

Nagios Plugins für NetApp-Filer entwickeln

Workshop am 12. September 2008 in Nürnberg

Vorgeschichte

- 2005 Spezialisierung auf Systemmonitoring, Trendanalyse, OSS, Nagios
- 2006 Upgrade und Administration des ÖGB-Nagios
- 2007 Erstellen von Plugins für VMware ESX3, Brocade Switches, HP iLO BMCs, u.a.
 - ▶ NetApp: erste Versuche mit SNMP und Erstellung von NetApp-Plugins (Telnet/SSH basierend)
- 2008 Auftrag größere Anzahl umfassender Plugins für NetApp-Filer zu erstellen

NetApp Plugins 2008



- Auftraggeber CUBiT
- Work in Progress ...
- Auf die Anforderungen dieses Kunden zugeschnittene Plugins
 - Datenbankabfragen statt Kommandozeilen-Parameter
 - Mehrsprachig (Deutsch als Default, Englisch und Schwedisch über GNU-gettext)
 - single-/multiline Ausgabe (für Nagios 3)
 - Umfassend (Hardware, ONTAP, Aggregate, traditionelle Volumes, Flexvols, Snapshots, WAFL-Scans, Plattenzustand, gcount, Spigelsynchronität, Brocade-Switches, vFiler, Deduplication (df -s), Domainchecks (CIFS), qtrees, sysstat, Cache-Effektivität, ...)

Vorgangsweise in diesem Workshop

1. Einige theoretische Erklärungen (Nagios-Plugins und DataONTAP Schnittstellen)
2. Vorstellung eines sehr einfachen Codegerüsts (bereits funktional)
3. Vorschlag Erweiterungen 1
4. Hands-On: Integration dieser Erweiterungen
5. Vorschlag Erweiterungen 2 - Ziel: vollwertiges Nagios-Plugin
6. Vorführung und Erklärung an Hand eines fertigen Beispiels (wäre sehr viel Tipparbeit)
7. Vorschlag Erweiterung 3 - Ziel: Performancedaten, Timeout-Management
8. Hands-On: Integration von Vorschlag 3

Datenquellen für Nagios-Plugins

- SNMP
 - schlankes, zuverlässiges Protokoll
 - Version 1: Counter laufen über
 - Version 2: unverschlüsselt (nur sichere Netze)
- SMTP: Verarbeiten von Warnmails
- HTTP: Perl&LWP
- Telnet/SSH: Net::SSH
- Webservices: HTTP(S), XML

Beispiel Telnet/SSH

```
# connect either via telnet or ssh
if (defined $opts{telnet}) {
    my $t = Net::Telnet->new( Timeout => 10,
        Prompt => '/>/',
        Host => $host,
        );

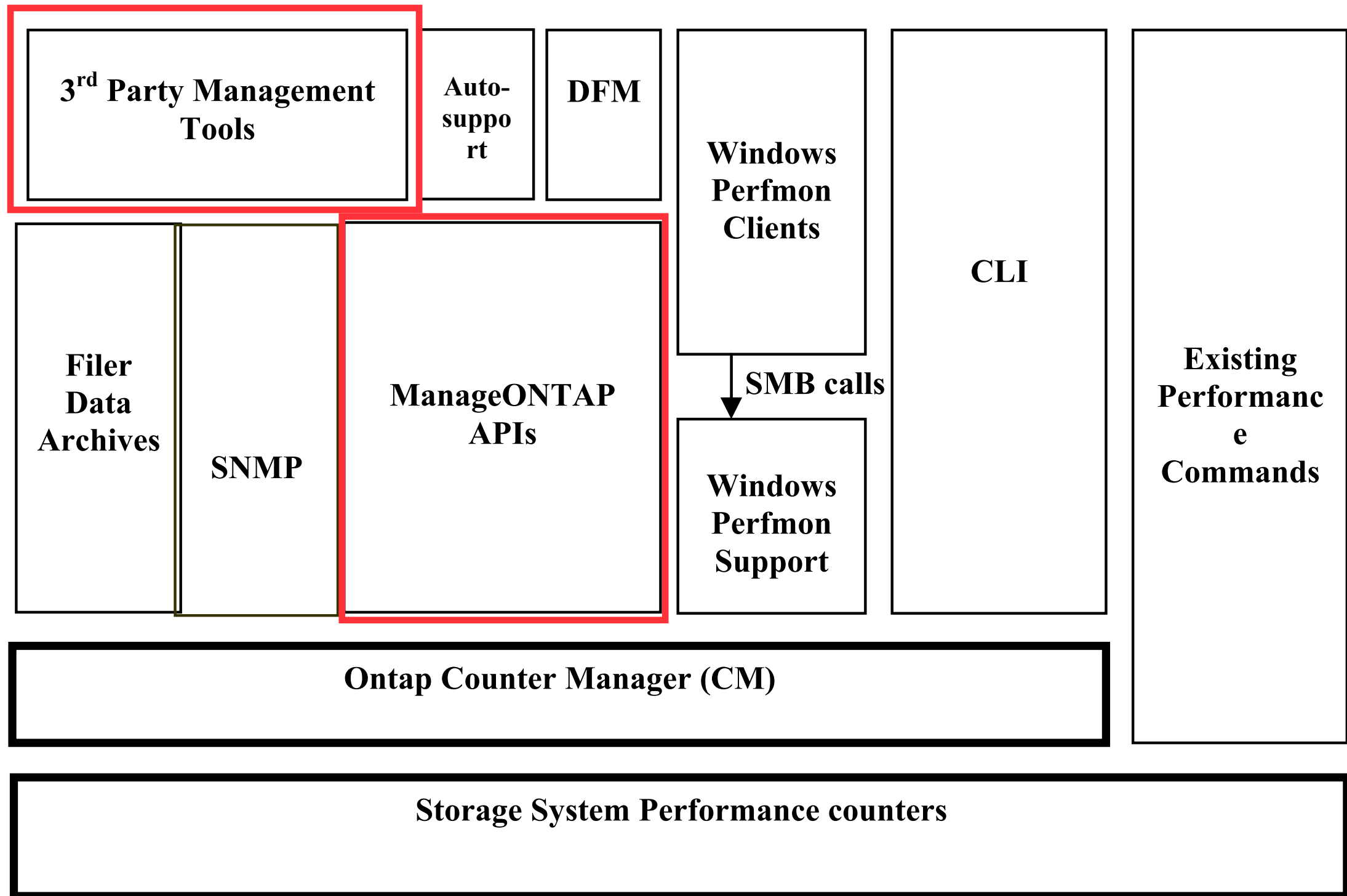
    $t->login($user, $pass);
    @dlist = $t->cmd("storage show disk");
    @dnames = get_disknames(); #(v4.15 v4.16 ...)
    %GCOUNTS = get_gcounts(\$t);
    foreach (keys %GCOUNTS) {
        if (defined $opts{debug}) {
            print "$_ --> $GCOUNTS{$_}\n";
        }
        if ( $GCOUNTS{$_} >= $warn and $exit!=2 ) {$exit = 1}
        if ( $GCOUNTS{$_} >= $crit )           {$exit = 2}
    }
} else {
    print "SSH not implemented yet - sorry! Pls. use --telnet.\n";
    exit 3;
}
```

Probleme Telnet/SSH

- Net::SSH ist teils schwer zu implementieren
- Andere SSH-Module forken - schlecht für Performance/Sicherheit
- semiprofessioneller Ansatz - Änderungen im CLI können fatale Auswirkungen haben. (Das CLI wurde nicht für automatisierte Abfragen konzipiert und wird vom Hersteller kaum unter diesem Aspekt gewartet und weiterentwickelt)
- Telnet machte mit neueren ONTAP-Versionen Probleme

Webservices Schnittstelle

- professioneller Ansatz (vom Hersteller unterstützt)
- relativ zukunftssicher, hat sich bei den ESX-Plugins schon bewährt
- sprachunabhängig (Java, C, Perl)
- Kommunikation über HTTP oder HTTPS
- Übertragung von XML-Files
- Strikt objektorientierte Datenstrukturen, die in Perl z.B. auch mit DataDumper ausgegeben werden können



Blockdiagramm

NetApp Performance
Infrastruktur

NetApps perf-API: Terminologie

Organisationsbez.	repräsentiert	Beispiel
Objekte	Subsysteme	processor
Instanzen	physische oder logische Einheiten	processor0
Counter	„datum“ (Messwert)	processor_busy

Manage-ONTAP-SDK



SDK_help.htm

Manage-ONTAP-SDK

The screenshot displays a web application interface for the Manage-ONTAP-SDK. The top navigation bar includes 'Contents', 'Index', and 'Search' tabs. The breadcrumb trail shows 'Home > Manage ONTAP S'. The left sidebar contains a tree view of the SDK structure, with 'Data ONTAP APIs' selected. The main content area features a blue header for 'Data ONTAP APIs', followed by an 'Introduction' section and a 'Content of this section' section with a bulleted list of API categories.

Home > Manage ONTAP S

Data ONTAP APIs

Introduction

The Data ONTAP APIs are used to access and manage the N of APIs. This set includes APIs for security management, lic replication, data archiving and so on.

Content of this section

This section provides API information for the Data ONTAP A

- [aggr](#)—provides aggregate information and managemen
- [cf](#)—provides the cluster-failover operations
- [cifs](#)—provides the CIFS setup, sessions and shares

NetApp API: Verwendung im Überblick

Schritt	Aufgabe	Parameter/Infos
1	Serverkontext erzeugen	Servername
2	Sessionparameter setzen	Protokoll (HTTP/HTTPS) user-credentials
3	Kommando an die Core API des Filers senden	Kommandoname
4	Datenstruktur auswerten	child_get_* ...

NetApp API: Verwendung im Detail

Schritt	Aufgabe	Parameter/Infos	Code
1	Serverkontext erzeugen	Servername	<code>\$s = NaServer->new (\$filer, \$major_version, \$minor_version);</code>
2	Session-parameter setzen	Protokoll (HTTP/HTTPS) user-credentials	<code>\$s->set_transport_type("HTTP"); \$s->set_style("LOGIN_PASSWORD"); \$s->set_admin_user(\$user, \$pass);</code>
3	Kommando an die Core API des Filers senden	Kommandoname	<code>\$out = \$s->invoke("volume-list-info"); # print Dumper(\$out) if \$debug > 2;</code>
4	Datenstruktur auswerten	child_get_* ...	<code>\$volume_info = \$out->child_get("volumes"); @result = \$volume_info->children_get(); foreach \$vol (@result){ \$vol_name = \$vol->child_get_string("name"); ... }</code>

NetApp Simulator installieren

Simulator läuft als eigene „Maschine“ in einem Linux (mit eigener IP)

- ➔ OS X: <http://lanti.wordpress.com/2008/04/09/netapp-simulator-fur-mac-os-x/>
- ➔ Linux/Windows: Doku von VMware im Installationskit (Vmware, Linux and Simulator installation.doc)

Netzwerkkonfiguration „Host-only“

- VMware Netzwerk auf Host-Only und verbinden
- Netzwerk der VM neu starten: `/etc/init.d/networking restart`
- VM bekommt eine Adresse wie `172.16.151.128`
- Simulator starten:
- Netzwerk des Simulators rekonfigurieren: `ifconfig ns0 172.16.151.132 netmask 0xffffffff`
- Testen: `$ telnet 172.16.151.132`
- Hosteintrag in `/etc/hosts` (spart viel Tipperei)

Starten des NetApp Simulators

1. VM starten
 2. Anmelden als ubuntu%ubuntu (= Username „ubuntu“ und Passwort „ubuntu“)
 3. Starten des Simulators: /sim3/runsim.sh
- Optional
 - ggf. Anmelden als root%
 - Useraccount für Abfragen über die API: nagios%nagios88

check_netapp_space.pl - Version 0.1

- Erstes Skript zur Abfrage von Performancewerten - noch kein Nagios-Plugin
- Anforderungen
 - Kontaktieren des Servers
 - Daten holen und auswerten
 - Ausgabe auf die Comandline

Code und Ausgabe Version 0.1

```
# Schritt 1: Servercontext erzeugen
my $s = NaServer->new ($filer, ...);

# Schritt 2: Sessionparameter setzen
check_out($s->set_transport_type("HTTP"));
check_out($s->set_style("LOGIN_PASSWORD"));
check_out($s->set_admin_user($user, $pass));

# Schritt 3: Sende Kommando an die Core API
$out = $s->invoke( "volume-list-info");
...
my $volume_info = $out->child_get("volumes");
my @result = $volume_info->children_get();

foreach my $vol (@result){
    my $vol_name = $vol->child_get_string("name");
    print "Volume name: $vol_name \n";
    my $size_total = $vol->child_get_int("size-total");
    print "Total Size: $size_total bytes \n";
    my $size_used = $vol->child_get_int("size-used");
    print "Used Size: $size_used bytes \n";
    my $usedPrzt = $vol->child_get_int("percentage-used");
    print "Used Size: $usedPrzt % \n-----\n";
}
}
```

```
$ ./check_netapp_space-01.pl
Volume name: cachevol
Total Size: 20971520 bytes
Used Size: 81920 bytes
Used Size: 0 %
-----
Volume name: vol0
Total Size: 253640704 bytes
Used Size: 66895872 bytes
Used Size: 26 %
-----
Volume name: sv_vol
Total Size: 16777216 bytes
Used Size: 278528 bytes
Used Size: 2 %
-----
```

Datenstrukturen 1: Data::Dumper

- Ausgabe mit Datadumper
- Ausgabe kann theoretisch auch geparkt werden
- Eigentliche Aufgabe:
 - Datenstrukturen auspacken
 - Datenstrukturen verstehen
 - Debugging (z.B. fehlerhafte Dokus aufdecken, ...)
- Code (wenn \$out eine Referenz auf eine Datenstruktur ist):

```
use Data::Dumper;  
....  
print Dumper($out);
```

Datenstrukturen 2: Ausgabe von Data::Dumper

```
$VAR1 = bless( {
  'content' => '',
  'name' => 'results',
  'children' => [
    bless( {
      'content' => '',
      'name' => 'volumes',
      'children' => [
        bless( {
          'content' => '',
          'name' => 'volume-info',
          'children' => [
            bless( {
              'content' => 'cachevol',
              'name' => 'name',
              'children' => [],
              'attrvals' => [],
              'attrkeys' => []
            }, 'NaElement' ),
            bless( {
              'content' => 'bb1ff1 ...',
              'name' => 'uuid',
              ... ),
            bless( {
              'content' => 'flex',
              'name' => 'type',
              ...),
            bless( {
              'content' => 'online',
              'name' => 'state',
              ... ),
            bless( {
              'content' => '20971520',
              'name' => 'size-total',
```

Datenstruktur und Befehle

```
$VAR1 = ( {'content' => '',
          'name' => 'results',
          'children' => [( {
                        'content' => '',
                        'name' => 'volumes',
                        'children' => [( {
                                    'content' => '',
                                    'name' => 'volume-info',
                                    'children' => [( {
                                                'content' => 'cachevol',
                                                'name' => 'name',
                                                'children' => [],
```

```
$out = $s->invoke( "volume-list-info" );
...
my $volume_info = $out->child_get("volumes");
my @result = $volume_info->children_get();

foreach my $vol (@result) {
    my $vol_name = $vol->child_get_string("name");
```

volume-list-info

Get volume status. Note that all RAID-related status items (e.g., 'raid-size', 'raid-status', 'checksum-style') reported for a flexible volume actually describe the state of its containing aggregate.

Input Name	Range	Type	Description
verbose		boolean optional	If set to "true", more detailed volume information is returned. If not supplied or set to "false", this extra information is not returned.
volume		string optional	The name of the volume for which we want status information. If not supplied, then we want status for all volumes on the filer.

Output Name	Range	Type	Description
volumes		volume-info[]	List of volumes and their status information.

```
$out = $s->invoke( "volume-list-info" );  
...  
my $volume_info = $out->child_get( "volumes" );  
my @result = $volume_info->children_get();  
  
foreach my $vol (@result) {  
    my $vol_name = $vol->child_get_string( "name" );
```

Hands-On (Version 0.1)

- ➔ Ausgabe der Volumeinfos auf die Comandline
- ➔ Einbau von Datadumper und Ausgabe der Datenstruktur



check_netapp_space.pl - Version 0.3

- prinzipiell sinnvolles Nagios-Plugin
- Anforderungen wie in Version 0.1 und zusätzlich
 - Nagios-konforme Übernahme von Servername, Credentials, Schwellwerten als Parameter (Nagios::Plugin)
 - Ermitteln und Ausgabe des Maximalwertes (das am meisten belegte Volume)
 - Rückgabewert (0,1,2) abhängig von den Schwellwerten

Demo und Erklärung Version 0.3



check_netapp_space.pl - Version 0.5

- vollwertiges Nagios-Plugin
- Anforderungen wie in Version 0.3 und zusätzlich
 - Nagios-konforme Ausgabe der Performance-Daten (Nagios::Plugin::Performance)
- Timeoutmanagement

Performance Daten aller Volumes plotten

```
foreach my $vol (@result) {  
    ...  
    $p->add_perfdata(  
        label => "$vol_name",  
        value => $belegtPrzt,  
        uom   => "%",  
        threshold => $p->threshold(),  
    );  
    ...  
}
```

Performance Daten der Volumes plotten - Ausgabe

```
foreach my $vol (@result) {  
    ...  
    $p->add_perfdata(  
        label => "$vol_name",  
        value => $belegtPrzt,  
        uom   => "%",  
        threshold => $p->threshold(),  
    );  
    ...  
}
```

```
$ ./check_netapp_space-05.pl -H sim3 -p nagios88
```

```
NETAPP_SPACE-05 OK - Max. used volume on sim3 is vol0 (26.00 % used) |
```

```
cachevol=0%;70;90 vol0=26%;70;90 sv_vol=2%;70;90
```

Timeout-Management

```
my $timeout = $p->opts->timeout;

$SIG{ALRM} = sub {
    $p->nagios_die("ERROR: Timeout ($timeout s) reached");
};

# hierher kommt der eigentliche Code des Plugins,
# vor allem Abfragen zum Filer und alles was sonst
# noch hängen bleiben kann.

alarm $timeout;
```

Timeout-Management - Ausgabe

```
my $timeout = $p->opts->timeout;

$SIG{ALRM} = sub {
    $p->nagios_die("ERROR: Timeout ($timeout s) reached");
};

# main code here

alarm $timeout;
```


```
$ ./check_netapp_space-05.pl -H sim3 -p nagios88
NETAPP_SPACE-05 UNKNOWN - ERROR: Timeout (15 s) reached
```

Hands-On (Version 0.5)

- ➔ Nagios-konforme Ausgabe der Performance-Daten
- ➔ Timeoutmanagement integrieren



To be continued ...

- wahlweise auch Aggregate abfragen
- Ausgabe umfangreicher gestalten
- und vieles anderes mehr
- Weiterentwicklungen ev. zu finden auf
 - [Nagiosexchange](#) 
 - <http://ingo.lantschner.name/Nagios.html>

Danke ...

... für's Mitmachen und noch eine
schöne Konferenz!

ingo@lantschner.name

+43-664-143 84 18

<http://ingo.lantschner.name>

<http://itblog.lantschner.name>

ingo@lantschner.name

