# Migrating HP OVO to Zabbix: our experience

QUADRATA
SERVICE GROUP S.R.L.

Quadrata is an Italian group of passionate IT professionals specialized in providing consulting services for medium and big enterprises.

Our solutions are designed for company process-flow optimizations and problem solving, thus improving productivity and reducing costs for our customers

With a decade of experience and thanks to the team members passion, Quadrata represent the best partner of your company for the IT management

We believe technology should not be the limit in a constantly evolving world, but the center piece of new and great opportunities.

# Agenda

What is HP OVO

OVO Main features

Migrating from OVO to Zabbix: our experience

Lesson learned

Zabbix and OVO: a better world

HP OpenView

- HP OpenView is the former name for a Hewlett-Packard product family for networks / systems monitoring and management
- Network Node Manager was the first product, released in the mid 1980s
- Not long after, HP sold the discovery engine to IBM for Tivoli.
- Operations Center came out in late 1980 as an "add-on" product to NNM to provide server and application management, and provided communication via RPC instead of just SNMP
- Around 1995, Operations Center and Network Node Manager were integrated and known as OVO (short for OpenView Operation)

HP OpenView

HP has a history of acquiring network management software companies.

- 2004: Novadigm and its Radia suite
- 2005: Peregrine Systems with its IT asset and service management software
- 2006:  Mercury Interactive Corp.
- 2007: Opsware.

**In early 2007**, alongside the integration of Mercury Interactive, Peregrine Systems and Opsware products, **HP rebranded OpenView products as HP BTO** (Business Technology Optimization) Software under the HP Software Division business, **and the OpenView** and **Mercury names were phased out**.

**In 2017, HP (now HPE) completed a split of its software division: ITOM is now part of Microfocus portfolio**

# OVO MAIN FEATURES (1)

**OVO is a server-client framework**, with agents installed on monitored nodes.
Agent usually acts like schedulers for user provided scripts distributed on each node with related configuration file; each script is responsible for some checks against thresholds and for sending results to OVO engine using “OPCmon” command.

**OVO Engine receive data and show problems on dashboards; moreover it can handle manual and automatic specific actions**

OVO Engine manages the centralized configuration too.

# OVO MAIN FEATURES (2)

So, if Customer needs to monitor **cpu load** on a server, then he has to develop a **specific script** and install it on each monitored device. It is Customer responsibility to know which command to use, accordingly to the O.S. flavor (Linux, HP-UX, Solaris, AIX, Windows, etc.)
Customer needs to manage thresholds comparison in the script sending result using OVO call «OPCmon»

Same rules (a specialized script with thresholds) apply for any other check (i.e. HTTP) but log.

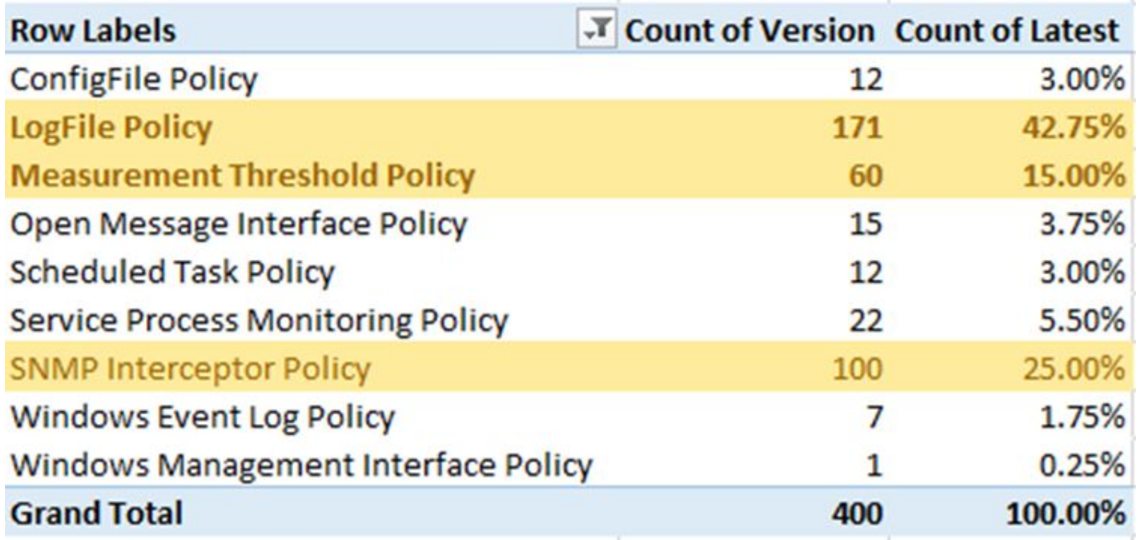
**Consequences:**

- *Customer needs to manage a lot of different scripts, and distribute on nodes possibly with specific configuration file (OVO engine handles the distribution)*
- *Customer needs to keep script versioning and document everything (OVO helps in this task)*
- ***OVO is used mainly as an alerting system, instead of a monitoring system***

HP OPENVIEW → ZABBIX

**Legenda:**

OVO Policy -> **Zabbix Template**
OVO Condition -> **Zabbix Trigger**
OVO Message Text -> **Zabbix Trigger Name**
OVO Help Text -> **Zabbix URL**
OVO Help Text -> **Zabbix Description**
OVO Automatic Action -> **Zabbix Scripts**
OVO Manual Action -> **Zabbix Scripts ??**



| Row Labels | Count of Version | Count of Latest |
|---|---|---|
| ConfigFile Policy | 12 | 3.00% |
| LogFile Policy | 171 | 42.75% |
| Measurement Threshold Policy | 60 | 15.00% |
| Open Message Interface Policy | 15 | 3.75% |
| Scheduled Task Policy | 12 | 3.00% |
| Service Process Monitoring Policy | 22 | 5.50% |
| SNMP Interceptor Policy | 100 | 25.00% |
| Windows Event Log Policy | 7 | 1.75% |
| Windows Management Interface Policy | 1 | 0.25% |
| **Grand Total** | **400** | **100.00%** |

# Our experience: OVO to Zabbix metrics and triggers migration strategy

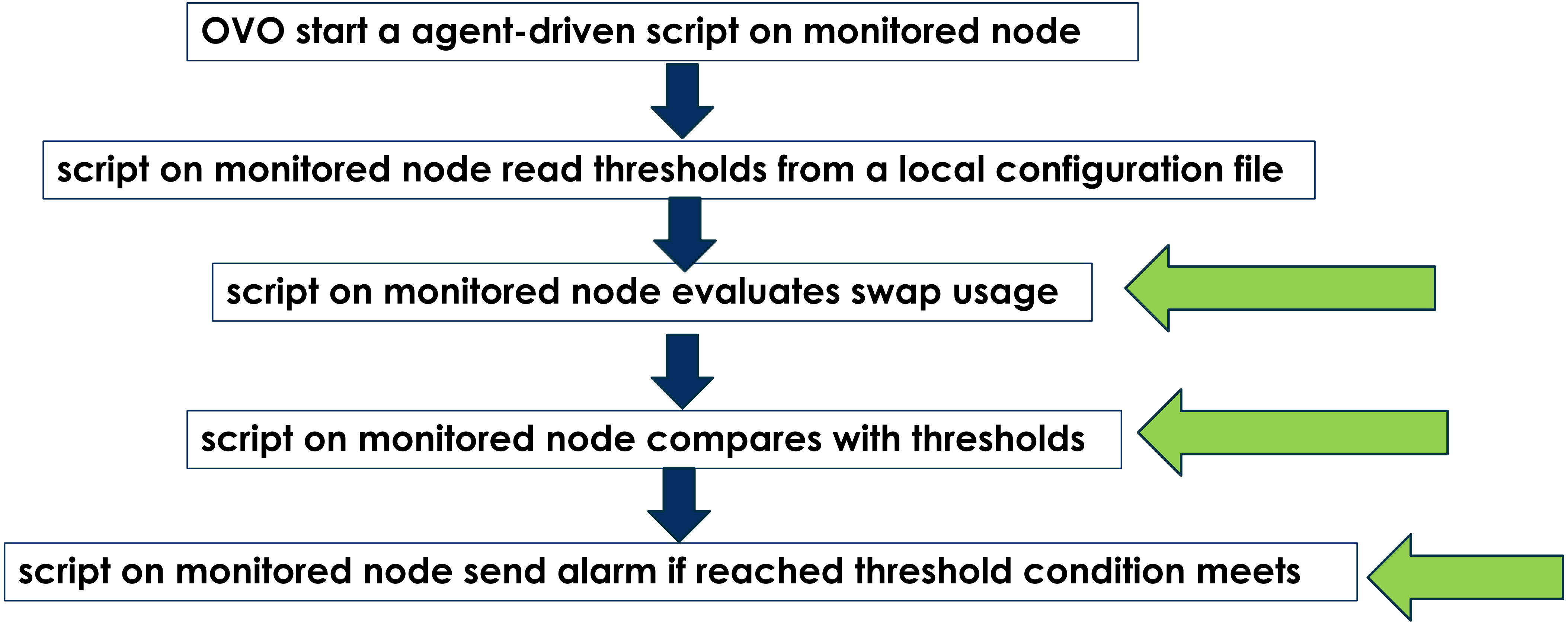
We planned migration in two steps:

- First step, translate every check from OVO to Zabbix without optimization. Exception is when a simple embedded Zabbix check perfectly fits OVO script checks
- Second step: optimize checks using Zabbix capabilities

First step:

| | |
|---|---|
| Monitoring scripts for Zabbix embedded metrics | -> **use Zabbix Agent metrics** |
| Monitoring scripts for other metrics | -> **use OVO script with Zabbix sender instead of OPCmon** |
| LOG monitoring | -> **translate to Zabbix log** |
| Monitoring scripts for http checks | -> **use Zabbix web scenario** |
| OVO SNMP traps | -> **translate to Zabbix SNMP traps** |

# EXAMPLE 1: MONITORING SWAP SPACE



OVO start a agent-driven script on monitored node

script on monitored node read thresholds from a local configuration file

script on monitored node evaluates swap usage

script on monitored node compares with thresholds

script on monitored node send alarm if reached threshold condition meets

# OVO Script for Unix swap evaluation

```
############################################################
# Check Swap used with Threshold
############################################################
case `uname` in
      "HP-UX" )
           SWAPUSED=`swapinfo -dft | grep '^total' | awk '{ print $5 }' | sed -e 's/%//'`
           ;;
      "SunOS" )
           #SWAPUSED=`swap -s | sed 's/k//g' | awk '{ printf("%.0f\n",100*($9/($9+$11))) }'`
                #FIX RM-10/2013: swap -s reports physical memory+swap
                SWAPUSED=`swap -l|awk '/[0-9]+/ {avail+=$4;free+=$5} END {print int(100-100*free/avail)}'`

           ;;
      "Linux" )
           dFreeOutput=`free -o -m | grep "^Swap:"`
           if [ $? -ne 0 ]; then
             dFreeOutput=`free -m | grep "^Swap:"`
           fi
                dUSED=`echo $dFreeOutput | awk '{print $3}'`
           dTOTAL=`echo $dFreeOutput | awk '{print $2}'`
           if [ $dUSED -gt 0 ]
           then
                   SWAPUSED=`expr $dUSED \* 100 / $dTOTAL`
           else
                   SWAPUSED=0
           fi
           ;;
esac
```

## OVO Script for swap evaluation (2)

```
MONVAL=0
if [ "${CRITPCT}" != "/" ] && [ ${SWAPUSED} -ge ${CRITPCT} ];then
        MONVAL=4; MSGPCT=$CRITPCT
elif [ "${MAJPCT}" != "/" ] && [ ${SWAPUSED} -ge ${MAJPCT} ]; then
        MONVAL=3; MSGPCT=$MAJPCT
elif [ "${MINPCT}" != "/" ] && [ ${SWAPUSED} -ge ${MINPCT} ]; then
        MONVAL=2; MSGPCT=$MINPCT
elif [ "${WRNPCT}" != "/" ] && [ ${SWAPUSED} -ge ${WRNPCT} ]; then
        MONVAL=1; MSGPCT=$WRNPCT
Fi

/opt/OV/bin/opcmon $MONITOR_NAME=$MONVAL -option VALUE="$SWAPUSED" –option THRESHOLD="$MSGPCT"
```

**Thresholds potentially tailored for each server**

ZABBIX
PREMIUM PARTNER

On **Zabbix** we use the agents, but:

**First request from Customer was to literally translate existing OVO way of life to Zabbix.**

Some issues:

- Each actual OVO script works using four thresholds described in a configuration file.
- Each configuration file could be specialized for some servers
- Some threshold are not actualized in the configuration file

- **So we decided to use specialized triggers with thresholds resolved by macros at template and host level**

# Example of OVO Thresholds for swap usage and translation for Zabbix using macro:

| OVO | value | **Zabbix** | **value** |
|---|---|---|---|
| 1-Warning | no set | $SWAP_WAR | 0 |
| 2-Minor | 90% | $SWAP_MIN | 90% |
| 3-Major | 99% | $SWAP_MAJ | 99% |
| 4-Critical | not set | $SWAP_CRI | 0 |

**Warning** triggers if value greater than or equal to Warning and less than Minor
**Minor** triggers if value greater than or equal to Minor and less than Major
**Major** triggers if value greater than or equal to Major and less than Critical
**Critical** triggers if value greater than or equal to Critical

## Trigger expressions will be more complex in order to reuse OVO thresholds and especially the “not set” ones; a simplified example for swap check is:

- **Warning triggers if**

```
(SWAP_WAR > 0 ) and (( SWAP_MIN > 0 and system.swap.size >= SWAP_WAR and system.swap.size
< SWAP_MIN) or (SWAP_MIN = 0 and system.swap.size >= SWAP_WAR))
```

- **Minor triggers if**

```
(SWAP_MIN > 0) and ((SWAP_MAJ > 0 and system.swap.size >= SWAP_MIN and system.swap.size <
SWAP_MAJ) or (SWAP_MAJ = 0 and system.swap.size >= SWAP_MIN))
```

- **Major triggers if**

```
(SWAP_MAJ > 0) and (( SWAP_CRI > 0 and system.swap.size >= SWAP_MAJ and system.swap.size <
SWAP_CRI) or (SWAP_CRI = 0 and system.swap.size >= SWAP_MAJ))
```

- **Critical triggers if**

```
SWAP_CRI > 0 and system.swap.size >= SWAP_CRI
```

At least two **corner case**

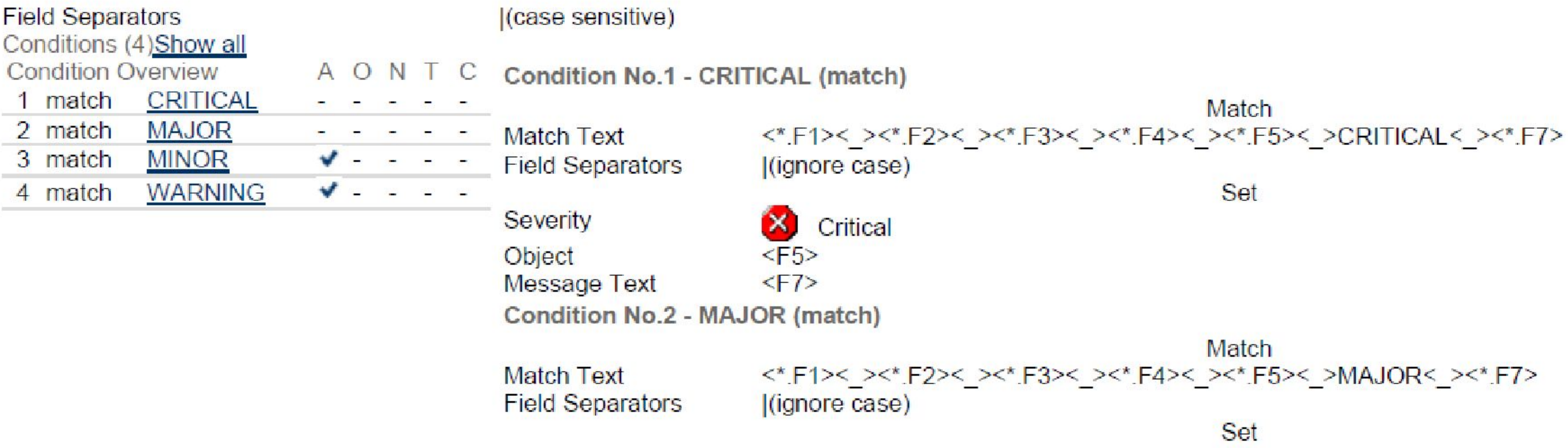
1. **What if a value increase** and from **WARNING** reach **MAJOR** threshold?

2. **What if a value decrease** and from **CRITICAL** pass to **MAJOR**?

**We decided to avoid automatic solution of problems (this solve case 2) and to leave multiple problems with different severity for case 1**

# EXAMPLE 2 : LOG MONITORING

**Starting from OVO policy:**



Field Separators |(case sensitive)

Conditions (4)Show all

| | | Condition Overview | A | O | N | T | C |
|---|---|---|---|---|---|---|---|
| 1 | match | CRITICAL | - | - | - | - | - |
| 2 | match | MAJOR | - | - | - | - | - |
| 3 | match | MINOR | ✓ | - | - | - | - |
| 4 | match | WARNING | ✓ | - | - | - | - |

**Condition No.1 - CRITICAL (match)**

Match

Match Text <*.F1><_><*.F2><_><*.F3><_><*.F4><_><*.F5><_>CRITICAL<_><*.F7>

Field Separators |(ignore case)

Set

Severity ☒ Critical

Object <F5>

Message Text <F7>

**Condition No.2 - MAJOR (match)**

Match

Match Text <*.F1><_><*.F2><_><*.F3><_><*.F4><_><*.F5><_>MAJOR<_><*.F7>

Field Separators |(ignore case)

Set

## Log parsing (i.e. Oracle logs)

In actual implementation, a scheduled script analyze Oracle log looking for errors, extract strings and build a intermediate log file like:

```
DB-ORA0001;pkgora1;ORA01;CRITICAL;other data
DB-ORA0003;pkgora3;ORA02;CRITICAL;DIRECT;other data
DB-ORA0004;pkgora4;ORA02;CRITICAL;LISTENER;other data
DB-ORA0005;pkgora01;ORA01;INFO;other data
```

## Representation in OVO policy is:

```
DB-ORA0001;<*.PKG>;<*.SID>;CRITICAL;<*.DATA>
DB-ORA0003;<*.PKG>;<*.SID>;CRITICAL;DIRECT;<*.DATA>
DB-ORA0004;<*.PKG>;<*.SID>;CRITICAL;LISTENER;<*.DATA>
DB-ORA0005;<*.PKG>;<*.INST>;INFO;<*.DATA>
```

## Log parsing (i.e. Oracle logs)

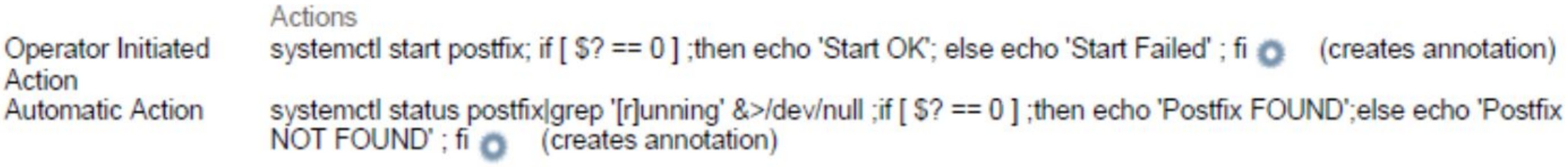
We decided to reuse this mechanism and configured one item and one trigger for each error condition, i.e. for first trigger:

**OVO Match Text**: **DB-ORA0001**;<*.PKG>;<*.LST>;**CRITICAL**;<*.DATA>

**Zabbix trigger expression**: {Template Oracle:log[{$ORA_LOG_NAME}].regexp(**DB-ORA0001.*CRITICAL**)}=1

**Zabbix trigger Name**: DB-ORA0001: Listener {{ITEM.VALUE}.regsub("(DB-ORA[0-9]+)\;(.*)\;(.*)\;(.*)\;(.*)", \3)} on package/host {{ITEM.VALUE}.regsub("(DB-ORA[0-9]+)\;(.*)\;(.*)", \2)} is NOT active

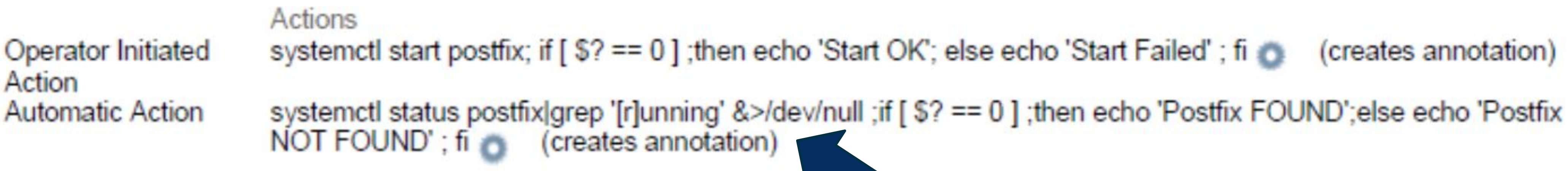
# OVO to Zabbix Automatic action strategy



| | Actions |
|---|---|
| Operator Initiated Action | systemctl start postfix; if [ $? == 0 ] ;then echo 'Start OK'; else echo 'Start Failed' ; fi ⚙ (creates annotation) |
| Automatic Action | systemctl status postfix\|grep '[r]unning' &>/dev/null ;if [ $? == 0 ] ;then echo 'Postfix FOUND';else echo 'Postfix NOT FOUND' ; fi ⚙ (creates annotation) |

On some triggers, OVO applies a specific automatic action: we can translate this into Zabbix actions, using **HOST.NAME** and **EVENT.ID** variables.

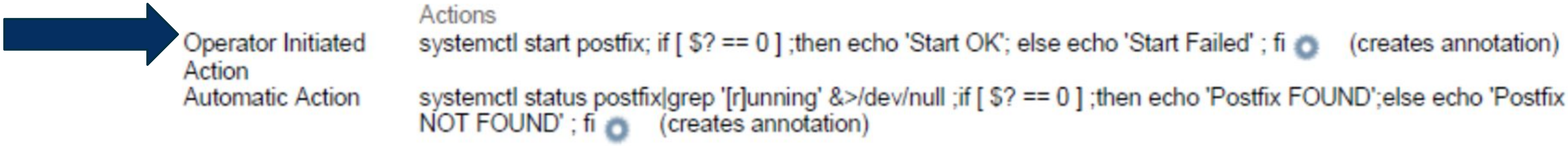
# OVO to Zabbix Automatic action result strategy



| | Actions |
|---|---|
| Operator Initiated Action | systemctl start postfix; if [ $? == 0 ] ;then echo 'Start OK'; else echo 'Start Failed' ; fi ⚙ (creates annotation) |
| Automatic Action | systemctl status postfix\|grep '[r]unning' &>/dev/null ;if [ $? == 0 ] ;then echo 'Postfix FOUND';else echo 'Postfix NOT FOUND' ; fi ⚙ (creates annotation) |

Very often OVO put **automatic action result** into **problem history**: we wrote a **post-action script** for keeping action results into «note» field of Zabbix event. **This implies executing automatic scripts only on Zabbix Server**.

# OVO to Zabbix Manual action strategy



|  | Actions |
|---|---|
| Operator Initiated Action | systemctl start postfix; if [ $? == 0 ] ;then echo 'Start OK'; else echo 'Start Failed' ; fi (creates annotation) |
| Automatic Action | systemctl status postfix\|grep '[r]unning' &>/dev/null ;if [ $? == 0 ] ;then echo 'Postfix FOUND';else echo 'Postfix NOT FOUND' ; fi (creates annotation) |

On many conditions, **OVO operator can run a specific manual action**: we cannot translate this into Zabbix, because **`EVENT.ID`** value is not visible to general scripts.

We modified Zabbix source code obtaining **`EVENT.ID`** for general scripts too, **while asking for a general solution**

**An example of this behavior is for *opening a Remedy Ticket***

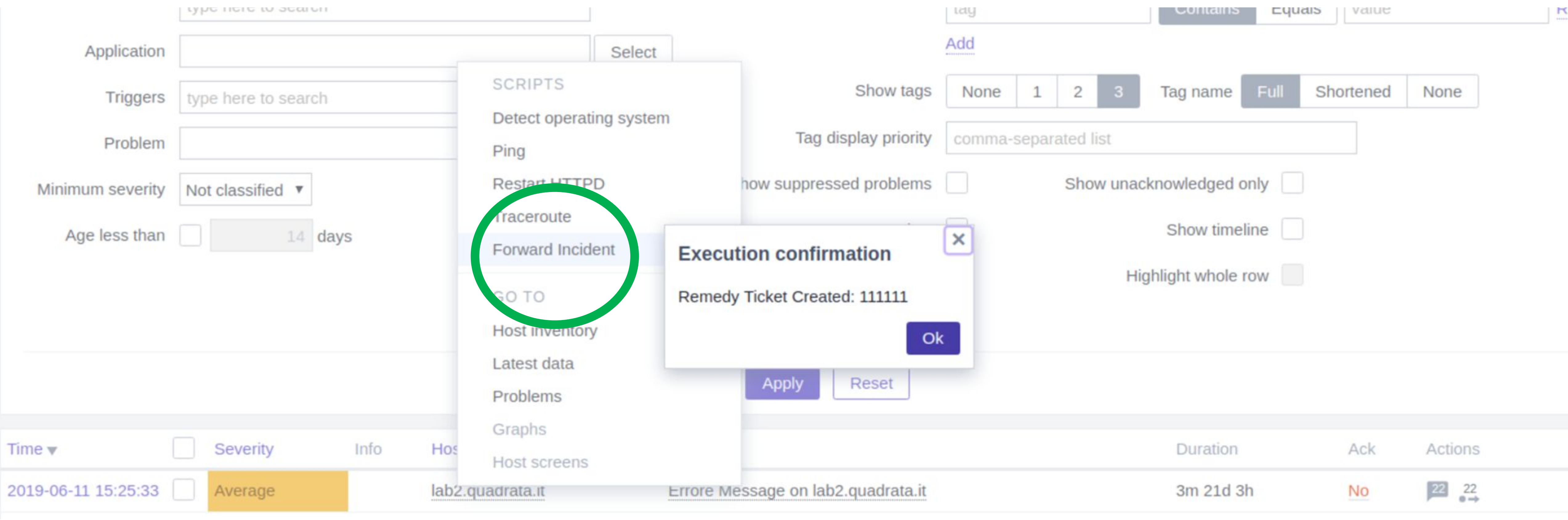
# OVO to Zabbix Remedy Integration: opening a ticket

**OVO operator can manually open a Remedy ticket** in case of suitable problems; **when Remedy ticket is solved, then OVO problem will automatically follow and closed too**.

For Remedy manual ticket opening, we followed the general solution for manual operator initiated action, using **`EVENT.ID`** for **extracting** Remedy needed data, and a Perl script for sending the Zabbix problem info to Remedy, while acquiring the Remedy ticket number and return code

**NOTE: OVO Operator can open a single Remedy ticket associated to multiple error conditions (this is difficult on Zabbix side)**

# OVO to Zabbix Remedy Integration: opening a ticket



Execution confirmation

Remedy Ticket Created: 111111

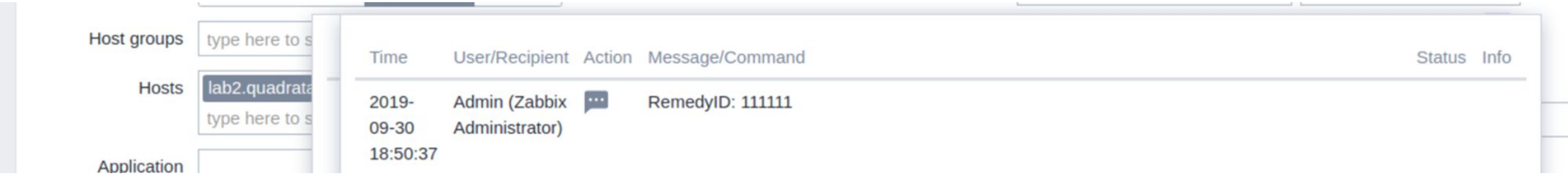
Ok

| Time | Severity | Info | Hos |  | Duration | Ack | Actions |
|---|---|---|---|---|---|---|---|
| 2019-06-11 15:25:33 | Average |  | lab2.quadrata.it | Errore Message on lab2.quadrata.it | 3m 21d 3h | No | 22 22 |

ZABBIX PREMIUM PARTNER

# OVO to Zabbix Remedy Integration: Ticket ID

Last step after opening a Remedy ticket is register the Remedy ID (ticket number) ***into Zabbix Event Actions Message/Command***



| Time | User/Recipient | Action | Message/Command | Status | Info |
|---|---|---|---|---|---|
| 2019-09-30 18:50:37 | Admin (Zabbix Administrator) | 💬 | RemedyID: 111111 | | |

## OVO to Zabbix Remedy Integration: closing a Zabbix problem

For Zabbix automatic problem closing following Remedy ticket closing, we developed a hourly scheduled script running on Zabbix Server, flow is:

- Scan all Zabbix open problems looking for Remedy ticket IDs
- Verify each ticket ID against closing info on Remedy
- Close all problems on Zabbix only if Remedy solved ticket exist

# Migrating OVO to Zabbix: lesson learned

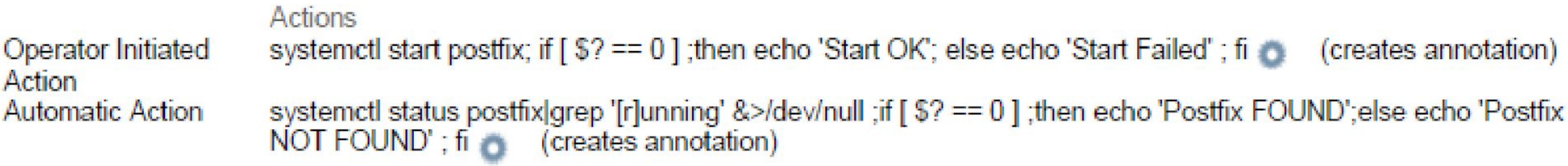
- We started the first step beginning of July 2019
- We are now closing first step (150 working days) – end of October 2019
- We will start second step in November, and we plan at least 150 more days

So:

- OVO to Zabbix migration is definitely not a simple task, but we can handle it
- Pay attention in evaluating efforts for the whole project
- Ask for Customer involvement

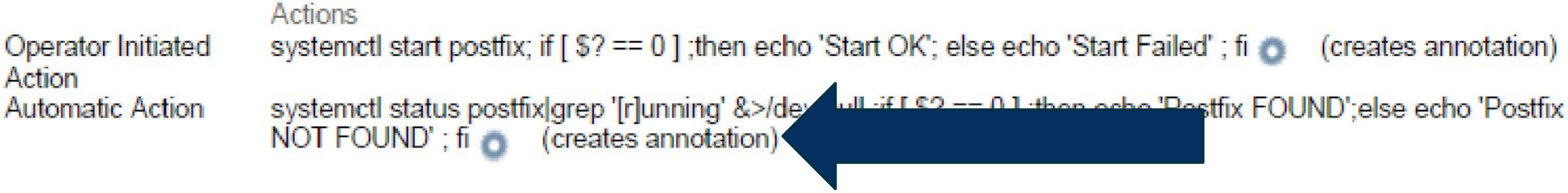
# What we like in OVO

Something, like **operator initiated (manual) action integrated in event handling**:



| | Actions |
|---|---|
| Operator Initiated Action | systemctl start postfix; if [ $? == 0 ] ;then echo 'Start OK'; else echo 'Start Failed' ; fi ⚙ (creates annotation) |
| Automatic Action | systemctl status postfix\|grep '[r]unning' &>/dev/null ;if [ $? == 0 ] ;then echo 'Postfix FOUND';else echo 'Postfix NOT FOUND' ; fi ⚙ (creates annotation) |

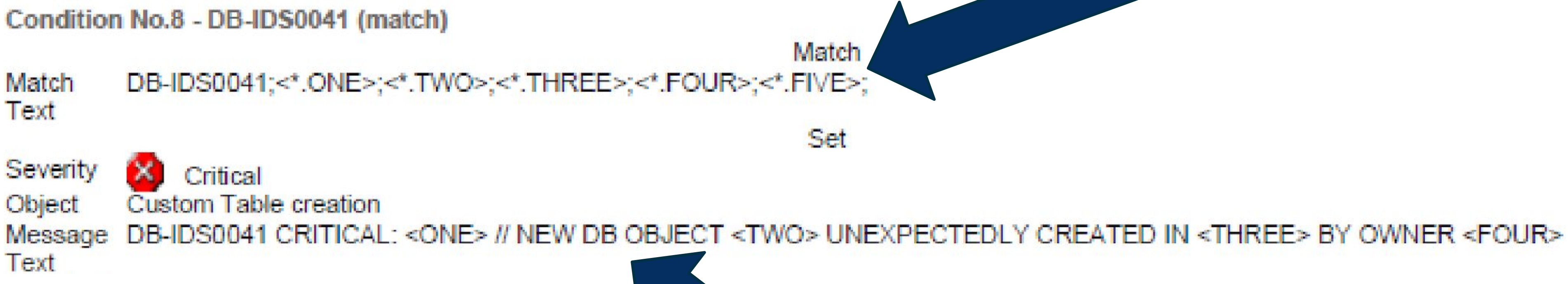
And **redirecting output in problem history**:



| | Actions |
|---|---|
| Operator Initiated Action | systemctl start postfix; if [ $? == 0 ] ;then echo 'Start OK'; else echo 'Start Failed' ; fi ⚙ (creates annotation) |
| Automatic Action | systemctl status postfix\|grep '[r]unning' &>/de [illegible] stfix FOUND';else echo 'Postfix NOT FOUND' ; fi ⚙ (creates annotation) |

# What we like in OVO (2)

Last but not least, **log fields simple extraction method**:



Condition No.8 - DB-IDS0041 (match)

Match

| | |
|---|---|
| Match Text | DB-IDS0041;<*.ONE>;<*.TWO>;<*.THREE>;<*.FOUR>;<*.FIVE>; |

Set

| | |
|---|---|
| Severity | ☒ Critical |
| Object | Custom Table creation |
| Message Text | DB-IDS0041 CRITICAL: <ONE> // NEW DB OBJECT <TWO> UNEXPECTEDLY CREATED IN <THREE> BY OWNER <FOUR> |

## What we ask to Zabbix

- A way for obtaining **`EVENT.ID`** inside Global Scripts
- Multi-selection in **Problem View / Dashboard** for associating a **Global Script**
- Official solution for Global Script/Automatic Action **Output redirected to Event Notes**
- An **Italian Espresso Machine**

# THANK YOU!

info@quadrata-group.com

https://t.me/zabbixitalia