Roadmap
Plans and Ideas for the future
Mathias Kettner & Lars Michelsen
Our Roadmap

“The way forward is clear.”
Our Roadmap (2nd try)
Our roadmap is created by you

Actual demands of customers have precedence over our own bright ideas!
Already commissioned

THIS WILL COME FOR SURE
Intelligent

„Activate Changes“
Current problem with „Activate Changes“

- You cannot activate just your changes
- You cannot undo just your changes
- You cannot undo selected changes anyway
Another problem with changes

- Some things get immediately active, some not
- Local site behaves differently
- Restart of monitoring implicitly activates some changes
Why do we need to activate changes in the first place??
Advantages of „Activate Changes“

• Control the exact time when a set of changes gets active

• Chance to review changes and maybe roll back
New concept

- Selected changes can be activated while keeping others pending
- Selected changes can be rolled back while keeping others
Sounds easy?
Example 1:

1. Create a folder
2. Create a host in that folder
3. Roll back Change 1

→ Changes depend on each other!
Example 2:

1. Delete host myserver123
2. Create (new) host myserver123
3. Rollback change 1

→ Rollback of 1 would create invalid configuration with duplicate host!
Example 3:

1. Delete a rule
2. Roll back change 1

→ Changes need knowledge about the previous situation
Example 4:

1. Delete host myserver123
2. Create (new) host myserver123
3. Rollback change 1

→ Rollback of 1 would create invalid configuration with duplicate host!
Implementation (1)

- Each change is an (intelligent) object that can apply / rollback itself
- And it can be saved / loaded on disk.
- Changes know about dependencies to other changes
Implementation (2)

- WATO will be reorganized internally so that everything it does is such a change.
WATO Web-API is the lucky guy

- Due to this formalization of changes every WATO operation will be available via the API
Time controlled check parameters
Check parameters

- Changing parameters based on time
- Can be used with all Check_MK checks
  → „Parameters for discovered services“
- Transparent for checks
  → Check gets the current parameters
How to specify the time?

• We are using time periods
• Problem:
  Only admins can configure time periods
• Solution:
  Make them configurable by users
Structured status data
Make checks produce more useful data

• We capture a lot of data
• Why not make it better usable?
• Example:
  - Export current database usage together with database inventory information
Status Quo

- Checks: Status code, text, perfdata
- HW/SW-Inventory: Structured data

VS.
Extend HW/SW inventory?

- Status data is captured often
- Data can change each interval

→ Conflict with HW/SW inventory history (differential mode)
How to?

- New check function „data_function‟
  - Extends a tree with own data
- Status GUI: Merges both trees
- Export merges data optionally
Plans
WE PROBABLY WILL DO THIS
Tickets via GUI
Make ticketing less painful

• Takes time: Collecting needed information
  – Meta data (OS, Check_MK version, ...)
  – Mandatory diagnose information

• Could be improved with some kind of guided ticket creation
GUI Integration

• Add a ticket creation module

• Dynamic form with mandatory fields based on categories

• Components: Core, GUI, Event Console, ...

• Categories: Bug, Configuration problem, Performance, Development
Example: Check development request

• Ask all common questions
  – What to monitor?
  – How to get data?
  – Provide MIBs
  – WATO configuration needed?
  – Metrics needed?
  – ...?
Check_MK Check Helper Memory

• Get deeper understanding
  – Need real environments for this
  – Measure

• Possibly change helper model
Improve Livestatus Proxy scaling

• We need more than 300 sites
• Limits:
  – Number of file descriptors, Python GIL
• Change to multi process architecture
Apache / GUI

- A lot of load is caused by inactive sessions (hidden tabs / windows)
- Disable update while inactive
- Restart the update again when focused
SLAs
## Current functionality:

<table>
<thead>
<tr>
<th>Host</th>
<th>Service</th>
<th>OK</th>
<th>WARN</th>
<th>CRIT</th>
<th>UNKNOWN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lastrechner</td>
<td>Filesystem /</td>
<td>83.52%</td>
<td>16.30%</td>
<td>0.16%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Lastrechner</td>
<td>Filesystem /bauwelt</td>
<td>95.15%</td>
<td>0.87%</td>
<td>3.97%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Lastrechner</td>
<td>Filesystem /chroots</td>
<td>99.44%</td>
<td>0.18%</td>
<td>0.37%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Lastrechner</td>
<td>Filesystem /boot</td>
<td>99.99%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Lastrechner</td>
<td>Filesystem /opt/omd</td>
<td>95.61%</td>
<td>4.15%</td>
<td>0.22%</td>
<td>0.00%</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td></td>
<td><strong>94.74%</strong></td>
<td><strong>4.30%</strong></td>
<td><strong>0.94%</strong></td>
<td><strong>0.00%</strong></td>
</tr>
</tbody>
</table>
• Precise analysis of actual availability
• Many ways to tune computation
• Also for BI aggregations
• Simplistic „SLA“-Check, just coloring cells according to percentages
Real SLA can be much more complex.
Idea - step 1:

- New GUI module for specifying more complex SLAs
  - Based on different time periods
  - Also number, length of outages
  - Expandable in a modular way
- Edited by users just like views and reports.
Idea - step 2:

- Assign these SLAs to hosts and services via rules
- Assign to BI aggregates via BI module
Idea - step 3:

- View have new optional columns for showing SLA status
- Not only yes / no, but also amount of (under-)fullfillment
- Also available in the current time period.
Idea - step 4:

- Actual SLA-Monitoring
- E.g. gets you warned if your **allowed** downtime for the current period is running low.
- Helps you fullfulling your SLAs!
- Helps you planning downtimes.
Thanks for listening!
YOUR FEEDBACK IS WELCOME