 30 years of successful operating
experience**

www.rietumu.com

Some words about my project

My speech at Zabbix Conference - 2017:

<https://www.youtube.com/watch?v=wxXamKDNeJs>



History

Operating systems

Windows, Unix (AIX and Solaris), Linux, OS/400

Monitoring system

IBM Tivoli Distributed Monitoring v3.6 -> 3.7 -> 4.x
-> (Omegamon 5.x) -> IBM Tivoli Monitoring 6.x

Zabbix era

Zabbix v2.0 -> 2.2 -> 3.0 -> 4.0 -> 5.0 -> 6.0 -
> 7.0

Few words about AS/400

Platform: IBM Power Systems

formerly known as: System i, eServer iSeries, AS/400

Operating system: IBM i

formerly known as: i5/OS, OS/400

Very specific

architecture, API, filesystem, etc.

There was no agent for Zabbix

We should monitor some things specific for this platform:

ASP used, disk unit state, jobs, subsystems, spooled files queues, message queues

Standard agent for this platform?

Does not exist, POSIX only.

POSIX subsystem?

PASE (Portable Application Solutions Environment), but limited.

SNMP?

Very limited and unreliable, no AS/400-specific information.

The IBM® Toolbox for Java

The IBM® Toolbox for Java: access classes represent OS/400 data and resources, including:

- [Job](#)
- [Subsystem](#)
- [QueuedMessage](#)
- [MessageQueue](#)

Open source version (JTOpen):

<https://javadoc.io/doc/net.sf.jt400/jt400/latest/index.html>

Idea of solution

JTOpen

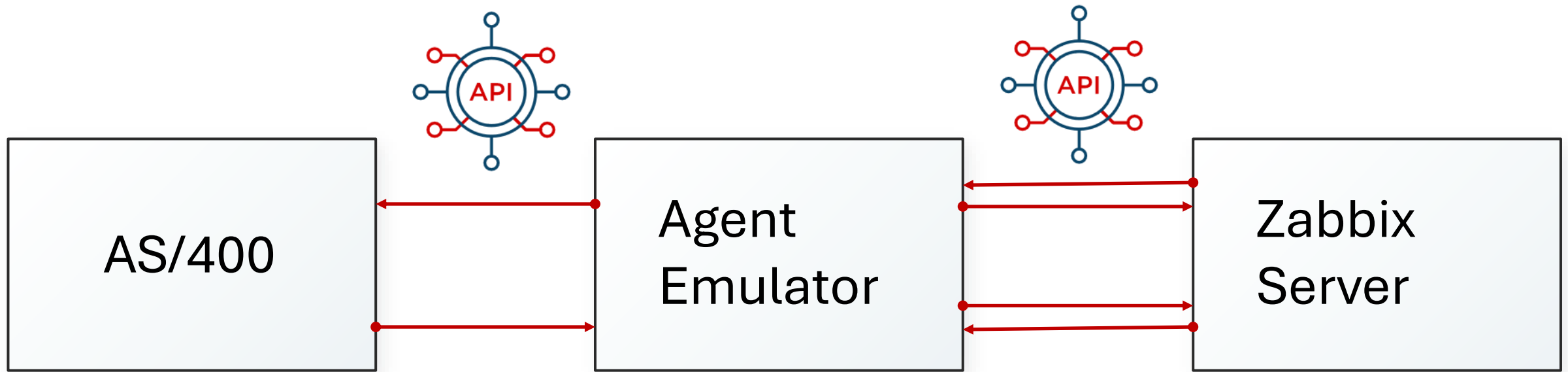
There is an API to AS/400 that allows to access the needed objects and attributes

Zabbix Agent/Server protocol

There is a documented protocol describing the interaction between Zabbix-server and Zabbix-agent

Example of Zabbix protocol

There is a "reference implementation" of this protocol: open sources of Zabbix agent



IBM Toolbox
for Java

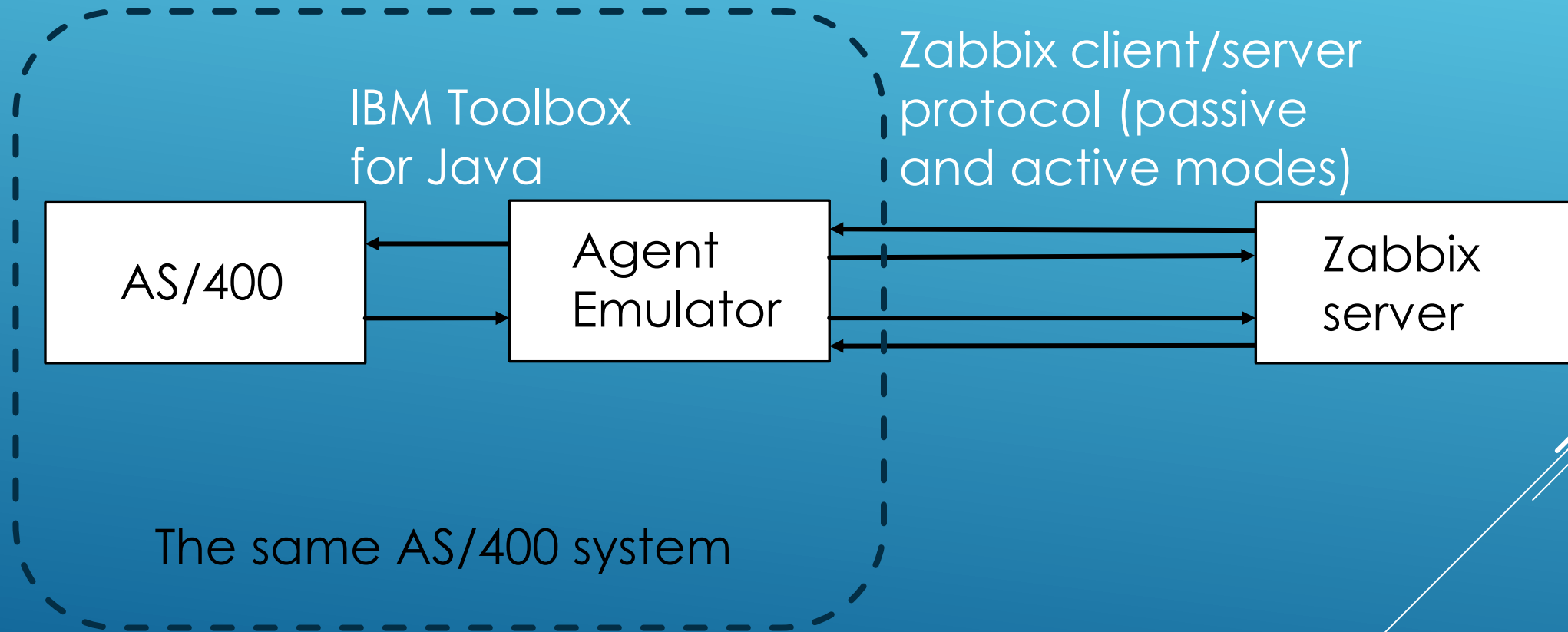
Zabbix client/server
protocol (passive
and active modes)

Solution

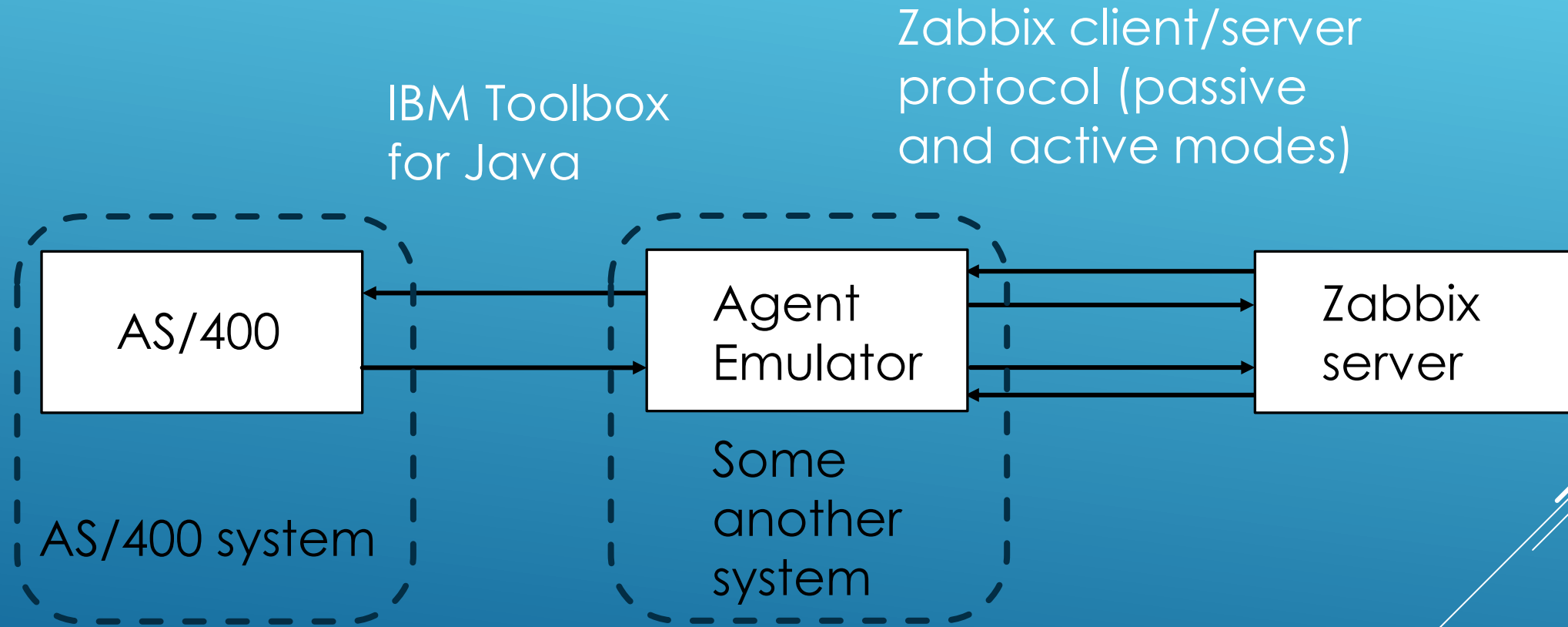
Several slides from my previous presentation

(Zabbix Summit 2017)

LOCAL MODE



REMOTE MODE



IMPLEMENTATION

- ▶ Communications to Zabbix-server: protocol description + sources of Zabbix-agent (just repeat the same logic)
- ▶ Using Java-specific features:
 - Threads
 - Exceptions (instead of return codes)
 - Standard data structures (**ArrayList**, **Hashtable**, **StringBuilder**, etc.)

IMPLEMENTATION

(continued)

► Communications to AS/400:

- Using the JTOpen classes where if possible (jobs, subsystems, queues)
- Otherwise: using JTOpen interface to native AS/400 API (ASP's, disk units)
- Collecting of CPU Usage statistics (by jobs) using a special "collector" thread
- Limitations for obtaining information about OS using direct Java API: data should arrive exactly from AS/400 system

IMPLEMENTATION

(continued)

- ▶ Minimum dependencies: **jt400.jar** and **json-simple-1.1.1.jar** (23.5 KB) only
- ▶ Parsing of metric key and calls of the appropriate processing method: code is shared between all threads (both for active and passive modes)

LIMITATIONS AND DISTINCTIONS FROM THE STANDARD AGENT

- ▶ Minimum JRE version is 1.7
- ▶ Lack of IPv6 (IPv4 only supported)
- ▶ No encryption (sorry, no plans)
- ▶ Some config file's parameters are recognized, but really ignored (PidFile, EnableRemoteCommand, Alias, AllowRoot, Include, UserParameter and LoadModule)

LIMITATIONS AND DISTINCTIONS FROM THE STANDARD AGENT

(continued)

- ▶ Configuration parameter **ListenIP** allows to set only one IP-address (contrary to list in original Zabbix-agent)
- ▶ Minimum value for the **StartAgents** parameter is 1 (i.e. active-only mode is not supported)
- ▶ Only limited subset of standard metrics supported, and even in this case: some of them have a bit different semantics of parameters (see **proc.num[]** or **eventlog[]** for examples)

EXAMPLES

How does it look?

Latest data



Filter ▲

Host groups	type here to search	Select	Name	
Hosts	as400-test x type here to search	Select	Show items without data	☒
Application		Select	Show details	☒

Filter

Reset

☐ Name ▲	Interval	History	Trends	Type	Last check	Last value	Change	Info
▶ ASP (12 Items)								
▶ CPU Usage by process (17 Items)								
▶ Disks (72 Items)								
▶ Generic (9 Items)								
▶ Message Queue (5 Items)								
▶ Output Queue (7 Items)								
▼ Zabbix agent (3 Items)								
☐ Agent ping agent.ping	60	7	365	Zabbix agent	2017-09-13 18:34:48	Up (1)		Graph
☐ Host name of zabbix_agentd running agent.hostname	3600	7		Zabbix agent	2017-09-13 18:25:49	as400-test		History
☐ Version of zabbix_agent(d) running agent.version	3600	7		Zabbix agent	2017-09-13 18:25:50	0.7.3		History

0 selected

Display stacked graph

Display graph

Generic (10 Items)									
☐	CPU Usage for all process proc.cpu.util[,total,]	60	7	365	Zabbix agent (...)	2017-09-13 19:04:54	0.35 %		Graph
☐	Number of jobs in state LCKW proc.num[,LCKW,]	300	7	365	Zabbix agent (...)	2017-09-13 19:03:39	0		Graph
☐	Number of jobs in state run with global RE (active) proc.num[,run,@test_as400]	300	7	365	Zabbix agent (...)	2017-09-13 19:03:43	0		Graph
☐	Number of jobs with global RE (passive) proc.num[,,@test_as400]	300	7	365	Zabbix agent	2017-09-13 19:00:53	0		Graph
☐	Number of running jobs (total) proc.num[,run]	300	7	365	Zabbix agent (...)	2017-09-13 19:03:44	216	+2	Graph
☐	Number of users system.users.num	60	7	365	Zabbix agent (...)	2017-09-13 19:03:58	0		Graph
☐	Services availability as400.services	300	7	365	Zabbix agent (...)	2017-09-13 19:03:42	0		Graph
☐	Status of subsystem QTRMEQIF in library QGPL as400.subsystem[QTRMEQIF,QGPL]	300	7		Zabbix agent (...)	2017-09-13 19:03:38	*INACTIVE		History
☐	Status of subsystem qtrmeqpl5 in library qgpl as400.subsystem[qtrmeqpl5,qgpl]	300	7		Zabbix agent (...)				History
☐	System information system.uname	120	7		Zabbix agent	2017-09-13 19:03:51	IBM OS/4		

There is no subsystem with name='qtrmeqpl5' in library 'qgpl'

```

▶ Message Queue (5 Items)
10:20170913:191428.847 Starting Zabbix Agent v0.7.3
10:20170913:191428.851 using configuration file: zabbix_agentd.conf
10:20170913:191428.852 Oracle Corporation Java version "1.8.0_20"
10:20170913:191428.852 Java(TM) SE Runtime Environment (build 1.8.0_20-b26)
10:20170913:191428.852 Java HotSpot(TM) 64-Bit Server VM (build 25.20-b23, mixed mode)
10:20170913:191428.853 Open Source Software, JOpen 9.1, codebase 5770-SS1 V7R3M0.00 built=20160705 @RF
10:20170913:191429.012 Agent hostname: 'as400-test', System info: IBM OS/400 V7R1M0, Oracle Corporation Java HotSpot(TM) 64-Bit Server VM (v1.8.0_20)
14:20170913:191429.033 agent #1 started [collector]
16:20170913:191429.037 agent #3 started [listener #1]
17:20170913:191429.038 agent #4 started [listener #2]
15:20170913:191429.038 agent #2 (zabbix.rietumu.lv) started [active checks #3]
15:20170913:191430.580 As400Metric.process() error: com.ibm.as400.access.AS400Exception: CPF9801 Object QPRINTPL5 in library QGPL not found.

```

COMMUNITY AND TECHNICAL SUPPORT

- ▶ License: BSD-like, open source. You can do what you want respecting my copyrights.
- ▶ Distribution: an archive containing both binary, sources, config file example and documentation could be downloaded from <https://share.zabbix.com/operating-systems/ibm-i-i5-os-os-400-for-ibm-system-i-as-400/zabbix-agent-emulator-for-as-400-platform>

COMMUNITY AND TECHNICAL SUPPORT

(continued)

- ▶ An example of the template can be downloaded from the same site: <https://share.zabbix.com/operating-systems/ibm-i-i5-os-os-400-for-ibm-system-i-as-400/template-example-for-as-400-agent-emulator>
- ▶ Discussion on the Zabbix forum: <https://www.zabbix.com/forum/showthread.php?t=47525>
- ▶ Thanks to the Zabbix support team!

State as of September 2017

share.zabbix.com:

Homepage of this project:

[sources/binaries/documentation](#)

[an example of template](#)

Information about this project:

[Zabbix forum](#)

[location on Zabbix Share site](#)

[presentation at Zabbix Conference 2017 \(and video on YouTube\)](#)

Communications with community

No promises

I did not promise to do anything, but it was interesting for me to understand if this project is useful for anybody else.

Feedback by all available channels

- Zabbix forum (both public discussion and private messages)
- email

Types of feedback

- Thanks
- Questions
- Enhancement requests
- Bug reports

New story



New story: Project history

September 2017 - December 2025

v0.7.4 – 0.7.9: "feedback-driven development"



Main drivers of development

Community

Feedback

Changes in Zabbix

Protocol changes

New features

Usually, new metrics:

- `as400.cpu.capacity`
- `as400.systemPool.state`
- `as400.systemPool.discovery`

Small new features/enhancements

- Monitoring of History Log (similarly to Message Queue)
- Additional configurational parameter "as400JobAsMessagePrefix" (similar to "as400EventIdAsMessagePrefix" and "as400UserAsMessagePrefix")
- Limitation on number of ASP's and disk units was cancelled
- Troubleshooting/diagnostics: "as400.services" checks in a separate threads, "agent.debug.active.thread" metric

Bug fixes

A lot of small bugs, it's normal :-) and optimizations

Changes

- Zabbix protocol changes (initially in Zabbix v4.0.0: binary headers for a passive mode, "id" field for an active mode)
- Summer's PTF 2022 on 'system i' v7.4 causes to wrong processing of all "vfs.fs.*" and "as400.disk.*" metrics - the correct info returned for the first element (ASP or disk) only

Summary

- Some modifications and enhancements, but no great enhancements in functionality
- Zabbix-related code was based on Zabbix sources v3.0.x
- The same main limitations:
 - No IPv6
 - No encryption
 - No “aliases” support in config.file
 - No “system.run” metric (no “UserParameter” support, no remote command execution)
- Frequently asked enhancement requests:
 - SQL interface usage
 - “system.run” metric

Recent history: this year's releases

Serious review and rework



Release 2026: v0.7.10

Current version of protocol

It pretends to be a real Zabbix Agent v7.0.0 (agent variant 1).

Configuration parameters

All configuration parameters were reviewed.

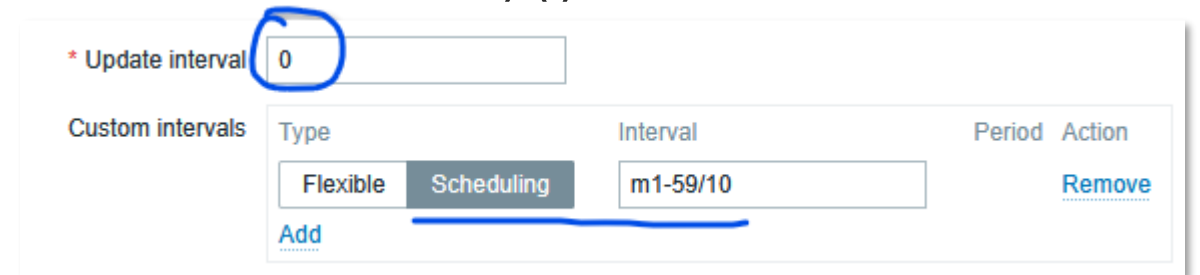
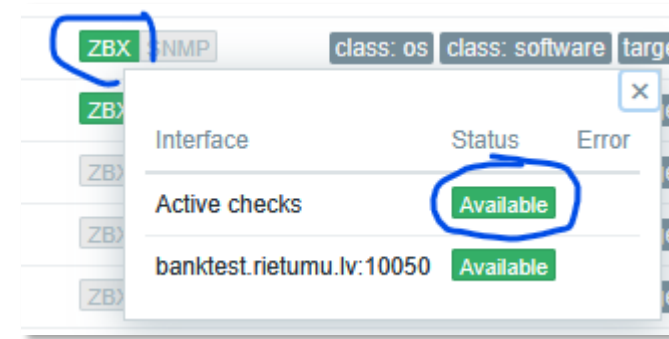
Metrics

Some new metrics were added or slightly modified.

Protocol v7.0.0

Agent v0.7.10 pretends to be a real Zabbix Agent v7.0.0 for Zabbix server/proxy (agent variant 1). Therefore, it supports such things as:

- Heartbeats (added in Zabbix v6.2) -> correct availability icon for active checks
- Incremental configuration update (added in Zabbix 6.4)
- Zabbix server clustering (added in Zabbix 6.0) and Zabbix proxy clustering (added in Zabbix 7.0), including redirection messages
- Custom intervals for active mode (both flexible and scheduled intervals) (!)
- Passive mode: support of new format of requests (in JSON form); at the same time compatibility with the old format, too.



Configuration file

All configuration parameters were reviewed. Now they can be processed in one of the following ways:

- Unknown parameters: error message and failure to start (it is the same behavior as before)
- Recognized and implemented: should be really used
- Recognized, but not implemented parameters (like "TLS*" or "LogType"): ignore, but write a warning to the log file

No more silently ignoring!

Configuration file

The following parameters now are really used or have been modified:

- **AllowKey/DenyKey** (with "our" metrics "agent.exit" and "agent.debug.active.thread" - denied by default)
- **Server** (is mandatory now, but currently it supports CIDR subnets)
- **ServerActive** (can contain clustered configurations)
- **HostMetadata** and **HostMetadataItem**: increased allowed length (from 255 chars to 2034 bytes and 65535 chars accordingly)
- **Alias** (was not used before)
- **ListenBacklog** (was not used before)
- **SourceIP** (was not used before)
- **HostInterface/HostInterfaceItem**, **ListenPort** and **ListenIP** parameters are really transferred (if they are) during autoregistration of active agent

Metrics modifications

All `"*.discovery"` metrics now return just arrays, without the `"data"` element (it was removed in Zabbix v4.2, so I don't think it is used now).

For example, `vfs.fs.discovery`:

Before:

```
{"data": [  
  {"{#FSNAME}": "1", "{#FSTYPE}": "00"}  
]}
```

Now:

```
[  
  {"{#FSNAME}": "1", "{#FSTYPE}": "00"}  
]
```

New metrics

New metrics were added:

- agent.hostmetadata
- agent.variant
- vfs.fs.get
- as400.disk.get
- as400.systemPool.get

Release 2026: v0.8.0

Per-item timeouts

Uses new protocol implementation.

Troubleshooting/Diagnostics

Additional debugging info in "agent.debug.active.thread" metric.

New metrics

SQL requests to AS/400 without unixODBC, "system.run"!

Other features available with "system.run"

UserParameter, remote commands.

Per-item timeouts

Based on the new protocol addition (both for active and passive modes): each metric can be accompanied with its own timeout.

- If timeout is not provided by server/proxy (version older than 7.0.x), then use the timeout from config. file as a fallback
- Each check is performed in its own child thread, but the parent thread waits for timeout
- **as400.services** metric: timeout is a total timeout for all checks; the timeout for each service check is the appropriate part of this total timeout
- The correct processing of timeouts: it's important to release resources, otherwise – memory leak (fast or slow)

Troubleshooting/Diagnostics

Additional debugging info in "agent.debug.active.thread" metric.

- This metric is disabled by default, use "AllowKey=agent.debug.active.thread" in the agent's config file to enable it
- Now contains info about memory used by JVM (total/free/max)
- Counter of timeout events

Troubleshooting/Diagnostics (during memory leak)



Troubleshooting/Diagnostics (the same host now)

Name ▲	Interval	History	Trends	Type	Last check	Last value	Change	Tags
Debug: ActiveCount debug.ActiveCount		31d	365d	Dependent...	1s	19		test: debug
Debug: Memory (free) debug.memory[free]		31d	365d	Dependent...	1s	88.88 MB	-15.54 MB	test: debug
Debug: Memory (max) debug.memory[max]		31d	365d	Dependent...	1s	2 GB		test: debug
Debug: Memory (total) debug.memory[total]		31d	365d	Dependent...	1s	172.19 MB		test: debug
Debug: Timeout count debug.timeoutCount		31d	365d	Dependent...	1s	16		test: debug

SQL interface to AS/400

One of the most requested features.

- AS/400 SQL interface (official name: IBM i Services):
 - <https://www.ibm.com/support/pages/ibm-i-services>
- More information about it for different versions of OS:
 - <https://www.ibm.com/docs/en/i/7.2.0?topic=optimization-i-services>
 - <https://www.ibm.com/docs/en/i/7.3.0?topic=optimization-i-services>
 - <https://www.ibm.com/docs/en/i/7.4.0?topic=optimization-i-services>
 - <https://www.ibm.com/docs/en/i/7.5.0?topic=optimization-i-services>
 - <https://www.ibm.com/docs/en/i/7.6.0?topic=optimization-i-services>
- Information about the supported IBM® i releases, including the database PTF groups where each service was introduced or enhanced, is available in the IBM i Technology Updates wiki. Details:
 - <https://www.ibm.com/support/pages/node/1119123>

SQL interface to AS/400

- AS/400 IBM Toolbox for Java includes the JDBC driver
- Allows the use of SQL requests without unixODBC middleware (we use JDBC instead)
- Metrics `"as400.sql.select"` and `"as400.sql.get"` – analogs of `"db.odbc.select"` and `"db.odbc.get"` (single value and result set)
- Uses the same credentials that Zabbix agent uses
- Uses read-only connections for security reasons

SQL interface to AS/400: examples

Total count of jobs in the system:

```
as400.sql.select["SELECT COUNT(*) FROM TABLE(QSYS2.ACTIVE_JOB_INFO(JOB_NAME_FILTER => '*ALL', DETAILED_INFO => 'ALL')) X"]
```

Result:

413

```
as400.sql.get["SELECT COUNT(*) AS COUNT FROM TABLE(QSYS2.ACTIVE_JOB_INFO(JOB_NAME_FILTER => '*ALL', DETAILED_INFO => 'ALL')) X"]
```

Result:

```
[  
{  
  "COUNT": "413"  
}]
```

SQL interface to AS/400: examples

```
as400.sql.get["SELECT COUNT(*) FROM TABLE(QSYS2.ACTIVE_JOB_INFO(JOB_NAME_FILTER => '*ALL',  
DETAILED_INFO => 'ALL')) X"]
```

Result:

```
[  
{  
  "00001": "413"  
}]
```

The "system.run" metric

Problem: AS/400 does not have the concepts of "standard output" and "standard errors" (stdout/stderr). Its commands use either screen interface or some specific files for results (spool files, user area files, etc.). It is the reason why we did not implement this feature earlier.

Solution (borrowed from the IBM company): use Qshell (QSH).

- QSH is part of PASE (POSIX subsystem), but can run also some set of standard UNIX utilities (sed, grep, awk, head, tail, ps, find, ls, xargs, etc.)
- It supports concepts of streams (stdin/stdout/stderr and pipes)
- It can also run AS/400 commands (via "system" command) and automatically transforms their results into stream available as stdout or stderr
- Moreover, it allows us to mix AS/400 commands and UNIX commands in pipes, for example:

#Output top-5 jobs sorted by total CPU usage

```
system -q 'WRKACTJOB SEQ(*CPU)' | tail -n+11 | head -6
```

The "system.run" metric

Implementation:

- Form the correct command line for QSH command from the first parameter of "system.run", redirecting the stderr and stdout to a temporary file
- Run QSH with this command line as an argument
- Read the content of the temporary file as the result, then delete this file
- Ignore the second parameter of "system.run" ("wait"/"nowait") – we always process it as "wait"

Example:

```
system.run["system -q 'WRKACTJOB SEQ(*CPU)'|tail -n+11|head -6"]
```

Other features after "system.run"

After "system.run[...]" metric implementation it was possible to implement some useful features:

- The "UserParameter=" option in the configuration file (including parameters substitution):

```
UserParameter=test3[*],system -q 'WRKACTJOB SEQ(*CPU)' | tail -n+$1 | head -$2
```

- Remote command execution (including multi-line scripts)
- Remote command execution in passive mode was implemented automatically (as it is directly called via "system.run")
- Remote command execution in active-only mode required some additional development: protocol usage for both receiving requests for remote commands and sending their results (including failures), logic in active-mode thread; but all these were implemented, too.

Summary

- Full support of Zabbix agent<->server protocol v7.0 (both active and passive modes)
- Zabbix-related code was based on Zabbix sources v7.0.x
- Main limitations
 - No IPv6
 - No encryption
 - ~~• no “aliases” support in config file;~~
 - ~~• no “system.run” metric (no “UserParameter” support, no remote command execution);~~
- Frequently asked enhancement requests:
 - SQL interface usage - implemented
 - “system.run” metric - implemented

Thanks!
Questions?