

- **Arbeitsspeicherauslastung** Beobachtet wie viel Arbeitsspeicher vom System verwendet wird und wieviel davon noch zur Verfügung steht.

### Prozesse

- **IdcServerNT.exe** Der Windowsprozess des Stellent-Servers
- **IdcAdminNT.exe** Der Windowsprozess für <sup>das</sup> die Administration ~~Web-~~ <sup>ja</sup> ~~interface~~ des Stellent-Servers
- **w3wp.exe** Der Windowsprozess des Microsoft „Internet Information Services“

### Services

Hier stellt sich die Frage was der Unterschied zu Prozesse ist. Wieso beides über- wacht wird. Evt. Referenzen zu Erläuterung

- • **IdcContentService** Den Zustand des „sccdms01“-Dienst überprüfen.
- **IdcAdminService** Den Zustand des „sccdms01\_admin“-Dienst für die Administration überprüfen.
- **Zeitsynchronisationsdienst** Überprüfen, ob der „W32TIME“-Dienst, der für den Zeitabgleich mit einem Zeitserver zuständig ist, läuft und die Abweichung zwischen Client und Zeitserver festhalten.
- **Antivirusdienst** Den Zustand den Dienstes überprüfen, der für die ständig Updates des Virusscanners Symantec AntiVirus notwendig ist. <sup>heißt wie?</sup>

## 6.2 Überwachung der Funktionalität

Durch die vorherigen Tests kann herausgefunden werden, ob eine Anwendung oder ein Dienst auf dem Server gestartet wurde. Die Funktionalität kann durch solche Überprüfungen jedoch **nicht** sichergestellt werden. Da beispielsweise der Prozess bzw. Dienst des Webserver gestartet ist, jedoch keine Webseite aufgerufen werden kann. Daher muss eine weitere Art von Überprüfungen/~~Checks~~ <sup>testen</sup> die Anwendungen auf ihre Funktionalität ~~(hin)~~ <sup>überprüfen</sup>.

- **Webserver Aufruf einer Webseite auf dem Server.** Wenn auf diese Anfrage eine gültige Antwort in Form einer Statuscode-Meldung erfolgt, <sup>die korrekte Funktion</sup> kann der reale/wirkliche Zustand des Webserver festgestellt werden.
- **Webinterface des Oracle UCM** Zusätzlich wird mit dieser Abfrage die Integration des Content-Management-Systems in den Webserver überwacht, da hier nicht nur der Webserver, sondern eine UCM spezifische Webseite abgefragt wird.
- **Benutzeranmeldung am Oracle UCM** Hier wird getestet, ob sich ein Benutzer erfolgreich am System anmelden kann. Dies wird mit Anmeldeungsdaten eines lokalen Benutzers und eines Active Directory-Benutzers durchgeführt um gleichzeitig/~~zusätzlich~~ die Verbindung zum ADS-Server zu testen.
- **Oracle Datenbank** <sup>überprüfen den Verbindungsaufbau zur Datenbank.</sup> Wenn keine Verbindung zur Oracle Datenbank möglich ist, können keine neuen Informationen gespeichert werden.
- **Anzahl Datenbankverbindungen** Anzahl der Verbindungen zur Datenbank, da aus Performanzgründen eine Obergrenze mit einer maximalen Anzahl festgelegt ist. <sup>oder "guten" Verbindungen?</sup>

### 6.3 Auswerten von Logdateien

In dieser Kategorie werden zusätzlich verschiedene Logdateien auf spezielle Warnungs- und Fehlermeldungen anhand eindeutigen Stopwörtern untersucht. Dies ist notwendig um reaktiv Fehlverhalten der Anwendung zu erkennen, das nicht mit den vorherigen Überwachungselementen entdeckt wurde. Des weiteren können durch die Analyse der Logdateien etwaige Alarmmeldungen der bisherigen Tests bestätigt, begründet oder aufgehoben werden. Somit bietet das Auswerten der Logdateien zusätzliche Sicherheit False Positive- oder False Negative-Meldungen auszuschließen.

sehr gut!

Die Oracle UCM Anwendung erstellt drei verschiedene Arten von Logdateien:<sup>26</sup>

- Content Server Log
- Inbound Refinery Log
- Archiver Log

Um alle Logs ohne Probleme im Internetbrowser anzuzeigen, liegen alle Logdateien im HTML-Format vor. Alle drei Arten von Logs bestehen jeweils aus 30 verschiedenen Dateien, die sich täglich abwechseln. Dadurch wird für jeden Tag im Monat eine separate Datei verwendet, um bei vielen Warnungs- und Fehlermeldungen durch die chronologische Anordnung/~~Hierarchie~~ den Überblick zu behalten. Dabei werden die Logdateien zwangsweise nach 30 Tagen nacheinander überschrieben.

Diese Rotation der Logdateien muss bei der Durchsuchung nach Signal/Stopwörter beachtet werden, damit stets die aktuelle Logdatei überwacht wird und keine veralteten Informationen für False Positives-Meldungen durch Nagios sorgen.

#### 6.4 Benutzersimulation

- **Einchecken** von Dokumenten Damit die eigentliche Aufgabe des Dokumentenverwaltungssystem überwacht werden kann, werden verschiedene Datenformate testweise eingchecked. Dabei wird die Antwort der Anwendung auf das Hinzufügen der Dateien analysiert.
- **Konvertierung** Da das hinzugefügte Dokument nicht nur einfach auf dem Server gespeichert wird, sondern dabei auch in ein anderes Format umgewandelt wird, muss diese Konvertierung zusätzlich überwacht werden. Wird beispielsweise ein Bild eingchecked, wird dieses mehrfach in

Ein Dokument nimmt in seinem Lebenszyklus/Abb. 3 verschied. Zustände an. So kommt es nach dem Eincheck ind. Zustand 'genutzt' bis die Konvertierung erfolgt ist.

Danach ist es im Zustand 'fertig' bis die erfolgreiche Indizierung durch den Zustand 'freigegeben' angezeigt wird.

Die Benutzersimulation hat die Aufgabe alle Schritte der Zustandsänderung durch typische Abfragen zu prüfen.

<sup>26</sup>Quelle: [UCMlog09]

( verschiedenen Auflösung <sup>er in einem</sup> oder ~~als anderen~~ Bilddateiformat gespeichert. Ob diese Transformation erfolgreich ablief, kann anhand dieser neuen Dateien festgestellt werden.

- **Indizierung** Bei dem Einchecken sollen auch gleichzeitig zusätzliche Informationen über das Dokument festgehalten werden. Diese Informationen können beispielsweise der Name des Autors, das Erstellungsdatum der Datei oder - bei Bildern - der verwendete Farbraum sein. Bei der Suche nach einem Dokument können diese Informationen als zusätzliche Suchkriterien verwendet werden. Daher muss überprüft werden, ob diese Daten richtig ausgelesen werden, der Datenbank hinzugefügt und vom Anwender abgefragt werden können. Dabei werden auch zuvor festgelegte/ausgewählte Testdateien verwendet.

- **Suchfunktion** Nach einer erfolgreichen Indizierung muss das eingetragene Dokument per Suchanfrage gefunden werden. Ob die Suche und Indizierung erfolgreich abgelaufen ist, wird zusätzlich überprüft. <sup>? sah veraltet ist nicht</sup>

Die Basis für die alle anderen Tests bildet die Systemüberwachung. An erster Stelle der Systemüberwachung steht die schlichte Erreichbarkeit über das Netzwerk per Ping. Wenn der Server nicht erreichbar ist, können auch keine weiterführende Prüfungen durchgeführt werden. Zur Systemüberwachung gehören auch allgemeine Informationen über die Systemressourcen wie freier Festplattenspeicher oder CPU Auslastung. Die nächste <sup>Stufe oder Ebene</sup> Grundlage für die ~~darüber liegenden Tests~~ bildet die Überprüfung der laufenden Prozesse und der Status verschiedener Dienste bzw. Services. Sollten bestimmte Prozesse nicht gefunden werden oder wichtige Dienste nicht gestartet sein, können auf diese Prozesse und Dienste aufbauende Checks nicht funktionieren. Beispielsweise kann der Funktionalitätstest der Benutzeranmeldung nicht realisierbar sein, wenn bereits zuvor in der Systemüberwachung der Prozess für den Webserver IIS nicht gefunden werden konnte. Alle Überwachungselemente

lassen sich inklusive ihrer Abhängigkeiten in einer Pyramide darstellen, <sup>des Abb. 24</sup> siehe ~~Abbildung 24.~~

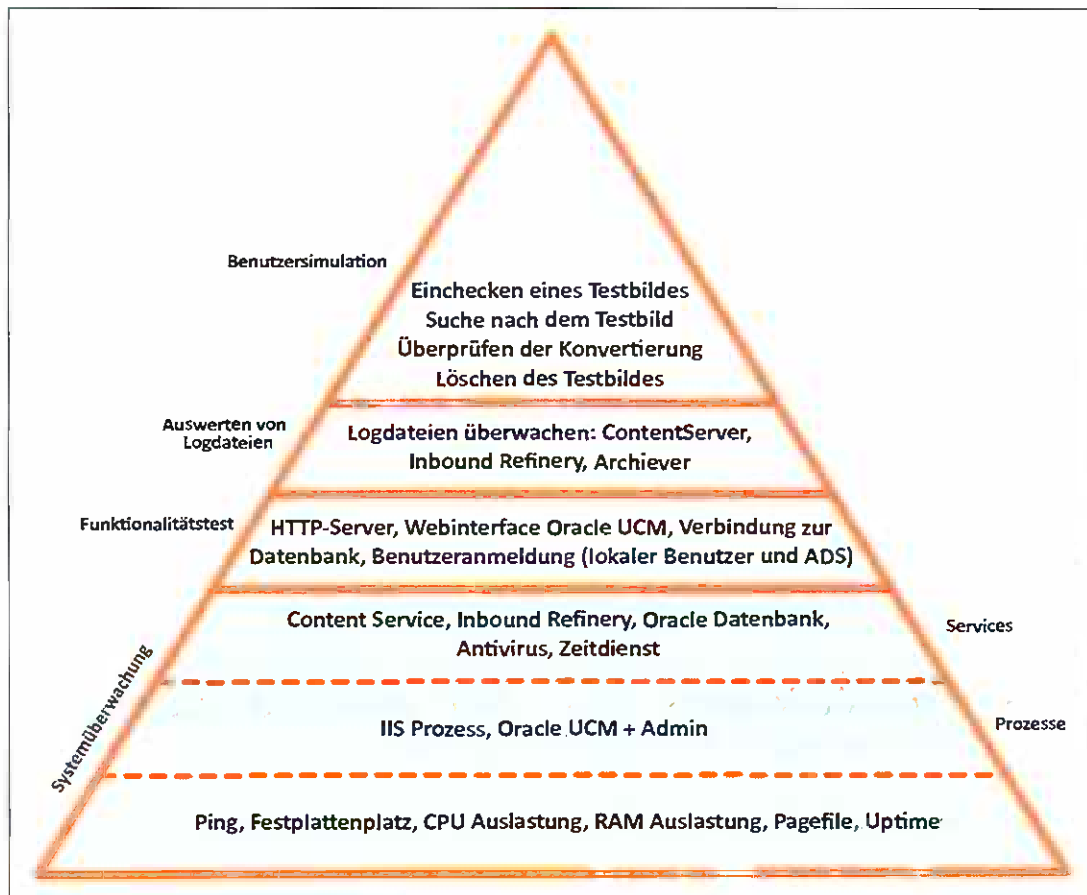


Abbildung 24: Überwachungselemente

## 7 Umsetzung

In diesem Kapitel wird die Vorgehensweise der zuvor beschriebenen Problemstellungen erörtert.

### 7.1 Aufbau der Testumgebung

#### 7.1.1 Aufsetzen eines Nagios-Test-Systems

Da die einzelnen Überwachungselemente in der Überwachungssoftware Nagios <sup>subressive</sup> nach ~~und nach~~ eingetragen (~~/ definiert / assoziiert / verbunden~~) werden müssen, ist ein häufiges Neustarten der Nagios-Anwendung notwendig, damit die neuen Konfigurationsdateien übernommen werden.

Damit dies nicht auf dem bereits verwendeten Nagios-Server durchgeführt werden muss, wird ein Nagios-Testserver für diesen Zweck eingesetzt.

Da Nagios ein Unix-ähnliches Betriebssystem erfordert, wird für diesen Zweck die Linux-Distribution Debian als Betriebssystem des Testservers verwendet, <sup>welche auch in den Produktivservern des KIT verwendet wird.</sup>

#### 7.1.2 Bilddatenbank als virtuelle Maschine

Für die Simulation der verschiedenen Fehlerzustände der einzelnen Überwachungselemente wird eine virtuelle Maschine mit einer Oracle UCM Prototypinstallation, ~~die extra~~ als Entwicklungsplattform ~~erstellt wurde~~, verwendet.

### 7.2 Übersicht Nagios-Agenten

<sup>Wie</sup> Im ~~vorherigen~~ Kapitel 4.3 wurde <sup>aufgeführt, ist für...</sup> erklärt, dass für das Überwachen von Diensten, die nicht über das Netzwerk zugreifbar sind, ein Nagios-Agent auf dem jeweiligen Rechner benötigt wird.

In diesem Unterkapitel werden die populärsten Agenten für Unix und Windows Betriebssysteme aufgelistet und nach/in den Punkten Sicherheit, subjektiver Aufwand für die Konfiguration und Art der Abfragemethode (aktiv oder passiv) vergleicht.

Es gab Alternativen...  
Wieso entscheidet Du Dich für Agenten?  
Ent danach kommt Dudec Auswahl bringen

### 7.2.1 Unix-Agenten

Für die auf Unix basierenden Betriebssysteme werden fünf verschiedene Möglichkeiten angeboten, die ~~(auch)~~ zuvor in der Abbildung 15 als verschiedene Überwachungsmöglichkeiten von Nagios aufgelistet wurden.

	SSH	NRPE	SNMP	SNMP Traps	NSCA
<b>Methode</b>					
<i>aktiv</i>	✓	✓	✓	-	-
<i>passiv</i>	-	-	-	✓	✓
<b>Sicherheit</b>					
<i>Passwort</i>	-	-	✓ (v3)	✓ (v3)	-
<i>Accesslist</i> <sup>1</sup>	✓ <sup>2</sup>	✓	✓ (v2)	✓ (v2)	✓
<i>Verschlüsselung</i>	✓	✓	✓ (v3)	✓ (v3)	✓
<b>Aufwand</b> <sup>3</sup>	niedrig	normal	hoch	hoch	normal

<sup>1</sup> Einschränkung der Abfrage der Überwachungsinformationen anhand der IP-Adresse

<sup>2</sup> Über zuvor ausgetauschte SSH-Schlüssel

<sup>3</sup> Subjektive Einschätzung

Tabelle 2: Übersicht der verschiedenen Unix-Agenten

Dabei werden drei Agenten genannt, die eine aktive Ausführung der Nagios-Plugins benutzen. Alle drei Agenten unterscheiden sich jedoch in den ~~(anderen)~~ Punkten Sicherheit und Aufwand. Der auf SSH basierende Agent besitzt einen relativ geringen Aufwand für die Installation, da für den Aufbau der Kommunikation zwischen Nagios-Server und Client nur der öffentliche Schlüssel des Servers auf dem Client eingetragen werden muss. Dadurch kann der Nagios-Server sich ohne Passwortabfrage an dem zu überwachendem Host anmelden und die sich darauf befindlichen Nagios-Plugins ausführen. Da auf den meisten Unix-Servern bereits ein SSH-Server läuft und deshalb kein wei-



terer Port geöffnet oder eine weitere Software installiert werden muss, ist diese Methode den anderen meist vorzuziehen.

Bei der NRPE-Methode wird eine weitere Softwarekomponente auf dem Client installiert, die einen separaten Port für die Kommunikation mit dem Nagios-Server öffnet. Wie bei dem Aufruf per SSH müssen sich hier die Nagios-Plugins bereits auf dem Rechner befinden. Dabei gilt als Unterschied dieser zwei ähnlichen Methoden zu beachten, dass für die Ausführung der Checks per SSH ein extra Benutzerkonto auf dem Client erstellt werden muss und somit beliebige Systembefehle ausgeführt werden können, während die Ausführung von Kommandos bei NRPE nur auf vorkonfigurierte Befehle beschränkt ist, *wie in* *aufgeführt.* *Siehe auch* Kapitel 4.3.2. Da SNMP plattformunabhängig funktioniert ist es mögliche diese Variante bei Unix- sowie bei Windowsservern einzusetzen. Die verwendete SNMP-Version bestimmt welche Sicherheitsmerkmale zur Verfügung stehen. Zwar gibt es bereits seit Version 1 die Möglichkeit den Zugriff per Passwort in drei Gruppen aufzuteilen: kein Zugriff, Leserecht und Lese- mit Schreibrecht<sup>27</sup>, jedoch wird dieses Passwort im Klartext übertragen, so dass es leicht auslesbar ist. Auch die SNMP-Version 2 inklusive der erweiterten Version 2c verwendet die gleiche unsichere Authentifizierung. Erst ab Version 3 wird das Passwort verschlüsselt übertragen. Während Barth behauptet, dass man bei SNMP generell kein Passwort verwenden soll<sup>28</sup>, da es leicht per Netzwerkmittschnittprogramme wie WireShark ausgelesen werden kann, wird in [Jose07] S. 121 klargestellt, dass die Version 3 eine verschlüsselte Authentifizierung durch den MD5- oder SHA-Algorithmus ermöglicht. *prima!*

Die passive Variante über SNMP bei der der Client die Ergebnisse der Checks an den Nagios-Server sendet, auch SNMP-Traps genannt, funktioniert nach dem gleichen Prinzip. Da das Auslesen der MIB per SNMP im Gegensatz zu

<sup>27</sup>Quelle: [Barth08] S. 237

<sup>28</sup>Quelle: [Barth08] S. 238



den anderen Varianten deutlich komplexer ist, wird der Aufwand als hoch eingestuft.

Ein weiterer Vertreter, der passive Checks ermöglicht, ist der NSCA-Agent. Wie die anderen Unix-Agenten bietet es die Möglichkeit den Datenaustausch zwischen Nagios-Server und Client zu verschlüsseln. Alle Unix-Agenten erlauben es den Zugriff auf die Nagios-Plugins auf bestimmte IP-Adressen zu beschränken. Die Liste mit diesen IP-Adressen nennt man auch *Accesslist*.

### 7.2.2 Windows-Agenten

Da die zu überwachende Oracle UCM Anwendung auf einem Windows-Server betrieben wird und die bereits vorgestellten Agenten mit Ausnahme der SNMP-Varianten nur unter Unix einsetzbar sind, müssen zusätzlich die explizit für Windows entwickelten Nagios-Agenten untersucht werden. Dabei wird die Auswahl der Kandidaten auf 4 Bewerber beschränkt, siehe Tabelle 3.

Der NSClient-Dienst liefert die Möglichkeit lokale Windows-Ressourcen über das Netzwerk mit eigenem Port (Standardport 1248) abzufragen. Das Plugin *check\_nt* wurde explizit für diesen NSClient-Dienst entwickelt und steht durch die Nagios-Plugins standardmäßig zur Verfügung. Dadurch können <sup>die</sup> grundlegende Informationen für die Systemüberwachung <sup>aus Kap. 6.1</sup> wie Zustände von Prozesse, Services, CPU-Auslastung, Festplattenplatz, usw. abgefragt werden, ~~siehe~~ Kapitel 6.1.

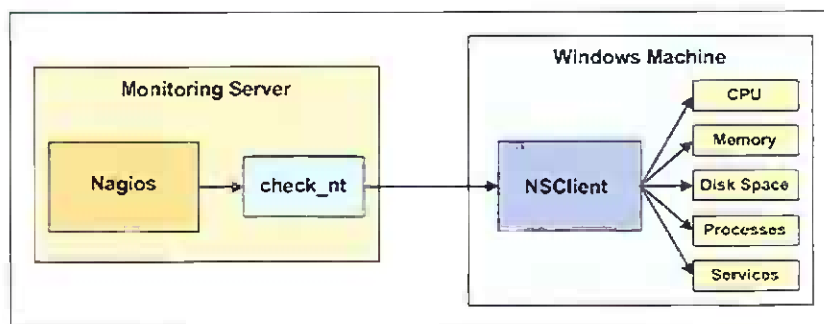


Abbildung 25: Abfrage von Windows-Ressourcen durch *check\_nt*

	NSClient	NRPE-NT	NC-net	NSClient++	OpMon Agent
<b>Methode</b>					
<i>aktiv</i>	✓	✓	✓	✓	✓
<i>passiv</i>	-	-	✓	✓	-
<i>NSClient</i> <sup>1</sup>	✓	-	✓	✓	✓
<i>NRPE</i> <sup>2</sup>	-	✓	✓	✓	✓
<b>Sicherheit</b>					
<i>Passwort</i>	✓	✓	-	✓	✓
<i>Accesslist</i> <sup>3</sup>	-	-	✓	✓	✓
<i>Verschlüsselung</i>	-	✓	✓	✓	-
<b>Aufwand</b> <sup>4</sup>	normal	hoch	normal	normal	normal

<sup>1</sup> Kompatibilität mit dem NSClient-Dienst

<sup>2</sup> Erlaubt Ausführung von vorkonfigurierten Kommandos

<sup>3</sup> Einschränkung der Abfrage der Überwachungsinformationen anhand der IP-Adresse

<sup>4</sup> Subjektive Einschätzung

Tabelle 3: Übersicht der verschiedenen Windows-Agenten

Diese Abfrage kann durch die Ausführung auf der Kommandozeile getestet werden:

```
root@nagiosdev:~/usr/lib/nagios/plugins# ./check_nt -H example.kit.edu -v CLIENTVERSION
NSClient++ 0.3.6.813 2009-06-14
```

Abbildung 26: Zugriff auf den NSClient-Dienst durch check\_nt

Der erste und zugleich älteste Agent NSClient wird nicht mehr aktiv entwickelt und ist als aktuellste Version 2.0.1 aus dem Jahre 2003 bereits recht alt. Daher wird auch keine Verschlüsselung der ein- und ausgehenden Daten unterstützt. Auch bietet NSClient keine Möglichkeit aktiv vom Nagios-Server aus Nagios-Plugins auszuführen, die sich auf dem zu überwachendem Host befinden.

len. Dabei werden diese Komponenten nicht standardmäßig geladen, sondern im nächsten Dialogfenster als Auswahl explizit/extra abgefragt:

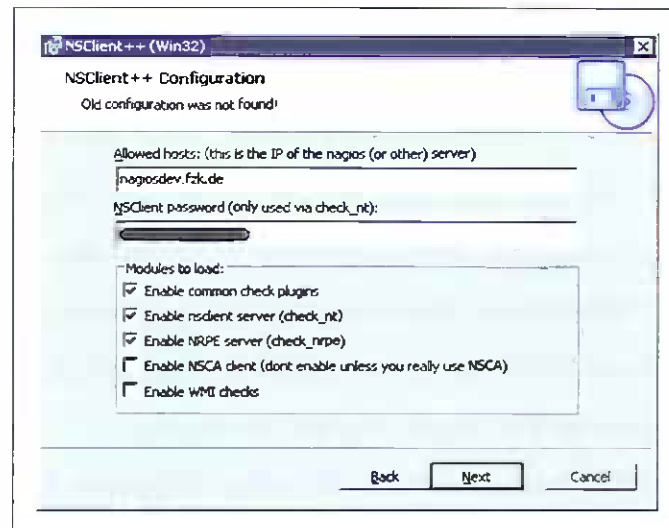


Abbildung 27: Konfiguration des NSClient++ während der Installation

Außerdem können direkt während der Installation die IP-Adresse bzw. der FQDN des Nagios-Servers und das gewünschte Passwort eingetragen werden. Durch die während des Installationsprozesses geladenen Komponenten für den NSClient- und NRPE-Dienst können die Standard-Nagios-Plugins *check\_nt* und *check\_nrpe* mit dem Windows-Server verwendet werden. Dabei läuft die Kommunikation zwischen dem Nagios- und dem Windows-Server folgendermaßen ab:

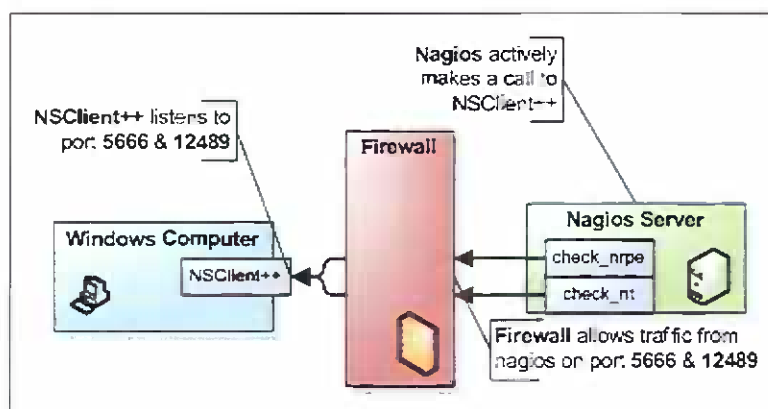


Abbildung 28: Kommunikation zwischen Nagios und NSClient++<sup>29</sup>

Um dies auch für Windows-Server zu ermöglichen gibt es eine auf Windows portierte NRPE-Variante, die sich NRPE\_NT nennt. Hier lassen sich die Plugins direkt über den Nagios-Server aufrufen und die ausgetauschten Informationen werden verschlüsselt über das Netzwerk übertragen.

Beide bisher genannte Windows-Agenten bieten keine Möglichkeit eine *Accesslist* anzulegen, erst das Programm NC\_net bietet diese Möglichkeit inklusive dem Sicherheitsmerkmal Verschlüsselung an. Außerdem können durch den eingebauten NRPE-Dienst aktiv Nagios-Plugins auf dem Client aufgerufen werden. Als Besonderheit lassen sich durch NC\_net sowohl aktiv als auch passiv Testergebnisse an den Nagios-Server übertragen.

Der Nagios-Agent NSClient++ besitzt diesselben Merkmale wie NC\_net, jedoch kann der Nagios-Server noch über ein Passwort zusätzlich verifiziert werden.

Die Möglichkeit Informationen per SNMP und SNMP-Traps abzufragen ist auch unter Windows möglich. Dabei gelten die gleichen Richtlinien, Hinweise und Einschränkungen wie zuvor in Kapitel 7.2.1.

### 7.2.3 Auswahl und Konfiguration des Nagios-Agenten

**Auswahl** Anhand der im vorherigen Kapitel beschriebenen Übersicht der Windows-Agenten und der daraus resultierenden Übersichtstabelle 3 wird ein geeigneter Kandidat für die Testumgebung ausgewählt. Da nur ein Windows-Agent alle drei Sicherheitsmerkmale anbietet und dabei aktive und passive (passiv eigntl nicht relevant zur Aufgabenlösung, jedoch nice2have) Überwachungsmethoden erlaubt, fällt die Wahl auf das Programm NSClient++.

**Installation und Konfiguration** Der Nagios-Agent NSClient++ kann im Gegensatz zu den meisten anderen Windows-Agenten komfortabel über einen graphischen Benutzerdialog installiert werden. Während der Installation kann auch festgelegt werden welche Komponenten installiert werden sol-

Wie ist es mit dem Community Support  
nach der Weiterentwicklung bei  
diesem Agenten

Bei Windows-Server mit vielen Verbindungen und Diensten <sup>Können</sup> zu Remote Procedure Calls (RPC) ~~kommen kann~~, bei denen dynamisch Ports ab 1025 verwendet werden, <sup>bereits da</sup> ~~könnte der~~ Standardport des NSClient-Dienstes (1248) ~~unter Umständen~~ <sup>schon</sup> bereits vor dem Start des Dienstes belegt sein.<sup>30</sup> Um dies zu verhindern wurde der Port des NSClient-Dienstes beim NSClient++ bereits auf einen höheren Port (12489) gewechselt.

Alle bisherigen Einstellungen können in der Konfigurationsdatei *NSC.ini*, die sich in dem Installationsverzeichnis des NSClient++ befindet, verändert werden. In dieser Datei befinden sich noch mehr Einstellungsmöglichkeiten; im Folgenden werden nur für die Umsetzung relevanten (notwendig essentiell benötigten) Parameter aufgelistet.

```

1 ;# NSCLIENT PORTNUMMER
2 ;# Die Portnummer des NSClient-Dienstes
3 port=13596
4
5 ;# NRPE PORTNUMMER
6 ;# Die Portnummer des NRPE-Dienstes
7 port=13597
8
9 ;# SSL SOCKET
10 ;# Die Aktivierung von SSL der Kommunikation zwischen Nagios- und Windows-
    Server
11 use_ssl=1
12
13 ;# NRPE BEFEHLSDEFINITIONEN
14 ;# Definitionen der Befehle, die durch den NRPE-Dienst aufrufbar sind
15 check_uname=scripts\check_uname.exe
16 check_reflog=scripts\check_logfiles.exe -f scripts\logfile.cfg

```

Listing 6: NSClient++ Konfigurationsdatei

Damit die vorgenommen Änderung übernommen werden, muss der Dienst des NSClient++ neu gestartet werden.

Durch das Ausweichen auf höher gelegene Portnummern können die zuvor genannten Probleme aufgrund der RPCs verhindert werden.

Der bereits höher liegende Standardport des NSClient-Dienstes beim NSClient++ wird zusätzlich noch abgeändert, damit die Tatsache, dass sich ein Nagios-Agent auf dem Computer befindet, nicht sofort ersichtlich ist. Dieser

<sup>29</sup><http://nsclient.org/nscp/>

<sup>30</sup>Quelle: [Barth08] S. 481

*Ist das auch eine zusätzliche  
Sicherheit?  
Never use Standardport?  
GReddy*

sicherheitstechnische Aspekt wurde bereits in Kapitel 3.1.3 behandelt.

Damit die SSL-Verschlüsselung zwischen den Servern aktiviert wird, muss man es explizit in der Konfigurationsdatei mit der Option *use\_ssl=1* angeben. Die Definitionen der NRPE-Kommandos dienen dafür, dass durch den Nagios-Server per *check\_nrpe* mit dem Befehlsnamen der darauf folgende Befehl ausgeführt wird.

Aufgrund der abgeänderten Portnummer muss man den Port bei dem Aufruf explizit angeben. Ein Aufruf eines solchen NRPE-Kommandos vom Nagios-Server wird in der folgenden Abbildung gezeigt:

```
root@nagiosdev:/usr/lib/nagios/plugins# ./check_nrpe -H example.kit.edu -p 13596 -c check_uname
Operating System UK - Microsoft(R) Windows(R) Server 2003 Standard Edition Service Pack 2
```

Abbildung 29: Aufruf eines NRPE-Kommandos

Anhand des Befehlsnamens *check\_uname* führt der NRPE-Dienst die in der Konfigurationsdatei eingetragene Datei *check\_uname.exe* aus.

Der Aufruf um Informationen durch den NSClient-Dienst abzufragen sieht ähnlich aus:

```
root@nagiosdev:/usr/lib/nagios/plugins# ./check_nt -H example.kit.edu -p 13596 -s secret -v UPTIME
System uptime - 49 day(s) 21 hour(s) 41 minute(s)
```

Abbildung 30: Aufruf des NSClient-Dienstes

Die Servicedefinition des vorherigen NSClient-Aufrufs muss wie nachfolgend/folgt in der Nagios-Konfiguration eingetragen:

```
1 define service{
2     use                generic-service
3     host_name          example.kit.edu
4     service_description Uptime
5     check_command      check_nt!-p 13596 -s secret -v UPTIME
6 }
```

Listing 7: Servicedefinition des NSClient-Checks

Damit nicht jeder einzelne Serviceeintrag abgeändert werden muss, falls sich der Port oder das Passwort des zu überwachenden Computers ändert, können eigene Befehlsdefinitionen erstellt werden.

```
1 define command{
2     command_name      check_nt_example
3     command_line       /usr/lib/nagios/plugins/check_nt -H $HOSTNAME$ -p
                        13597 -p secret -v $ARG1$
4 }
```

Listing 8: Server spezifische Befehlsdefinition

Dadurch muss nur diese Befehlsdefinition bei einer Änderung bearbeitet werden. Die vorherige Servicedefinition in Listing 30 kann dann in verkürzter Form eingetragen werden:

```
1 define service{
2     use                  generic-service
3     host_name            example.kit.edu
4     service_description  Uptime
5     check_command        check_nt_example!UPTIME
6 }
```

Listing 9: Verkürzte Servicedefinition des NSClient-Checks

### 7.3 Umsetzung der Systemüberwachung

Die in Kapitel 6.1 aufgelisteten Prozesse und Services können durch den NSClient-Dienst vom Nagios-Server überwacht werden. Dafür wird der in Listing 8 definierte verkürzte Befehl für *check\_nt* benutzt.

```
1 #Prozess des IIS Webservers
2 define service{
3     use                  generic-service
4     host_name            example.kit.edu
5     service_description  IIS Prozess
6     check_command        check_nt_example!PROCSTATE -l w3wp.exe
7 }
8
9 #Zeitdienst
10 define service{
11     use                  generic-service
12     host_name            example.kit.edu
13     service_description  Zeitdienst
14     check_command        check_nt_example!SERVICESTATE -l W32TIME
15 }
```

Listing 10: Prozess- und Service-Check Servicedefinitionen

Mit diesen zwei Einträgen wird der Prozess des IIS-Webservers und der Status des Dienstes zum Zeitabgleich überwacht. Andere Prozesse und Dienste lassen sich nach dem gleichen Schema überwachen.



Nach einem Neustart von Nagios werden beide Einträge im Webinterface angezeigt:

MS Prozess	OK	2009-07-22 15:14:08 3d 16h 42m 6s	1/4	w3wp.exe: Running
Zeitdienst	OK	2009-07-22 15:14:42 6d 6h 9m 6s	1/4	W32TIME: Started

Abbildung 31: Prozess- und Dienstüberwachung im Nagios-Webinterface

Die Festplattenspeicherausnutzung und die Prozessorauslastung wird auf ähnliche Weise überwacht. Hierbei muss beachtet werden, dass die Testergebnisse nicht eindeutig sind, im Gegensatz zu der Service- und Prozessüberwachung. Wann Nagios alarmieren soll muss vom Anwender in Form von Parametern festgelegt werden.

```

1 #Belegung der Partition C:
2 define service{
3     use                generic-service
4     host_name          example.kit.edu
5     service_description C:\ Drive Space
6     check_command      check_nt_example!USEDISKSPACE -l c -w 85 -c
7                       100
8 }
9 #CPU Auslastung der letzten 5 Minuten
10 define service{
11     use                generic-service
12     host_name          example.kit.edu
13     service_description CPU Load
14     check_command      check_nt_example!CPULOAD -l 5,80,100
15 }

```

Listing 11: Überwachung der Festplatten- und Prozessorauslastung

Für diese Festplattenüberwachung versendet Nagios eine Warnung, wenn der belegte Speicherplatz auf der C-Partition die 85% Marke überschreitet und meldet einen kritischen Fehler bei 100%. Bei der Prozessorüberwachung schlägt Nagios Alarm, wenn der Mittelwert der Auslastung in den letzten fünf Minuten mehr als 80% bzw. 100% betragen hat.

## 7.4 Umsetzung der Funktionlitätstest

Für die Ausführung der einfachen Funktionlitätstest aus Kapitel 6.2 werden Benutzerinformationen zur Anmeldung benötigt. Nagios besitzt extra hierfür

die Möglichkeit diese Benutzerinformationen in Variablen zu speichern, damit sie nicht einzeln bei jeder Servicedefinition verändert werden müssen. Da sich die Definition dieser Variablen in einer externen Datei befindet, können die Zugriffsrechte auf diese Datei eingeschränkt werden, wodurch die Anmelde- und Daten bei den Servicedefinitionen nicht auslesbar sind.

```

1 #Anmeldung an Oracle UCM mit lokalem Benutzerkonto
2 define service{
3     use                generic-service
4     host_name          example.kit.edu
5     service_description Anmeldung Oracle UCM als lokaler Benutzer
6     check_command       check_http!-u "/bdb/idcplg?IdcService=LOGIN&
                        Action=GetTemplatePage&Page=HOME\_PAGE&Auth=Internet" -a
                        $USER3$:$USER4$ -e "Sie sind angemeldet als" -S
7 }

```

Listing 12: Funktionalitätstest der Benutzeranmeldung

Dabei werden dem Nagios-Plugin *check\_http* mit dem „u“-Parameter die URL zur Benutzeranmeldungsseite und mit dem Parameter „a“ der Benutzername und -passwort mitgegeben. Der nach dem Parameter „e“ folgende String wird dann in der Antwort des Servers gesucht. Sollte dieser String nicht gefunden werden ist die Authentifizierung fehlgeschlagen und es wird durch Nagios eine Meldung versendet. Mit dem Parameter „S“ wird angegeben, dass eine SSL-verschlüsselte Verbindung zum Webserver über HTTPS hergestellt werden soll, ansonsten würden die Benutzerinformationen im Klartext übertragen werden, wodurch sie leicht für Angreifer auslesbar wären.

Für das Auslesen von Informationen aus der Statusseite der Oracle UCM-Anwendung wurde ein einfaches BASH-Skript entwickelt:

```

1 #!/bin/bash
2 E_BADARGS=2
3 if [ ! -n "$6" ]
4 then
5     echo "Usage: 'basename $0' -url <URL> -u <username> -p <password>"
6     exit $E_BADARGS
7 fi
8
9 DBCONNECTIONS=$(wget -qO- --user $4 --password $6 $2 | grep "System
Database")
10 DBCONNECTIONS=${DBCONNECTIONS##*>}
11 DBCONNECTIONS=${DBCONNECTIONS%% *}
12 echo $DBCONNECTIONS

```

Listing 13: Auslesen der Verbindungen zur Datenbank

Dabei muss als URL die Seite mit den Datenbankverbindungen und gültige Benutzerinformationen mitgegeben werden. Anschließend wird die aufgerufene Seite nach der gewünschten Informationen untersucht und ausgegeben. Dieses einfache Script kann auch dazu verwendet um andere Informationen von der Statusseite abzufragen.

## 7.5 Auswertung der Logdateien

Um die drei genannten Logdateien aus Kapitel 6.3 auszuwerten wird durch den NRPE-Dienst das Plugin *check\_logfiles* von Gerhard Laußer eingesetzt. Dieses Plugin besitzt bereits einige nützliche Funktionen für die Überwachung von Logdateien. Durch das Setzen eines Zeitstempels filtert das Plugin veraltete Einträge heraus und untersucht nur neu hinzugekommene Zeilen. Der Rotationsalgorithmus der Oracle UCM-Logdateien kann dem Plugin durch die Verwendung einer Konfigurationsdatei mitgeteilt werden.

Die Konfigurationsdatei für das *check\_logfiles*-Plugin wird im folgendem Listing gezeigt:

```
1 @searches = ({
2   tag => 'ucmlogs',
3   type => 'rotating::uniform',
4   logfile => 'D:/bdb2/weblayout/groups/secure/logs/bdb/IdcLnLog.htm',
5   rotation => 'refinery\d{2}\.htm',
6   warningpatterns => [
7     'Cannot identify file',
8     'Bad CRC value in IHDR chunk',
9     'Der Dateiname darf nicht länger als 80 Zeichen sein',
10  ],
11 });
```

Listing 14: Konfigurationsdatei für *check\_logfiles*

Mit dem „tag“-Attribut wird diese Auswertung eindeutig identifizierbar gemacht, da man in der gleichen Konfigurationsdatei mehrere Logdateien bzw. weitere Durchsuchungen definieren kann. Das Attribut „type“ muss hier so gesetzt werden, da die aktuelle Logdatei und die wegrotierte Logdatei das gleichen Namensschema benutzen. Der Pfad zu den Logdateien wird über das Attribut „logfile“ gesetzt. Da die einzelnen Logdateien mit einem be-

stimmten Namensmuster erstellt werden (siehe Kapitel 6.3), muss dieses Namensmuster hier direkt angegeben werden. In diesem Falle durchsucht das *check\_logfiles*-Plugin alle Logdateien mit dem Namen *refinery00.htm* bis *refinery99.htm*. Alle gefundenen Dateien werden dem Datum nach sortiert und die aktuellste Datei wird untersucht. Nach welchen Stopwörtern gesucht werden soll wird mit dem Attribut „warningpatterns“ angegeben, sofern mindestens einer dieser Strings gefunden wurde liefert das Plugin ein WARNING als Rückmeldung inklusive der Zeile in dem das Stopwort gefunden wurde.

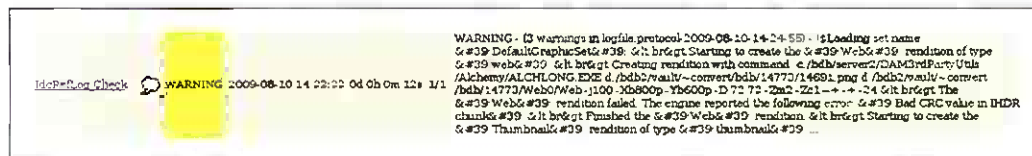


Abbildung 32: Ausgabe der betreffenden Zeile in der Logdatei

## 7.6 Benutzersimulation

Um die Funktionalität der Anwendung eindeutig festzustellen werden typische Benutzeraktionen simuliert und die Ergebnisse an Nagios übermittelt. Solche typische Aktionen sind das Einchecken eines Bildes, Suche nach einem Bild und schließlich die Anforderung des Originalbildes und der konvertierten Bilder. Da Nagios standardmäßig ein Plugin periodisch jede fünf Minuten aufruft, würde sich die Festplatte und die Datenbank des Oracle UCM-Servers im Laufe der Zeit an ihre Kapazitäten stoßen. Daher werden nach der Anforderung und Überprüfung der Bilder alle Testbilder vom Server entfernt. Per Webservice soll ein Testbild an den Server geschickt und eingchecked werden. Der Ablauf der Benutzersimulation soll in verkürzter Form durch das Struktogramm 33 verdeutlicht werden.

In diesem ersten Schritt wird auch gleichzeitig die Erreichbarkeit der Anwendung über das Netzwerk getestet.

Wenn die Übertragung des Bildes erfolgreich wird anschließend nach dem

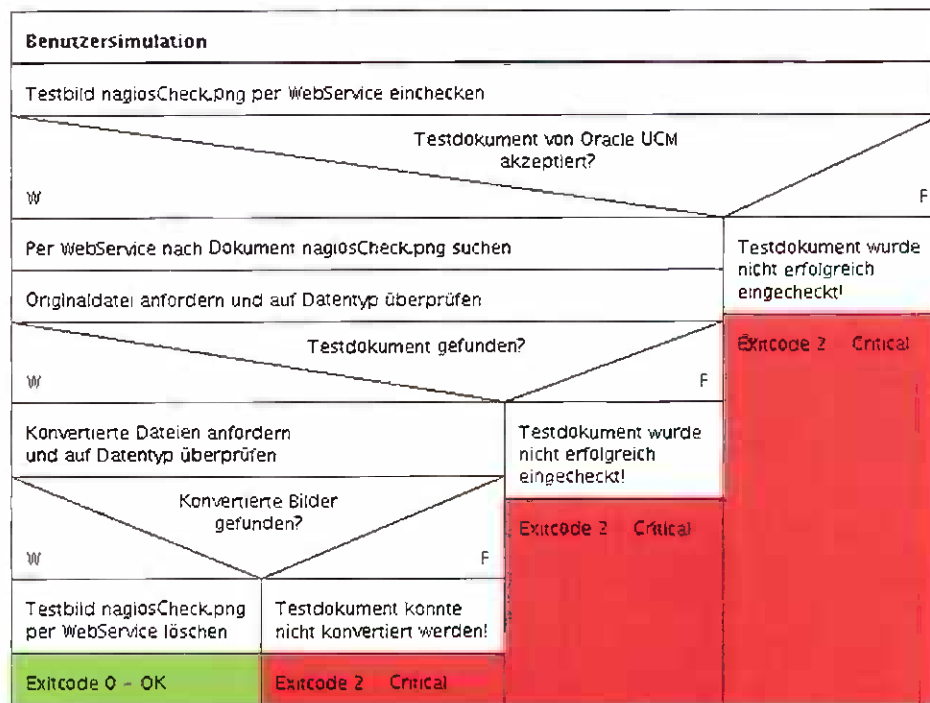


Abbildung 33: Geplanter Ablauf der Benutzersimulation

soeben eingecheckten Bild per Dateinamen gesucht um die Funktionalität der Indizierung zu kontrollieren.

Sollte das Bild gefunden werden, wird es vom Nagios-Server angefordert und auf seine Korrektheit überprüft. Der gleiche Test wird mit den konvertierten Bildversionen durchgeführt, um die Funktion der Konvertierung zu überwachen.

Falls alle Tests erfolgreich waren, wird das Testbild und alle konvertierten Bilder vom Oracle UCM-Server gelöscht. Bei den anderen Szenarien gibt das Plugin den Wert 2 für den Status „CRITICAL“ zurück.

Die Realisierung dieser Simulation wird durch zwei Plugins realisiert.

**Einchecken eines Testbildes** Das erste Plugin dient zum Einchecken des Testbildes. Dabei ruft der Nagios-Server ein auf PHP-basierendes Script auf. In diesem Script wird die PHP-Klasse *nuSOAP* eingebunden, damit man

vereinfacht auf Web Services zugreifen kann. Die Kommunikation zwischen Client und Server bei der Benutzung eines Web Services findet, wie im Kapitel 3.5 beschrieben, im XML-Format statt. Um den Aufwand zu vermeiden diese XML-Datei immer selbst zu erstellen, wird mit Hilfe der WSDL-Datei auf dem Oracle UCM-Server die benötigten Parameter beim Aufruf eines Web Services von *nuSOAP* ausgelesen.

In der folgenden Abbildung werden aus der WSDL-Datei alle möglichen Anforderungsparameter für den Web Service *CheckInUniversal* gezeigt.

```
<s:element name="CheckInUniversal">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="dDocName" type="s:string"/>
      <s:element minOccurs="0" maxOccurs="1" name="dDocTitle" type="s:string"/>
      <s:element minOccurs="0" maxOccurs="1" name="dDocType" type="s:string"/>
      <s:element minOccurs="0" maxOccurs="1" name="dDocAuthor" type="s:string"/>
      <s:element minOccurs="0" maxOccurs="1" name="dSecurityGroup" type="s:string"/>
      <s:element minOccurs="0" maxOccurs="1" name="dDocAccount" type="s:string"/>
      <s:element minOccurs="0" maxOccurs="1" name="CustomDocMetaData" type="s0:IdcPropertyList"/>
      <s:element minOccurs="0" maxOccurs="1" name="primaryFile" type="s0:IdcFile"/>
      <s:element minOccurs="0" maxOccurs="1" name="alternateFile" type="s0:IdcFile"/>
      <s:element minOccurs="0" maxOccurs="1" name="extraProps" type="s0:IdcPropertyList"/>
    </s:sequence>
  </s:complexType>
</s:element>
```

Abbildung 34: Anforderungsparameter für CheckInUniversal aus der WSDL-Datei

Im PHP-Script werden nach dem Einlesen dieser WSDL-Datei und der Authentifizierung am Oracle UCM-Server die benötigten Parameter beim Aufruf des Web Services gesetzt und die Ausgabe des Servers ausgewertet.

```
1 $soap = new soapclient($WSDL-URL, //WSDL-Datei einlesen
2 array('login' => $user, 'password' => $password)); //Authentifizierung am
   Oracle UCM-Server
3
4 //Aufruf des Web Services
5 $ergebnis = $soap->CheckInUniversal(array(
6   'dDocAuthor'=>$user, //Autor des Bildes
7   'dDocTitle'=>'testBild4nagios', //Titel des Bildes
8   'dSecurityGroup'=>'private', //Sichtbarkeit des Bildes
9   'dDocAccount'=>'NAGIOS/TEST', //Angabe einer Gruppe
10  'dInDate'=>date("d.m.y H:i"), //Aktuelles Datum
11  'dDocType'=>'Picture', //Dokumententyp
12  'doFileCopy'=>'1', //Datei nur kopieren, nicht verschieben
13  'dDocFormat'=>'image/png', //MIME-Type
14  'primaryFile'=>array(
15    'fileName'=>'testBild4nagios',
16    'fileContent'=>$content) //Byteweise eingelesenes Bild
17 ));
```

```

18 [...]
19 //Auswertung der Antwort des Servers
20 if (ereg(' erfolgreich eingecheckt.', $output)) {
21     echo('CHECKIN OK - '.$output);
22     die(0); //Einchecken erfolgreich
23 } else {
24     echo('CHECKIN CRITICAL - '.$output);
25     die(2); //Einchecken fehlgeschlagen
26 }

```

Listing 15: Aufruf des Web Services CheckInUniversal

Die Bilddatei muss für die Übertragung über HTTP zuerst byteweise eingelesen werden und anschließend mit dem Base64-Algorithmus kodiert werden.

Dabei übernimmt die nuSOAP-Klasse die Base64-Enkodierung.

Der Ablauf dieses Plugins soll durch folgende Abbildung verdeutlicht werden.

zur Abwechslung:-

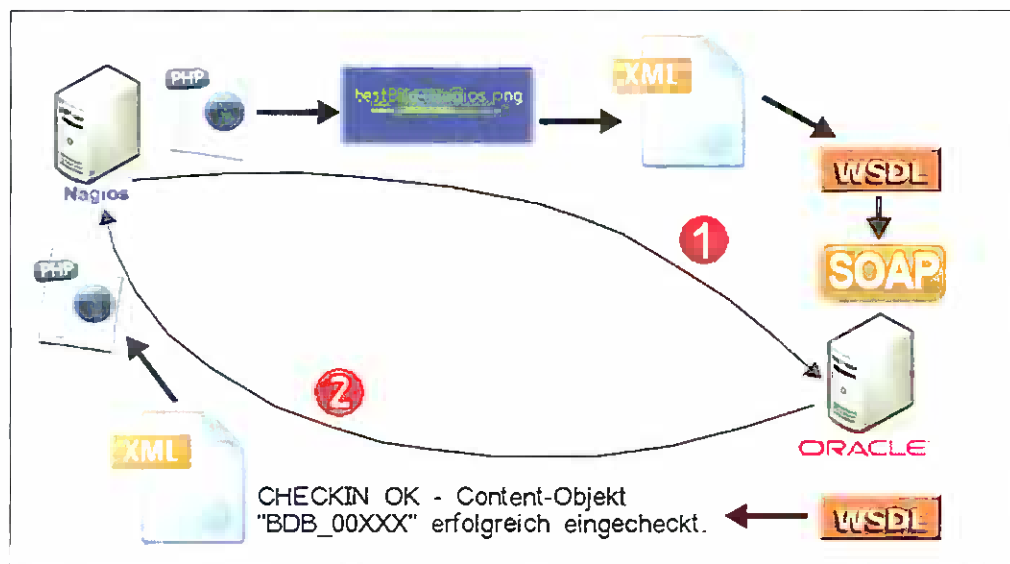


Abbildung 35: Einchecken eines Testbildes

1. Im ersten Schritt wird das PHP-Skript vom Nagios-Server aufgerufen und liest das Testbild ein. Anschließend verwendet es die WSDL-Datei des Web Services um das entsprechende XML-Dokument zu erstellen. Diese XML-Datei wird anhand der nuSOAP-Klasse SOAP-konform an den Oracle UCM-Servers gesendet.



2. Die XML-basierende Rückantwort des Servers wird auch anhand der WSDL-Datei erstellt und kann vom PHP-Script ausgewertet werden.

**Validierung der Indizierung und Konvertierung** Der zweite Teil der Benutzersimulation überprüft, ob das Testbild erfolgreich indiziert und konvertiert wurde. Dabei verwendet es die gleichen Grundfunktionen wie das erste Plugin. Jedoch wird anstatt dem Web Service *CheckInUniversal* die WSDL-Datei des Web Services *AdvancedSearch* verwendet und aufgerufen.

```
1 $ergebnis = $soap->AdvancedSearch(  
2     'queryText'=>"dDocTitle <substring> 'testBild4Nagios"  
3 );
```

Listing 16: Überprüfen der Indizierung anhand einer Suchanfrage

Durch diesen Aufruf wird anhand seines Titels nach einem zuvor eingetragtem Testbild gesucht. Dadurch kann überprüft werden, ob das Bild korrekt vom Server angenommen und indiziert wurde. In der Rückantwort des Servers befindet sich unter anderem die eindeutige Identifikationsnummer des Testbildes. Diese Nummer wird für die Validierung der Konvertierung verwendet.

```
1 //Test des Originalbildes  
2 $ergebnisGet = $soap->GetFileByID('dID'=>$dID);  
3 [...]  
4 if(!mb_ereg('PNG', $outputGetOrig))  
5 {  
6     echo('SEARCH CRITICAL - Originalbild ist nicht im PNG Format!');  
7     die(2); //Originalbild korrupt  
8 }  
9 //Test der Thumbnailversion des Testbildes  
10 $ergebnisGetThumbnail = $soap->GetFileByID(array('dID'=>$dID, 'rendition' =>  
    'Thumbnail'));  
11 [...]  
12 if(!mb_ereg('JFIF', $outputGetThumbnail))  
13 {  
14     echo('SEARCH CRITICAL - Thumbnailversion des Testbildes ist nicht im  
        JPEG Format!');  
15     die(2); //Thumbnailversion korrupt  
16 }  
17 [...] //Überprüfung der anderen konvertierten Bilder
```

Listing 17: Überprüfen der Indizierung anhand einer Suchanfrage

Sofern das Bild gefunden wurde, wird es vom Plugin anschließend angefordert und nach dem Dateityp untersucht. Die Konvertierung wird dadurch über-

prüft indem die konvertierten Versionen des Testbildes angefordert werden und wie das Originalbild auf einen gültigen Dateityp getestet werden.

Der Ablauf dieses Plugins ist dem ersten sehr ähnlich, <sup>wie</sup> siehe Abbildung 36 <sup>zeigt</sup>.

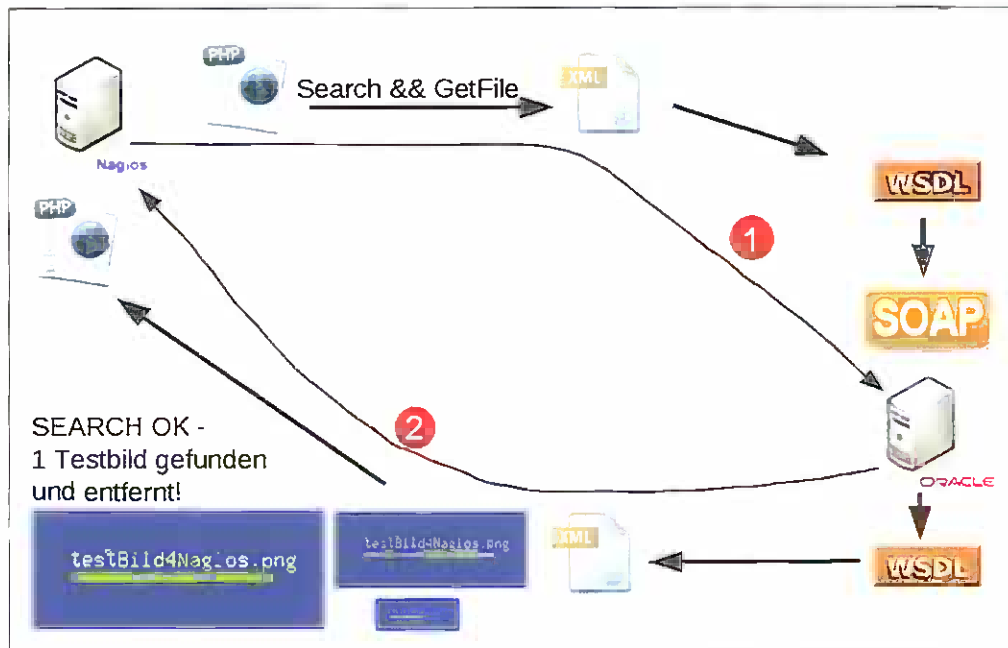


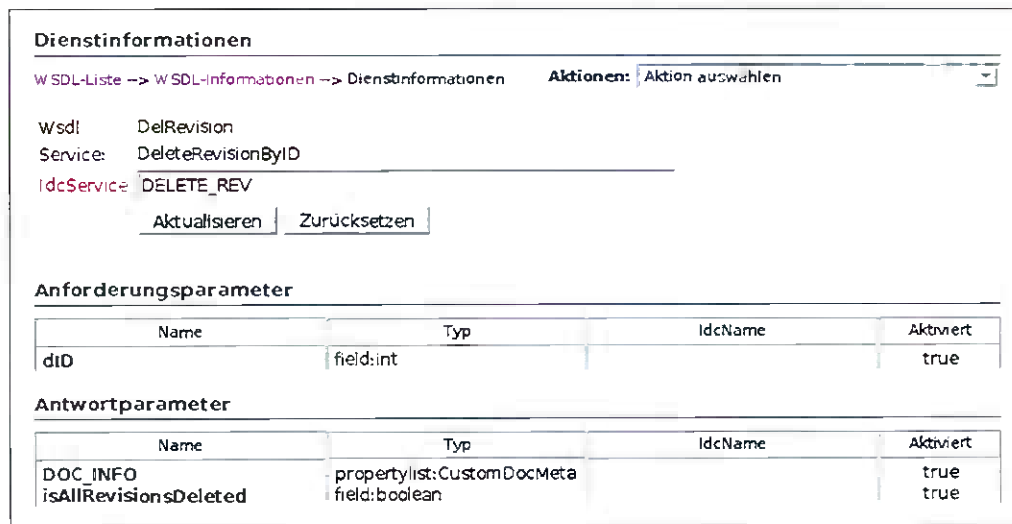
Abbildung 36: Validierung der Indizierung und Konvertierung

1. Zuerst wird die WSDL-Datei für den Web Service *AdvancedSearch* eingelesen. Eine Suchanfrage nach dem Testbild wird wieder über eine XML-Datei an den Server gesendet. Wenn eine Datei gefunden wurde, wird das Originalbild und die konvertierten Versionen anhand der Identifikationsnummer angefordert.
2. Diese Bilder werden wieder byteweise innerhalb der XML-Datei der Serverantwort an den Nagios-Server übertragen und auf ihre Korrektheit überprüft. Sollten alle bisherigen Tests ohne Probleme abgelaufen sein, wird das Testbild und die konvertierten Bilder vom Server entfernt.

Damit nicht unnötiger Festplattenplatz durch die Testbilder verschwendet wird, sollen die Testbilder mit den konvertierten Version vom Server gelöscht werden.

Hier frage ich mich zum wiederholten Male "wie"? Hat ich was überlesen?

Standardmäßig bietet Oracle UCM keinen Web Service zum Löschen von Dokumenten an. Daher muss zunächst eine WSDL-Datei dafür erstellt werden, siehe Abbildung 37.



**Dienstinformationen**

WSDL-Liste -> WSDL-Informationen -> Dienstinformationen    Aktionen:

WSDL: DelRevision  
Service: DeleteRevisionByID  
IdCService: DELETE\_REV

**Anforderungsparameter**

Name	Typ	IdCName	Aktiviert
diID	field:int		true

**Antwortparameter**

Name	Typ	IdCName	Aktiviert
DOC_INFO	propertylist:CustomDocMeta		true
isAllRevisionsDeleted	field:boolean		true

Abbildung 37: Anlegen eines eigenen Web Services

Der Name der WSDL-Datei und des Web Services kann beliebig gewählt werden, solange als Service die entsprechende interne Bezeichnung zum Löschen von Dokumenten „DELETE\_REV“ verwendet wird.<sup>31</sup> Um Zweideutigkeiten zu vermeiden, wird als Anforderungsparameter die eindeutige Identifikationsnummer verwendet.

Der eigene Web Service wird wie die anderen im Anschluss aufgerufen:

```
1 $ergebnisDelete = $soap->DeleteRevisionByID('diID'=>$diID);
```

Listing 18: Aufruf des eigenen Web Services

Die erstellten PHP-Dateien müssen noch Nagios als Befehl hinzugefügt werden.

```
1 #Einchecken eines Testbildes
2 define command{
3     command_name    check_ucm_checkin
4     command_line    /usr/lib/nagios/plugins/check_ucm/nagiosCheckin.php
5 }
6 #Validierung der Indizierung und Konvertierung
7 define command{
8     command_name    check_ucm_search
```

<sup>31</sup>Quelle: [Huff06] S. 379

```
9      command_line    /usr/lib/nagios/plugins/check_ucm/nagiosSearch.php
10    }
```

Listing 19: Befehlsdefinitionen der Benutzersimulation

Die Aufteilung der Benutzersimulation in zwei Nagios-Kommandos sorgt dafür, dass es keine Garantie für die Reihenfolge der Ausführung der Plugins gibt. Aufgrund dieser Asynchronität und dem Umstand, dass die Indizierung und Konvertierung des Testbildes abhängig von der Auslastung des Oracle UCM-Servers ist, wird die Anzahl der Ausführungen des Plugins für den Wechsel von Soft zum Hard State erhöht.

```
1 define service{
2     use                generic-service
3     host_name          example.kit.edu
4     service_description Oracle UCM Search Delete
5     max_check_attempts 8
6     check_command       check_ucm_search
7 }
```

Listing 20: Angepasste Servicedefinition für die Benutzersimulation

## 8 Ergebnis

In ~~Die in dieser Arbeit erbrachte Lösung realisiert/ermöglicht die Überwachung~~ wurde eine Nagios Umgebung realisiert, die eine ~~umfassende Überwachung von Oracle VCIT ermöglicht~~ ~~der in Kapitel 6 genannten Elemente. Das Produkt dieser Lösung lässt sich~~ ~~Ergebnis~~ im Webinterface von Nagios betrachten, siehe Abbildung 38.

Host ↑	Service ↑	Status	Last Check	Next Check	Status Information
blddatenbank.ku.fhn.ch	BDB logical user	OK	2009-07-22 15:17:48	2009-07-22 15:17:48	HTTP OK: HTTP/1.1 200 OK - 35543 bytes in 0.01 seconds
	C:\ Drive Space	OK	2009-07-22 15:16:19	2009-07-22 15:16:19	c - total: 19.90 Gb - used: 16.31 Gb (82%) - free: 3.67 Gb (18%)
	CPU Load	OK	2009-07-22 15:13:40	2009-07-22 15:13:40	CPU Load 3% (5 min average)
	CPU Load active	OK	2009-07-22 15:18:20	2009-07-22 15:18:20	CPU OK - CPU0 = 0%
	D:\ Drive Space	OK	2009-07-22 15:17:07	2009-07-22 15:17:07	d - total: 20.00 Gb - used: 9.36 Gb (47%) - free: 10.64 Gb (53%)
	HTTP	OK	2009-07-22 15:17:07	2009-07-22 15:17:07	HTTP OK: HTTP/1.1 200 OK - 35543 bytes in 0.008 seconds
	IIS Admin Service	OK	2009-07-22 15:17:30	2009-07-22 15:17:30	IIS Admin Service MIB Admin Started
	IIS Admin Service MIB	OK	2009-07-22 15:14:14	2009-07-22 15:14:14	IIS Admin Service MIB Started
	IIS Refinery Service MIB	OK	2009-07-22 15:16:59	2009-07-22 15:16:59	IIS Refinery Service MIB Started
	IIS Admin Service	OK	2009-07-22 15:16:47	2009-07-22 15:16:47	IISADMIN Started
	IIS Process	OK	2009-07-22 15:17:40	2009-07-22 15:17:40	IIS Process Running
	IIS AdminNT	OK	2009-07-22 15:16:20	2009-07-22 15:16:20	IIS AdminNT Running
	IIS Refinery Check	OK	2009-07-22 15:17:32	2009-07-22 15:17:32	OK - no errors or warnings
	IIS RefineryNT	OK	2009-07-22 15:17:36	2009-07-22 15:17:36	IIS RefineryNT Running
	Memory Usage	OK	2009-07-22 15:14:27	2009-07-22 15:14:27	Memory usage: total 31 MB - used 1339.87 MB (43%) - free: 305.64 MB (37%)
	NSClient++ Version	OK	2009-07-22 15:14:35	2009-07-22 15:14:35	NSClient++ 0.3.6.810 2009-06-14
	Oracle UCM Check	OK	2009-07-22 15:14:40	2009-07-22 15:14:40	CHECKIN OK - Content-Object "BDB_005201" erfolgreich empfangen
	Oracle UCM Search Delete	OK	2009-07-22 15:15:56	2009-07-22 15:15:56	SEARCH OK - 1 Testbild gefunden und entfernt!
	Oracle UCM Session	OK	2009-07-22 15:15:01	2009-07-22 15:15:01	Oracle OK - result 20 match none
	Oracle ServiceNT	OK	2009-07-22 15:14:30	2009-07-22 15:14:30	OracleServiceNT Started
	Oracle XEINS Listener	OK	2009-07-22 15:15:33	2009-07-22 15:15:33	OracleXEINS Listener Started
	PING	OK	2009-07-22 15:17:46	2009-07-22 15:17:46	PING OK - Packet loss = 0% RTT = 0.45 ms
	Symantec AntiVirus	OK	2009-07-22 15:17:44	2009-07-22 15:17:44	Symantec AntiVirus Started
	TNS Listener	OK	2009-07-22 15:14:43	2009-07-22 15:14:43	OK (10 ms)
	Windows Uptime	OK	2009-07-22 15:17:41	2009-07-22 15:17:41	System Uptime - 35 day(s) 2 hour(s) 21 minute(s)
	Zentao	OK	2009-07-22 15:14:42	2009-07-22 15:14:42	WGETIME Started
	pagefile active	OK	2009-07-22 15:14:32	2009-07-22 15:14:32	Page File Utilization OK - D:\pagefile.sys = 52%
	uptime active	OK	2009-07-22 15:13:46	2009-07-22 15:13:46	Operating System OK - Microsoft Windows(R) Server 2003 Standard Edition Service Pack 2

Abbildung 38: Webinterface von Nagios

Die einzelnen Elemente der Überwachung lassen sich mit Ausnahme der Benutzersimulation durch Konfiguration von bereits vorhandenen Nagios-Plugins überprüfen. Bei dieser Konfiguration müssen die bereits im Kapitel 3.1 erörterten Gesichtspunkte wie Netzwerkabhängigkeiten, -belastung und sicherheitstechnische Aspekte bedacht werden. Gerade bei der Verwendung von Benutzerinformationen zur Authentifizierung ist es notwendig zu überprüfen, ob die Informationen als Klartext oder verschlüsselt übertragen werden.

Die durch die einzelnen Plugins gesammelten Informationen lassen sich zusätzlich im Detail aufrufen:

Service State Information	
Current Status:	<b>OK</b> (for 2d 17h 41m 56s)
Status Information:	CPU Load 2% (5 min average)
Performance Data:	'5 min avg Load'=2%;80;100;0;100
Current Attempt:	1/4 (HARD state)
Last Check Time:	2009-08-11 15:17:31
Check Type:	ACTIVE
Check Latency / Duration:	0.180 / 0.021 seconds
Next Scheduled Check:	2009-08-11 15:22:31
Last State Change:	2009-08-08 21:37:31
Last Notification:	N/A (notification 0)
Is This Service Flapping?	<b>NO</b> (0.00% state change)
In Scheduled Downtime?	<b>NO</b>
Last Update:	2009-08-11 15:19:20 ( 0d 0h 0m 7s ago)

Abbildung 39: Details der Prozessorauslastung

## 9 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde eine Lösung entwickelt um mit der Open Source-Überwachungssoftware Nagios den Betrieb des im Forschungszentrum Karlsruhe verwendeten Dokumenten-Management-Systems Oracle UCM zu überwachen. Eine solche Überwachung ist notwendig um den Mitarbeitern des Forschungszentrums Karlsruhe einen möglichst zuverlässigen Dienst anbieten zu können. Dabei sollte die Überwachung proaktiv auf mögliche Fehlzustände testen und bei einer Störung eine Alarmmeldung an die verantwortlichen Kontaktpersonen versenden. Das Überwachungssystem sorgt dafür, dass jeder Fehler sofort gemeldet wird, damit die Problemquellen gefunden und eventuell behoben werden können, bevor die Endbenutzer Störungen bei der Nutzung des Dienstes bemerken.

Für die Bearbeitung der Aufgabe war es notwendig sich mit den Grundlagen von Überwachungssystemen auseinander zusetzen. Darunter fielen die Punkte Netzwerkstruktur, -abhängigkeit und verschiedene Sicherheitsaspekte die beim Einsatz einer Überwachungssoftware eine Rolle spielen. Um die eigentliche Funktions- und Arbeitsweise eines Dokumenten-Management-Systems zu verstehen wurde die grundsätzliche Art eines Dokumentes im Vergleich zu Daten betrachtet. Auf diesem Wissen aufbauend konnten die Aufgabenbereiche Eingabe, Verwaltung, Archivierung und Ausgabe eines Dokumenten-Management-Systems untersucht und Vergleiche zu Content-Management-Systemen gezogen werden.

Für die Umsetzung wurde die Service-orientierte Architektur der Oracle UCM-Anwendung in Verbindung mit Web-Services verwendet. Hierfür war es notwendig sich mit Grundprinzipien dieser Architekturen, deren Funktionsweise und verwendete Elemente vertraut zu machen. Dadurch konnte später korrekt auf die benötigten Funktionen zugegriffen werden.



Im Forschungszentrum Karlsruhe wird als Überwachungssoftware das Open Source-Programm Nagios für die Überwachung von Netzwerken, Server und Dienste verwendet. Damit Fehler korrekt von Nagios erkannt werden, bestand die Notwendigkeit die Funktionsweise und den Aufbau dieser Software zu studieren. Das Einholen von Informationen zur Auswertung wird durch Plugins ermöglicht. Das Verständnis über die Struktur und Richtlinien dieser Plugins wurde benötigt um später eigene zu entwickeln und sie effektiv zu verwenden. Dabei galt es die speziellen Funktionen von Nagios wie die Hard und Soft States oder das Flapping von Zustände bei der spätere Verwendung zu berücksichtigen. Über die verschiedenen Möglichkeiten die benötigten Informationen zu sammeln wurde ein kurzer Überblick gegeben.

Oracle UCM wird im Forschungszentrum Karlsruhe für die Verwaltung von Webseiten, Dokumenten und Bilder eingesetzt. Für die Ermittlung der Überwachungselemente wurde der allgemeine interne Aufbau und die Arbeitsweise dieser Anwendung *in allen Einsatzfällen* untersucht. *Für diese Arbeit wurde der* ~~Der~~ konkrete Einsatz von Oracle UCM ~~wurde~~ als Bilddatenbank verwendet. Die dabei auftretenden typischen Benutzerinteraktionen wurden für die später folgende Benutzersimulation verwendet. Die einzelnen Überwachungselemente wurden in die Ebenen Statusabfragen, Funktionalitätstest, Auswerten von Logdateien und Benutzersimulation unterteilt. Dabei führte die Abhängigkeit der Elemente zueinander zu der Einordnung in die verschiedenen Ebenen. Unter den Statusabfragen befinden sich einfache Test wie ein Ping, Arbeitsspeicherauslastung oder der Zustand eines Prozesses. Bei den Funktionalitätstest werden Anwendungen verwendet und die Antwort ausgewertet wie beispielsweise eine Anmeldung an Webserver mit Benutzerdaten. Die Benutzersimulation beinhaltet verschiedene Benutzeraktionen und überprüft, ob die Anwendung noch alle Funktionen erfüllt. Diese Einteilung in die verschiedenen Überwachungsebenen gibt den Verantwortlichen einen besseren Überblick über die Fehlersituation, so dass Fehlerquellen

schneller entdeckt werden können.

Für die Umsetzung wurde ein Testsystem aufgesetzt, das aus einer separaten Nagios-Installation zum Testen der Überwachung und einer virtuellen Maschine, als Klon der Bilddatenbank zum Simulieren der einzelnen Fehlzustände, bestand. Da es sich bei der zu überwachenden Bilddatenbank um einen Windows-Server handelte, wurde ein passender Nagios-Agent ausgewählt und dessen Installation und Konfiguration erläutert. Durch den Einsatz dieses Agenten konnten die verschiedenen Ebenen der Überwachung durch Verwendung von verschiedenen Überwachungsmethoden realisiert werden. Dabei wurde für die Benutzersimulation eigene Plugins entwickelt, die die Benutzeraktionen per Web Service ausführen. Das Plugin testet mit dem Hinzufügen eines Testbildes die Erreichbarkeit und einen Teil der Funktionalität des Oracle UCM-Servers. Durch eine Suchanfrage wird die Indizierung überprüft. Die Konvertierung wird anhand der angeforderten Testbilder validiert. Dabei wurden weitere Web Services verwendet. Die Konsequenzen einer automatischen Benutzersimulation mussten beachtet werden. Durch die ständige Ausführung der Benutzersimulation würden die Ressourcen des Servers wie der Festplattenspeicherplatz an ihre Kapazitäten stoßen.

~~Das Problem konnte durch das Löschen des Testbildes und die konvertierten Versionen gelöst werden. Standardmäßig gibt es keinen Web Service zum Löschen von Dokumenten. Daher wurde ein eigener Web Service im Oracle UCM angelegt.~~

~~Die Überwachungsdienste wurden in das Webinterface von Nagios aufgenommen. Dadurch konnte die Überwachung aller zuvor ermittelten Elemente realisiert, die Informationen der einzelnen Plugins von Nagios ausgewertet und im Webinterface angezeigt werden. Die korrekte Benachrichtigung über Störungen konnte durch die Simulation der Fehlzustände in der virtuellen Maschine sichergestellt werden.~~

Bei einer Überwachung ist es es notwendig zuvor verschiedene Schwellwer-

Das kam bisher zu kurz.  
Was, wie hast du die Prüfung gemacht?  
=> Kap. 8.

Siehe zusammenfassen

Damit das Problem

konnten, musste ein

und erlauben dem Administ einen schnellen Überblick über die korrekte Funktion der auftretende Probleme dieses Dienstes

te zu setzen. An diesen Werten kann die Überwachungssoftware festlegen, ob ein Objekt einen kritischen Zustand erreicht hat oder nicht. Für die Ermittlung dieser Größen müssen die Werte der Überwachungselemente über einen längeren Zeitraum beobachtet und analysiert werden. Aufgrund des begrenzten zeitlichen Rahmens dieser Arbeit konnte dies nicht vollständig umgesetzt werden, so dass es während dem Betrieb fortgesetzt werden muss. Hierunter fallen vor allem spezifische Merkmale eines bestimmten Servers. Eine ungewöhnlich hohe Prozessorauslastung, die durch eine zeitlich gesteuerte Sicherung entstehen kann, sollte von der Überwachungssoftware nicht als Fehlverhalten interpretiert werden. Auch die Liste der Stopwörter für die Auswertung der Logdateien muss für neue bisher unbekannte Fehler immer wieder erweitert werden.

Das entwickelte Plugin für die Benutzersimulation kann auch mit zusätzliche Funktionen versehen werden. Durch die Verwendung von anderen Web Services können weitere Funktionalitäten überprüft werden. Dabei kann die Benutzersimulation auch auf anderen Dokumenten-Management-Systemen eingesetzt werden. Hierfür müsste nur anstatt eines Testbildes beispielsweise eine PDF-Testdatei oder Word-Dokument verwendet werden. Für die Validierung der Indizierung und Konvertierung müsste das Plugin nur leicht angepasst werden.

Das Plugin kann leicht angepasst werden um  
Indizierung und Konvertierung eines ~~ein~~ <sup>zu testen</sup> ~~den~~ <sup>den Check-J4,</sup>  
PDF-Testdatei oder Word-Dokument verwendet werden. Für die Validierung  
der Indizierung und Konvertierung müsste das Plugin nur leicht angepasst  
werden.



## Glossar

Bezeichnung	Beschreibung	Seiten
CI	Coded Information	11, 14, 16
CMS	Content-Management-System	16–19
DMS	Dokumenten-Management-System	1, 9, 13–15, 17
DNS	Domain Name System	35
DoS	Denial of Service	9
ECM	Enterprise-Content-Management	40
FQDN	Fully Qualified Domain Name	54
FTP	File Transfer Protokoll	33
HTTP	Hypertext Transfer Protocol	21, 22, 64
IIS	Internet Information Service	46, 57
IP	Identifikation Protokoll	6, 9, 34, 49, 51, 52, 54
LDAP	Lightweight Directory Access Protocol	35



---

Bezeichnung	Beschreibung	Seiten
MIB	Management Information Base	35, 36, 50
NCI	None-Coded Information	10, 11, 14, 16
NRPE	Nagios Remote Plugin Executor	34, 50, 53, 54, 56, 60
NSCA	Nagios Service Check Acceptor	36, 51
OCR	Optical Character Recognition	14
OracleUCM	Content-Management-System von Oracle	3, 38, 40, 41, 48, 61-64, 67, 68, 71-73
PDF	Portable Document Format	1, 14, 40, 74
PHP	PHP: Hypertext Preprocessor	62, 64, 65
RPC	Remote Procedure Call	55



---

Bezeichnung	Beschreibung	Seiten
SNMP	Simple Network Management Protocol	35, 36, 50, 51, 53
SOA	Service-Oriented Architecture	19–21, 39
SOAP	Simple Object Access Protocol	21, 22, 24, 64
SSH	Secure Shell	33, 34, 49, 50
SSL	Secure Sockets Layer	56, 59
UDDI	Universal Description, Discovery and Integration protocol	23, 24
W3C	World Wide Web Consortium	21
WSDL	Web Services Description Language	21, 23, 24, 63– 67
XML	Extensible Markup Language	21–23, 63–66



## Abbildungsverzeichnis

1	Zusätzliche Netzwerkabhängigkeit und Netzwerkbelastung . . .	7
2	Anteil an strukturierten Informationen . . . . .	11
3	Aufgabenbereiche eines Dokumenten-Management-Systems . .	13
4	Sichtweise CMS gegenüber DMS . . . . .	17
5	„any-to-any“ Content-Management Konzept . . . . .	18
6	Simple Software Architektur eines Webshops . . . . .	20
7	Hinzugefügte Service-orientierte Komponente . . . . .	20
8	Kommunikationsprotokoll SOAP . . . . .	22
9	Ablauf einer Web Service-Benutzung . . . . .	23
10	Plugins als separate Komponente . . . . .	26
11	Anzeige des Servers im Webinterface von Nagios . . . . .	27
12	Beispielhafte manuelle Ausführung eines Servicechecks . . . .	28
13	Beispiel für den zeitlichen Verlauf durch vers. Zustände . . . .	30
14	Verlauf von sich schnell wechselnden Zuständen . . . . .	31
15	Verschiedene Überwachungsmöglichkeiten von Nagios . . . . .	32
16	Ausführung eines netzwerkbasierenden Servicechecks . . . . .	33
17	Manuelle Ausführung eines Servicechecks über SSH . . . . .	34
18	Aktive Checks mit NRPE . . . . .	34
19	Struktur der Management Information Base . . . . .	36
20	Passive Checks mit NSCA . . . . .	37
21	Oracle UCM Architektur . . . . .	38
22	Beispielhafter Einsatz eines Content Servers . . . . .	39
23	Bilddatenbank als Anwendung . . . . .	41
24	Überwachungselemente . . . . .	47
25	Abfrage von Windows-Ressourcen durch <i>check_nt</i> . . . . .	51
26	Zugriff auf den NSClient-Dienst durch <i>check_nt</i> . . . . .	52
27	Konfiguration des NSClient++ während der Installation . . . .	54





---

28	Kommunikation zwischen Nagios und NSClient++ . . . . .	54
29	Aufruf eines NRPE-Kommandos . . . . .	56
30	Aufruf des NSClient-Dienstes . . . . .	56
31	Prozess- und Dienstüberwachung im Nagios-Webinterface . . .	58
32	Ausgabe der betreffenden Zeile in der Logdatei . . . . .	61
33	Geplanter Ablauf der Benutzersimulation . . . . .	62
34	Anforderungsparameter für CheckInUniversal aus der WSDL- Datei . . . . .	63
35	Einchecken eines Testbildes . . . . .	64
36	Validierung der Indizierung und Konvertierung . . . . .	66
37	Anlegen eines eigenen Web Services . . . . .	67
38	Webinterface von Nagios . . . . .	69
39	Details der Prozessorauslastung . . . . .	70



## Codelistingverzeichnis

1	Nagiosschema für Objektdefinitionen . . . . .	27
2	Definition eines Hostobjektes . . . . .	27
3	Verkürzte Definition eines Hostobjektes . . . . .	27
4	Verkürzte Definition eines Hostobjektes . . . . .	28
5	Beispielhafte Definition eines Servicechecks . . . . .	29
6	NSClient++ Konfigurationsdatei . . . . .	55
7	Servicedefinition des NSClient-Checks . . . . .	56
8	Server spezifische Befehlsdefinition . . . . .	57
9	Verkürzte Servicedefinition des NSClient-Checks . . . . .	57
10	Prozess- und Service-Check Servicedefinitionen . . . . .	57
11	Überwachung der Festplatten- und Prozessorauslastung . . . . .	58
12	Funktionalitätstest der Benutzeranmeldung . . . . .	59
13	Auslesen der Verbindungen zur Datenbank . . . . .	59
14	Konfigurationsdatei für <i>check_logfiles</i> . . . . .	60
15	Aufruf des Web Services CheckInUniversal . . . . .	63
16	Überprüfen der Indizierung anhand einer Suchanfrage . . . . .	65
17	Überprüfen der Indizierung anhand einer Suchanfrage . . . . .	65
18	Aufruf des eigenen Web Services . . . . .	67
19	Befehlsdefinitionen der Benutzersimulation . . . . .	67
20	Angepasste Servicedefinition für die Benutzersimulation . . . . .	68



---

## Tabellenverzeichnis

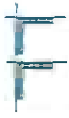
1	Rückgabewerte für Nagios-Plugins . . . . .	29
2	Übersicht der verschiedenen Unix-Agenten . . . . .	49
3	Übersicht der verschiedenen Windows-Agenten . . . . .	52

## Quellverzeichnis

- [DMS08] Götzer; Schmale; Maier; Komke (2008) „Dokumenten-Management - Informationen im Unternehmen effizient nutzen“  
4. Auflage,  
dpunkt.verlag GmbH Heidelberg, ISBN13: 978-3-89864-529-4,  
Stand: ????, Einsichtnahme: 25.06.2009
- [Barth08] Wolfgang Barth (2008) „Nagios - System- und Netzwerk-Monitoring“ 2. Auflage,  
ISBN13: 978-3-937514-46-8,  
Stand: ????, Einsichtnahme: 25.05.2009
- [Huff06] Brian Huff (2006) „The Definitive Guide to Stelent Content Server Development“,  
ISBN13: 978-1-59059-684-5,  
Stand: ????, Einsichtnahme: 25.05.2009
- [Jose07] David Josephsen (2007) „Bulding a monitoring infrastructure with Nagios“,  
ISBN13: 0-132-23693-1,  
Stand: ????, Einsichtnahme: 16.06.2009
- [SOA07] Hurwitz; Bloor; Baroudi; Kaufman; (2007) „Service Oriented Architecture For Dummies“,  
ISBN13: 978-0-470-05435-2,  
Stand: ????, Einsichtnahme: 29.07.2009
- [Melzer08] Ingo Melzer (2007) „Service-orientierte Architekturen mit Web Services: Konzepte - Standards - Praxis“,  
ISBN13: 978-9-8274-1993-4,  
Stand: ????, Einsichtnahme: 29.07.2009



- [Munin08] Gabriele Pohl und Michael Renner (2008) „Munin - Graphisches Netzwerk- und System-Monitoring“,  
ISBN13: 978-3-937514-48-2, Einsichtnahme: 05.04.2009
- [UCM07] Ohne Verfasser (2007) „Oracle Application Server Documentation Library - Oracle Content Management 10gR3“,  
Quelle: [http://download-west.oracle.com/docs/cd/E10316\\_01/cs/cs\\_doc\\_10/documentation/integrator/getting\\_started\\_10en.pdf](http://download-west.oracle.com/docs/cd/E10316_01/cs/cs_doc_10/documentation/integrator/getting_started_10en.pdf)  
Stand: unbekannt, Einsichtnahme: 16.06.2009
- [UCMlog09] Unbekannter Author „vramanat“ (2009) „Universal Content Management 10gR3 - Content Server Log File Information“,  
Quelle: <http://www.oracle.com/technology/products/content-management/cdfs/loginfo.pdf>  
Stand: 20.01.2009, Einsichtnahme: 05.06.2009
- [OraPress] Letty Ledbetter (2009) „Oracle Press Release - Oracle Buys Stellent“,  
Quelle: [http://www.oracle.com/corporate/press/2006\\_nov/stellent.html](http://www.oracle.com/corporate/press/2006_nov/stellent.html)  
Stand: 02.11.2006, Einsichtnahme: 16.06.2009
- [W3WS04] Booth; Haas; McCabe u.a. (2004) „Web Services Architecture - W3C Working Group Note 11 February 2004“,  
Quelle: <http://www.w3.org/TR/ws-arch/wsa.pdf>  
Stand: 11.02.2004, Einsichtnahme: 29.07.2009
- [NagiosFAQ] Ethan Galstad (2009) „What does Nagios mean?“,  
Quelle: [http://support.nagios.com/knowledgebase/faqs/index.php?option=com\\_content&view=article&id=](http://support.nagios.com/knowledgebase/faqs/index.php?option=com_content&view=article&id=)



---

52&catid=35&faq\_id=2&expand=false&showdesc=true

Stand: 02.06.2009, Einsichtnahme: 09.06.2009