

TECHNISCHE UNIVERSITÄT MÜNCHEN  
INSTITUT FÜR INFORMATIK

**Forschungs- und Lehrereinheit Informatik / LRZ**

Integration der Web-Server-Administration in ein unternehmensweites  
Systemmanagement auf der Basis von Tivoli TME10

**Einjähriges Systementwicklungsprojekt**

**MArk Schenkl**  
[MArk@Schenkl.de](mailto:MArk@Schenkl.de)  
<http://www.schenkl.de>

**Aufgabensteller:** Prof. Dr. Heinz-Gerd Hegering,

**Betreuer:** Dr. Kirsten Heiler (BMW AG)  
Dr. Bernhard Neumair

**Abgabetermin:** 31.10.98

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>3</b>
1.1. Systemmanagement	3
1.2. Aufgabenstellung	4
1.3. Begriffserklärung	5
1.4. Iconzuordnung	8
<b>2. Das integrierte Webserver Management bei BMW</b>	<b>9</b>
2.1. Tivoli Einbettung in die Unternehmensstruktur von BMW	9
2.2. Allgemeine Monitore	11
2.3. Die Möglichkeiten des Tivoli SuiteSpot Moduls	12
2.4. Bei BMW eingesetzte Monitore	15
2.5. Infrastruktur der Web-Server bei BMW	17
2.6. Installation in die Produktivumgebung	17
<b>3. Installation und Konfiguration des SuiteSpot Moduls</b>	<b>19</b>
3.1. Installationsanforderungen der Tivoli Software	19
3.2. Integration eines Rechners in die TMR	20
3.3. Integration des SuiteSpot Moduls in das TME10 Framework	21
3.4. Grundlegende Anpassung der Installation	22
3.5. Integration eigener virtueller Server	24
3.6. Konfiguration der Monitore	25
3.7. Konfiguration der Tasks	26
3.8. Indicator Collection	26
3.9. Anbindung der SuiteSpot Monitore an die TEC	26
3.10. Remedy Anbindung des SuiteSpot Moduls	27
<b>4. Fazit/Zusammenfassung Abschlussbemerkung</b>	<b>30</b>
<b>5. Anhang</b>	<b>31</b>

# 1. Einleitung

## 1.1. *Systemmanagement*

### **Die Situation**

Vor dem Hintergrund einer raschen Zunahme vernetzter Arbeitsplätze und verteilter Server sowie einer ständig steigenden Komplexität der Anwendungen muß die Betriebbarkeit der Client/Server-Strukturen weiterhin gesichert werden. Viele Großunternehmen stehen hierbei vor der Herausforderung, die sich rasch ändernden Hardware- und Softwarekonfigurationen zu erfassen und zu verwalten. Systemadministratoren stehen heute vor dem Problem nur unzulänglich die Hardware- und Softwarekonfiguration auch entfernter Rechner automatisch zu inventarisieren, anzusprechen, zu überwachen und zu verändern.

Der Betriebs- und Betreuungsaufwand ist aufgrund individueller Systemkonfigurationen, der Vielfalt an Anwendungssoftware und der durch zum Teil autarke, individuelle Projektlösungen geschaffenen, heterogenen IV-Infrastruktur ständig gestiegen, bei gleichzeitig stagnierenden bzw. zum Teil reduzierten Personalkapazitäten.

Die heutige Situation im Betrieb moderner Client/Server Infrastrukturen läßt sich am treffendsten durch folgendes Zitat beschreiben: "Ein verteiltes System (Client/Server) ist ..., wenn ein Benutzer nicht weiterarbeiten kann und daran ein Rechner schuld ist, von dessen Existenz er nicht einmal etwas geahnt hat."

### **Das Ziel**

Durch das Beherrschen der Heterogenität der verteilten Systemlandschaft soll eine erhöhte Verfügbarkeit der Client/Server-Anwendungssysteme erzielt werden. Dies führt unmittelbar zu einer qualitativen Verbesserung beim Anwender.

### **Die Lösung**

Um die Heterogenität der IV-Infrastruktur zu beherrschen, muß ein integriertes "Systems Management" implementiert werden. Dieses muß über ein einheitliches Konzept als Basis eines globalen Managements verfügen, das heterogene Systeme unterstützt und möglichst umfassende, einheitliche Schnittstellen sowie eine einheitliche Bedieneroberfläche anbietet. Nur so kann gewährleistet werden, unterschiedlichste IV-Systeme und Management-Tools zentral zu steuern und zu überwachen.

## **1.2. Aufgabenstellung**

### **Die Situation**

Die Zentrale IV (speziell Projekt Internet) der BMW AG hat die Aufgabe der gemeinsamen Wartung und Verwaltung der BMW Intra- / Internet Server. Probleme werden derzeit per „Zufall“ (Administrator überprüft die Webserver) oder durch „Zuruf“ (Telefon klingelt, Fachabteilung beschwert sich) erkannt und anschließend manuell behoben. Diese Verfahrensweise resultiert daher, daß die Web Technologie bislang bei BMW mehr oder weniger als Spielwiese diente. Allerdings haben sich im Laufe der Zeit die Anforderungen an das Intranet bzw. Internet geändert (zunehmende Anzahl von Servern und Projekten, unternehmenskritische Anwendungen etc.). Daher stößt die bisherige Verfahrensweise, insbesondere das Problemmanagement immer stärker an ihre Grenzen (keine 24 Stunden Bereitschaft, Problembehebung nur falls Mitglied der Abteilung Internet verfügbar). Es müssen neue Wege Richtung „integriertes Systemmanagement“ beschritten werden.

### **Das Ziel**

Im Bereich der zentralen IV (speziell Projekt Internet) der BMW AG soll eine zentralistische Fernüberwachung der im Zuständigkeitsbereich der Abteilung liegenden Webserver realisiert werden. Dabei soll nach Möglichkeit eine weitgehende Integration in den Helpdesk und die bestehende 24 Stunden Bereitschaft erreicht werden.

### **Die Lösung**

Bei der hier eingesetzten Serversoftware handelt es sich ausschließlich um Programme aus der „Netscape SuiteSpot“ (im speziellen Newsserver & WWW-Server). Es ist Firmenpolitik, das unternehmensweite Systemmanagement auf Tivoli abzustützen, also folgte man dieser Strategie und entschied sich für den Einsatz eines speziell von [Tivoli](#) für das Management der „Netscape SuiteSpot“ entwickelten Produktes. Es handelt sich hierbei um ein Modul namens „*TME10 Module for SuiteSpot*“ welches in das bereits im BMW Umfeld befindliche TME10 Framework (TMP) integriert werden soll. Ziel dieser Arbeit war Mittels einer Testinstallation zu evaluieren, welche Möglichkeiten zur integrierten Überwachung und Steuerung der BMW Webserver durch den Einsatz des SuiteSpot Moduls ergeben. Für den Testbetrieb wurde ein Webserver, welcher als Evaluierungsplattform diente, in die bei BMW installierte Tivoli Testumgebung integriert (siehe Kapitel 2.1). Die Hauptaufgabe bestand darin die mitgelieferten Monitore auszutesten und anzupassen.

## 1.3. Begriffserklärung

Dieses Kapitel dient als Einführung in die Tivoli Terminologie. Alle in dieser Ausarbeitung verwendeten *kursiv* gedruckten Fachwörter werden im folgenden erklärt.

- Bulletin Board

Das *Bulletin Board* ist die Schnittstelle zwischen TME und den Administratoren. Alle wichtigen Aktionen, die von TME ausgeführt werden, können im Bulletin Board festgehalten werden.

- Distributed Monitoring (ehem. Sentry)

Die Monitore zur Überwachung der Systemressourcen werden in *Sentry Profiles* definiert und dann an die zu überwachenden Systeme verteilt. Ein *Sentry Profile* kann Monitore verschiedenen Typs enthalten. Durch den Verteilvorgang werden die Monitore auf den entsprechenden Systemen gestartet, d.h. sie laufen lokal ab.

- Endpoint

Zu managende Ressource (z.B. Server, Workstation, Router, Mail-Server, SAP-System etc.), und damit letzter Empfänger (= *Subscriber*) eines Profiles.  
ACHTUNG: Endpoint kann sowohl ein physikalisch existierendes Gerät als auch eine funktionale Einheit (z.B. HTTP-Server) sein, was impliziert, daß ein oder mehrere Endpoints auf einem anderem „leben“.

- Indicator Collection

Icon, das auf Ereignisse reagiert (diese archiviert) und sein Aussehen verändert, wodurch der Administrator auf Engpässe/Probleme im System hingewiesen wird. Gesteuert werden die *Indikator Collections* mittels der *Sentry Monitore*, welche bestimmte Ereignisse an eine *Indicator Collection* weiterleiten können.

- Inventory

Tivoli Modul zur Inventarisierung der DV-Landschaften. Die hier gesammelten Daten können u.a. zu einer gezielten Steuerung der Softwareverteilung eingesetzt werden.

- Job

Verknüpfung eines Task mit einem oder mehreren *ENDPOINTS*

- Logfile Adapter

Programm, welches Daten aus Logfiles (i.d.R. syslog) ausliest, analysiert und an die TEC weiterleitet

- Managed Node

Bezeichnet alle PC's, Server und Clients in einer *TMR*.

- Notice Groups

Sammlungen von Nachrichten (ähnlich einer Internet Newsgroup); die *Notice Groups* werden im *Bulletin Board* zusammengefaßt bzw. abgelegt.

- Policy Region

Dienen der Organisation und Verwaltung von Systemressourcen. Über die *Policy Regions* wird die Rechteverteilung gemanagt.

- Sie dienen als Container, um die Systemressourcen (hierarchisch) anzuordnen.
- Sie dienen als Authorisierungseinheit für Administratoren, d.h. ein Administrator hat einen Satz von Rechten über einer *Policy Region*.

- Profiles

Ein *Profile* ist ein Satz von Applikations-spezifischen Informationen. Es besteht aus beliebig vielen Datensätzen, die Konfigurationsinfos eines Systems beinhalten. Die Informationen, welche in einem *Profile* abgelegt werden, sind dabei von dem Profil Typ abhängig, so enthält z.B. ein *User Profile* Informationen über verschiedene Accounts.

- Profile Manager

Sind Container für die verschiedenen *Profiles*, des weiteren legen sie fest, an welche Systeme die Informationen der *Profiles* verteilt werden (=Subscriber). *Profile Manager* sind dediziert einer Management Disziplin (Monitoring, Softwareverteilung, Inventarisierung, Benutzer-Administration) zugeordnet.

- Remedy ARS (Action Request System)

Bei BMW eingesetztes Trouble Ticket System, welches eine Schnittstelle zu

TME10 besitzt. Nähere Informationen hierzu findet man u.a. auf der Remedy Homepage <http://www.remedy.com>.

- Software Distribution (ehem. Courier)

Tivoli Modul zur Softwareverteilung

- Subscriber

Die Systeme, welche die Informationen der *Profiles* erhalten. Sie sind neben den *Profiles* selbst der zweite Bestandteil der *Profile Manager*.

- SuiteSpot

Tivoli Modul zum Management der SuiteSpot Internet Server von Netscape.

- Task

TME bietet die Möglichkeit komplexe Operationen (Skripten etc.) in einem Task abzulegen. Ein solcher Task kann für verschiedene Plattformen getrennt implementiert werden, d.h. der Entwickler schreibt ein für jede Plattform angepaßtes Programm. Für den Anwender ist es dann völlig belanglos auf welcher Plattform er den Task ausführen läßt, da Tivoli das jeweils passende Programm aufruft.

- TEC (Tivoli Enterprise Console)

Die TEC ist ein Event Management System, das Ereignisse von verschiedenen *Event Adaptern* (z.B. *Sentry*) entgegennimmt. Alle eingegangenen Events werden in einer Datenbank abgelegt und nach frei definierbaren Regeln (Rules) abgearbeitet. Dadurch ist es möglich mehrere Events miteinander zu korrelieren.

- TME Administratoren

Standardmäßig existieren 4 (exklusive) Administrationsrollen:

- *super*: Management der TMR
- *senior*: Aufbauen/Verändern der TME Struktur (Erzeugen der *Policy Region* Hierarchie, Erzeugen der *Profile Manager* Hierarchien, Erzeugen von *Profiles*, Definition der *Policies*)
- *admin*: Verändern von Konfigurationsdaten (Verteilen von *Profiles*)
- *user*: Anschauen (kein Ändern) von Systemressourcen.

Des weiteren können beliebige benutzerdefinierte Administrationsrollen (alle exklusiv) erstellt werden.

- TME Desktop

Graphische Oberfläche von Tivoli TME10 zum zentralen Management der Systemressourcen. Jeder Tivoli Administrator besitzt seine eigene Oberfläche, welche entsprechend seinen Aufgaben mit verschiedenen Ressourcen „gefüllt“ ist.

- TMR (Tivoli Managed Region)

Die Grundeinheit einer Tivoli Installation bildet die *Tivoli Managed Region* (TMR). Eine TMR besteht aus einem TMR Server und einer Menge von Clients. Der TMR Server besitzt einen Satz von Tivoli Software, der es möglich macht, die Clients zu verwalten und Management-Operationen auszuführen.

Eine TMR besitzt genau einen Server und bis zu 200 Clients (UNIX Plattform) und beliebig viele PC Managed Nodes. Es ist auch möglich, mehrere TMR's miteinander zu verbinden.

- User Administration

Tivoli Modul zur zentralen Benutzerverwaltung

## 1.4. Iconzuordnung



Bulletin Board mit  
gelesenen Nachrichten

un-



Tivoli Enterprise Console  
(TEC)



Scheduler



Policy Region



Spezielle Internet Manager  
Policy Region



Policy Region für  
Newsserver



Policy Region für Webserver



Profile Manager










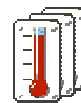




File Package



Sentry Monitor



	Profile Manager für allg. Internet-Server		Profile Manager für SuiteSpot Server
	Netscape News-Server Endpoint		Netscape Web-Server Endpoint
	(Windows NT) Workstation Endpoint		(Solaris) Server Endpoint
	Task Library		Task
	Job		Indicator Collection
	Indicator im Zustand „CRITICAL“		Internet Browser Verteilung

## 2. Das integrierte Webserver Management bei BMW

### 2.1. Tivoli Einbettung in die Unternehmensstruktur von BMW

Bei BMW wurden, um einen sicheren Betrieb und gleichzeitig ausgiebige Tests zu ermöglichen, drei Tivoli Umgebungen errichtet (siehe Schaubild Abbildung 1):

#### Testumgebung (Development)

Dient dem entwickeln (Skripte, Monitore, Tasks etc.) und testen neuer Produkte (neue Software und neue Softwareversionen). Sie erzwingt eine hohe Dynamik im Systemzustand. Nichtverfügbarkeit und Neuinstallationen sind hier die Regel. Sie ist die am wenigsten stabile Umgebung. Die Testumgebung besteht aus einem TMR

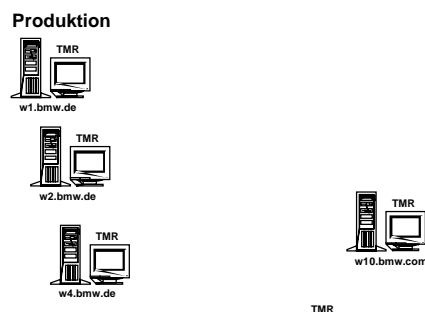
Server (tics02) und einer TEC (läuft auf tics01), als Managed Nodes sind die verschiedensten Rechnerarchitekturen eingebunden, u.a. auch ein Sun Rechner, welcher als Webserver dient.

## Integrationsumgebung

In der Integrationsumgebung wird die Einführung neuer Produkte vorbereitet, entwickelt und getestet. Diese Umgebung ist weniger stabil als die Produktionsumgebung. Änderungen am System unterliegen in der Regel nicht den strengen Auflagen der Produktionsumgebung. Ein näher zu beschreibender Prozeß regelt den Vorgang der Übernahme von Ergebnissen der Integrationsumgebung in die Produktionsumgebung. Auch sie besteht aus einem TMR Server (tics02) und einigen Managed Nodes. Ich hatte jedoch mit dieser Umgebung keinerlei Berührung.

## Produktionsumgebung

Die eigentliche produktive Tivoli Umgebung von BMW. Die Produktionsumgebung garantiert Verfügbarkeit und Stabilität des Systems während des Betriebes. In dieser stabilen Umgebung können Änderungen nur unter strenger Konfigurationskontrolle und in der Regel außerhalb der Betriebsstunden durchgeführt werden (z.B. einbinden neuer Regeln in die TEC). Weiterhin müssen notwendige Änderungen am System langfristig geplant werden, um Produktionsausfälle zu vermeiden. Die Produktionsumgebung besteht aus einem Main/Haupt TMR Server (SP 2) und mehreren untergeordneten TMR Servern. Sie beinhaltet sogar mehrere Enterprise Consolen. Die Aufteilung in mehrere *Tivoli Managed Regions* resultiert vor allem aus der Limitierung der derzeitigen TME Versionen auf 200 Unix Rechner.



## **2.2. Allgemeine Monitore**

Bei der Integration eines neuen *Managed Node* werden automatisch einige vordefinierte Skripten installiert. Es handelt sich dabei um speziell an die Rechnerumgebung von BMW angepaßte Monitore zur Überwachung des Filesystemes, Prozessor, Speicher und ähnlichem:

### Unix Heartbeat

Überwacht mittels des Tivoli Kommandos `wping` das Vorhanden sein der *Managed Nodes* im Netz sowie den *oserv* Daemon.

### Uptime

Gibt Alarm falls der Rechner erst vor kurzem gebootet/gestartet wurde.

### Filesysteme

Es wird standardmäßig der freie Platz auf sechs Filesystemen überprüft (root; /tmp; /usr; /var; /usr/tivoli; /var/tivoli)

### Prozessüberwachung

Von vier Prozessen (syslogd; cron; inetd; snmpd) wird eine up/down Überprüfung vorgenommen.

### Weiteres

Es wird noch eine Auswertung des Syslog Files vorgenommen und die Anzahl der Zombie Prozesse bzw. die Größe des verfügbaren Swap Space überprüft.

Alle diese Monitore lösen, wenn bestimmte Grenzwerte überschritten werden, Events aus, welche an die TEC zur Verarbeitung weitergereicht werden.

Nähere Informationen mit Angabe der Schwellwerte, ausgelösten Reaktionen und Aufrufintervallen befinden sich im [Anhang](#).

## 2.3. Die Möglichkeiten des Tivoli SuiteSpot Moduls

Neben den soeben vorgestellten Standardmonitoren werden in unserem Fall spezielle an die zentrale Überwachung der Netscape Webserver angepasste Monitore benötigt.

Die Netscape Serverfamilie beinhaltet folgende Produkte:

- Directory Server:

LDAP fähiger Directory Server, für Telefonverzeichnisse, Mitarbeiterkonten etc.

- HTTP Server:

Server zur Integration eigener Seiten in das WWW.

- Mail Server:

Internet Mail Server basierend auf dem SMTP, POP3 und IMAP4 Protokoll.

- News Server:

NNTP Server für den Anschluß an das Usenet bzw. die Erstellung einer eigenen News Architektur.

- Proxy Server:

Software zur Weiterleitung bestimmter Protokolle (z.B. HTTP-Proxy)

Für deren Verwaltung und Überwachung bietet das SuiteSpot Modul schon fertig konfigurierte Monitore und Tasks (die bei BMW eingesetzten Monitore sind mit ⊗ gekennzeichnet, eine ausführliche Begründung befindet sich im [nächsten Kapitel](#)):

### Monitore

#### Allgemeine Monitore (Generic / All Managed Server Monitors)

- **Server Process Status (1h):** überprüft, ob Server Daemon läuft (JA/NEIN Überprüfung). ⊗
- **Server Process Size (30min):** Überprüft den Verbrauch an physikalischem RAM (*Resident Set Size*)

ACHTUNG: der Monitor kann nicht verschiedene Server (auf unterschiedlichen Ports) unterscheiden, und liefert den Gesamtverbrauch zurück.

- **Log File Size (All Logs) (2h)**
- **Server Process Count (30min):** Liefert Anzahl der httpd Dämonen, welche auf dem Server laufen (→ ist die Performance der HW noch ausreichend).  
ACHTUNG: der Monitor kann nicht verschiedene Server (auf unterschiedlichen Ports) unterscheiden, und liefert die Gesamtanzahl zurück.
- **Log File Free Space (2h):** Liefert freien Speicherplatz auf dem Filesystem des Logfiles. ☒

#### Spezielle Monitore für WWW Server (HTTP Server Monitors):

- **Document Root Size (2h):** Liefert den vom *Dokument Root* (bei FI-6 meist /www) verbrauchten Speicherplatz.
- **Document Root Free Space (2h):** Gibt den freien Speicherplatz auf dem Filesystem des *Document Root* (bei FI-6 meist /www) wieder. ☒
- **Logfile Analysis Capacity (1h):** Liefert Zeit welche die *SuiteSpot* Monitore zur Abarbeitung des Logfiles benötigen. ☒
- **Logfile Entries (total) (4h)**
- **Logfile Entries (Unparsed) (1d):** Gibt Anzahl der Einträge zurück welche das *SuiteSpot Modul* nicht parsen kann.
- **Server Requests (2h):** Gibt Anzahl Requests zurück (*god, all, failed*). ☒
- **Data Transmitted (4h):** Gibt Anzahl der gesendeten KB seit dem letzten Abschneiden des Logfiles zurück (*total, average, min*).
- **Server Error Cnt (4h):** Liefert Anzahl Fehler aus dem Logfile zurück (*all, request, auth, perm, missing, timeout, server*). ☒
- **Netscape HTTP Monitoring Collection:** Existiert leider nur unter NT zur Performanceüberwachung.

#### Spezielle Monitore für News Server (News Server Monitors):

- **Free Space (2h):** Liefert freien Speicherplatz auf dem Filesystem der (*active, archived, incoming, outgoing*) Newsgruppen.
- **Space Used (2d):** ditto
- **Logging/History Size (1d)**
- **News Group Count (1d):** Bestimmt Anzahl Dokumente spezifizierter Newsgroups (*all, local, disabled, remote, moderated, junk, mapped*)
- **Global Remember Span (2d):** Anzahl Tage welche News maximal auf dem Server liegen.
- **News Group Expiration (2d):** Wie oben nur auf spezielle Newsgroup bezogen
- **Article Count (4h):** Gibt Anzahl Artikel in einer Newsgruppe wieder
- **News Group Status (8h):** Gibt den (*reading/posting*) Status einer Newsgruppe wieder.
- **Rejected Articles (8h):** Anzahl der Artikel (einer Newsgruppe) welche zurückgewiesen wurden.
- **NNTP Send Events (4h):** Gibt Anzahl von verschiedenen Events zurück (*MissingArgument, IllegalOption, CantReadControlFile, BatchDirMissing, ControlFileError, HistoryDatabaseError, HistoryFileError etc.*)

## Tasks

### Allgemeine Tasks (Generic / All Managed Servers Tasks Task Library):

- **Archive All Managed Server Logs**
- **Delete All Managed Server Logs**
- **Get All Managed Server Status** (Active/Inactive)
- **Get All Managed Server Info** (Server Name, Server Typ, Log Verzeichnis und Hostname)
- **Get All Managed Server Version**
- **Restart All Managed Servers**
- **Rotate All Managed Server Logs**
- **Start All Managed Servers**
- **Stop All Managed Server**

### Spezielle Tasks für WWW Server (Webserver Tasks Task Library):

- **Discovering Your Webservers:** Überwachung der Antwortzeiten auf verschiedenen Ports (mittels CLI).
- **Archive Webserver Logs**
- **Delete Webserver Logs**
- **Get Webserver Information**
- **Get Webserver Status**
- **Get Webserver Version**
- **Restart Webservers**
- **Rotate Webserver Logs**
- **Start Webservers**
- **Stop Webservers**
- **Webserver Access Report** (für uns relativ uninteressant)
- **Webserver Statistics Report** (für uns relativ uninteressant)

### Spezielle Tasks für News Server (News Server Tasks Task Library):

- **Archive News Server Logs**
- **Delete News Server Logs**
- **Get News Server Info**
- **Get News Server Status**
- **Get News Server Version**
- **Restart News Servers**
- **Rotate News Server Logs**
- **Start News Servers**
- **Stop News Servers**
- **Allow News Server Readers:** Erlaubt Clients Daten aus dem News Server zu lesen (was mit schreiben??)
- **Deny News Server Readers**
- **List News Server Newsgroups**

- **Receive News Server Articles:** Erlaubt Server die Annahme von Artikeln anderer News Server
- **Reject News Server Articles**
- **Remove News Server Newsgroups**
- **Expire News Server Articles**
- **Get News Server Statistics**
- **Add INND News Server News Group:** Neue Newsgruppe zum Server hinzufügen

## Automatische Client (Browser) Verteilung

Dazu ist das TME10 Modul *Courier* nötig, welches bei BMW installiert ist. Allerdings wird derzeit noch evaluiert, welches Produkt überhaupt zu Softwareverteilung eingesetzt werden soll (INTEL LANDesk oder Tivoli). Es existieren zwar einige Bereiche, in welchen eine automatisierte Softwareverteilung eingesetzt wird (z.B. Venus im CA-Bereich), aber einen konzernübergreifenden Standard zur Softwareverteilung gibt es derzeit noch nicht.

### 2.4. Bei BMW eingesetzte Monitore

Nach ausgiebigen Tests der im *SuiteSpot* Modul mitgelieferten Monitore wurde entschieden lediglich 5 (im vorherigen Kapitel mit ⊗ markiert) der 14 zur Verfügung stehenden Monitore zu verwenden. Aufgrund seiner exakt auf unsere Problemstellung passenden Funktionalität wurde beschlossen, auch das Korn Shell Skript eines Mitarbeiters (Erläuterung folgt [weiter unten](#) in diesem Kapitel) in einen Monitor umzuschreiben und in Tivoli zu integrieren.

#### Logfile Analysis Capacity

Ist notwendig, da es der „Selbstüberwachung“ dient. Dazu muß gesagt werden, daß die meisten Monitore das Logfile des jeweiligen HTTP Server *Endpoints* auswerten, dazu benötigen sie je nach Rechnerauslastung und Logfilegröße eine gewisse Zeit. Standardmäßig bekommt jeder Monitor ein Timeout von 60 Sekunden, d.h. er muß das Logfile innerhalb dieser Zeit abgearbeitet haben. Dieser Prozeß bestimmt nun generisch, wie lange eine Auswertung des Logfiles dauern würde. Sollte hier ein Wert größer als 100 zurückgegeben werden (der Prozeß liefert Werte zwischen 0 und 100% des Timeout zurück), müssen geeignete Maßnahmen ergriffen werden, z.B. durch vergrößern des Timeout; verringern der Last auf dem Server; kürzen (bzw. sichern) des Logfiles.

#### Document Root Free Space

Ermittelt den freien Speicherplatz im Dokumenten-Verzeichnisbaum. Dieser Monitor ist nützlich, da jede Fachabteilung in ihrem Verzeichnisbaum uneingeschränkte

Schreibrechte besitzt. und theoretisch in der Lage wäre, das Filesystem vollzuschreiben

### Logfile Free Space

Ermittelt den freien Speicherplatz im Filesystem des Logfiles. Da das Logfile nicht besonders schnell wächst (pro Zugriff nur ein paar Byte), wird dieser Monitor nur relativ selten aufgerufen.

### Server Process Status

Vollzieht eine einfache up/down Prüfung des Serverprozesses. Dieser Monitor wird einmal pro Minute ausgeführt. Sollte sich der Status dieses Monitors von up nach down ändern (up down) so wird ein „FATAL EVENT“ an die TEC geschickt. Verbleibt der Server im Zustand down, geschieht nichts (es werden auch keine weiteren *Events* an die TEC geschickt) erst wenn sich der Status wieder nach up ändert (down up) wird ein „HARMLESS EVENT“ an die TEC geschickt. Geschieht das innerhalb der vom Systems Management vorgegebenen 15 Minuten Frist, wird kein Trouble Ticket generiert.

### Numeric Skript

Hierbei handelt es sich um das schon erwähnte Korn Shell Skript, welches bereits zur Serverüberwachung eingesetzt wird (Listing siehe [Anhang](#)).

Sollte ein Fehlercode größer als Hundert zurückkommen so wird ein „FATAL EVENT“ und gleichzeitig ein Restart des Servers versucht.

Dieser Prozeß nimmt also immer nur eine lokale Überwachung des HTTP Ports vor, d.h. Netzwerkfehler können damit nicht erkannt werden. Das erschien mir hier auch nicht sinnvoll da das Netzwerk speziell mit NetView überwacht wird. Die Erreichbarkeit des Servers wird obendrein durch den „Heartbeat“ Monitor sichergestellt. Selbst wenn das nicht der Fall gewesen wäre, bliebe immer noch die Frage von welchem Host aus überwache ich das System?? Da dieser Monitor wohl der wichtigste von allen ist wird auch er einmal pro Minute aufgerufen. Im Fehlerfall liefert er Werte größer 100 zurück (Code siehe Anhang).

### Server Error Count

Dieser Monitor wird zweimal eingesetzt, einmal mit dem Attribut „all“ (d.h. er zählt alle möglichen Fehler welche im Logfile aufgetreten sind) , und zum zweiten mit dem Attribut „timeout“ (d.h. hier werden nur die aufgetretenen timeouts gezählt). Ich habe diesen Monitor gesondert mit einbezogen, um eventuelle Netzwerkfehler (Überläßt etc.) besser erkennen zu können. Auf dem Testrechner traten jedoch keine Server Fehler, egal welcher Art auf.

### Server Requests

Dieser Monitor wird nur mit dem Attribut „bad“ aufgerufen (andere Möglichkeiten wären „all“ und „good“). Schlägt Alarm, falls sich Fehlermeldungen im Logfile häufen



sollten, in unserer Testumgebung traten jedoch keine diesbezüglichen Events auf. Daher wird auch dieser Monitor nur relativ selten aufgerufen.

Die genauen Einstellungen der hier aufgezählten Monitore, wie Grenzwerte, ausgelöste Alarme, Aufrufintervalle entnehmen Sie bitte dem [Anhang](#). Dort sind alle auf den Webservern eingesetzten Monitore aufgezählt.

## **Fazit**

Nach Abschluß meiner Betrachtung läßt sich sagen, daß der Funktionsumfang der Monitore nicht generell überzeugen kann, insbesondere wenn es um die Funktionalitätsprüfung des httpd Daemon geht. Dazu existiert nur ein Perl Skript welches weder als Monitor noch als Task implementiert wurde und im Funktionsumfang nicht an das von meinem Kollegen erstellte Korn Shell Skript heranreicht.

## **2.5. Infrastruktur der Web-Server bei BMW**

Da sich die Konfiguration/Hardware der Webserver gerade in einer Umstrukturierung befindet, werde ich mich in meiner Ausarbeitung auf den zukünftig vorgesehenen Aufbau beschränken. Wenn es an einzelnen Stellen notwendig ist, werde ich auf die Änderungen hinweisen. Die Testumgebung besteht im wesentlichen aus 2 Rechnern (intra3.muc und watertest bzw. intra4.muc). Die Produktivumgebung besteht aus 3 Rechnern (wwwa, wwwc und wwwe). Jeder dieser Rechner verfügt über mehrere Netzwerkkarten. Die Rechner in der Testumgebung besitzen jeweils eine Ethernet und eine FDDI, die in der Produktivumgebung noch eine FDDI Karte zusätzlich. Des weiteren existieren auf diesen physikalischen Interfaces (Netzwerkkarten) meist noch eine Reihe logischer (zur einfacheren Adressierung der virtuellen Webserver). Das heißt, daß beispielsweise der DNS Eintrag [www.muc](#) einer logischen IP Adresse entspricht, welche wiederum dem FDDI Interface der wwwe zugeordnet ist. Aufgrund dieser Konfiguration ergab sich in unserem Testbetrieb das Problem, daß Tivoli nicht genau wußte, welche Interfaces alle zu einem Rechner gehören, Abhilfe kann hier mittels des „odadmin“ Kommandos geschaffen werden. Nähere Informationen siehe [Kapitel 3.2](#).

Genauere Informationen zur Hardware Struktur finden Sie im [ANHANG](#).

Bei den Rechnern der Produktivumgebung dienen zwei dem Intranetangebot (wwwa/wwwc), und einer dem Internetangebot (wwwe). Da auf das BMW Intranet demnächst auch sensible Anwendungsgebiete transferiert werden, wird hier ein HA (high availability) Lösung angestrebt. Da diese jedoch Hardwaremäßig noch nicht implementiert ist, ist dies auch in Tivoli nicht gesondert berücksichtigt worden.

## **2.6. Installation in die Produktivumgebung**

### **Eingebundene Rechner**

In die Produktivumgebung von Tivoli sollen die folgenden Rechner eingebunden werden:

Rechnername	1. FDDI Interface	2. FDDI Interface	Ethernet Interface
wwwa	wwwa	wwwb	Intra20
wwwc	wwwc	Wwwd	Intra21
wwwe	wwwe	wwwf	Inter1
Intra3	Intra3		Intra1
watertest	Intra4	-	Intra2

Es ist zu beachten, daß mittels des Tivoli Kommandos:

```
odamin odlist add_hostname_alias
```

zumindest den Rechnernamen und das Interfaces mit welchem der oserv kommuniziert an den Managed Node zu binden (Alias), da es ansonsten zu Problemen mit Tivoli kommen kann (wie der Testbetrieb gezeigt hat). Sollte dies nicht reichen müssen eventuell weitere Interface Namen mit einbezogen werden.

Jeder dieser physikalischen Rechner beherbergt mehrere virtuelle Netscape Server der SuiteSpot Familie (hauptsächlich Enterprise 3.0), welche als *Netscape Webserver Endpoint* in Tivoli eingebunden wurden. Die Rechner selbst sind als *Managed Node* in die Tivoli Umgebung integriert.

Eine Liste mit (fast) allen virtuellen und physikalischen Servern in der BMW Umgebung finden Sie [im Anhang](#).

#### Beschreibung der Rechner:

- watertest: WWW Testumgebung mit virtuellen Enterprise Servern der Version 3 & 2 z.B. wwwtest.muc, bikedetest, bikecomtest etc.
- intra3: Produktivumgebung für [www.muc](#) (Zentral-Intranet Server von BMW). Soll komplett auf wwwa/wwwc migriert werden. Dieser Server soll später, wie auch die watertest als Testrechner dienen.
- wwwa/wwwc: sind zwei Solaris Rechner, welche zukünftig alle Intranet Server übernehmen sollen, dabei sollen die Rechner als HA (Hochverfügbarkeitslösung) implementiert werden, d.h. sollte einer der beiden ausfallen, übernimmt der andere dessen Aufgaben.
- wwwe: wird die Produktivumgebung für die Internetserver ([www.bmw.de](#), [www.motorrad.bmw.de](#) etc.)

## 3. Installation und Konfiguration des SuiteSpot Moduls

Diese Installationsanleitung wurde anhand der von mir gemachten Erfahrungen bei der Integration eines Rechners (watertest) in die Tivoli Entwicklungs- (Development-) Umgebung von BMW erstellt.

### 3.1. Installationsanforderungen der Tivoli Software

Das Tivoli Modul ist für folgende Plattformen erhältlich:

- HP9000 mit Motif 1.2
- HP9000 mit PA RISC 1.1 Architektur und HP/UX10.x mit Motif1.2
- IBM RS600 oder Power PC Architektur mit AIX 3.2.5, 4.1, 4.2 und Motif1.2
- Sun Sparc mit SunOs 4.1.2, 4.1.3, 4.1.4 und Motif 1.2
- Sun Sparc mit Solarix 2.3, 2.4, 2.5 und OpenLook mit Motif 1.2
- PC mit Windows NT 3.51 und Windows NT CSD 4
- PC mit Windows NT 4.0
- oder neuere Versionen.

Tivoli-seitig muß in UNIX Umgebungen bereits folgende Software installiert sein:

#### Zwingend:

- TME10 Framework 3.0 Rev C mit TMP Service Pack 1
- TME10 Distributed Monitoring 3.0 Rev A

#### Optional:

- TME10 Software Distribution 3.0 Rev B (Wenn man das *Internet Client Deployment* nutzen will, d.h. automatische Verteilung der Internetagenten)
- TME10 Enterprise Console 2.6 Rev B

In einem NT Umfeld gelten die folgenden Voraussetzungen:

#### Zwingend:

- TME10 Framework 3.1 mit TMP Service Pack 1, TME10 Framework 3.1 patch 3.1-TMP-0007, BASH patch 3.1-TMP-0015
- TME10 Distributed Monitoring 3.0

#### Optional:

- TME10 Software Distribution 3.1
- TME10 Enterprise Console 3.1

Speicherbedarf für Bin Verzeichnis: 65MB  
 Speicherbedarf für DB Verzeichnis: 6MB  
 Speicherbedarf im /etc Verzeichnis ist minimal.  
 Trotz dieser Erkenntnis sollte man sich an die Standardvorgaben der BMW Abteilung *Systems Management* halten, welche für unsere spezielle Sun Umgebung angepaßt wurden:

**TIVOLI CHECKLISTE für Sun-Sparc-Rechner:**

<b>Voraussetzung</b>	<b>Erfüllt (JA/NEIN)</b>
Können alle Hostnamen aufgelöst werden	
Routing und ARP funktionsfähig	
Ist <i>nobody</i> :*:60001:60001::: eingerichtet	
Ist Zugang mittels <i>rexec</i> möglich	
Speicherplatz für Datenbank 20MB in <i>\$DBDIR</i> (+11MB reserve)	
Speicherplatz für Binaries 90MB in <i>\$BINDIR</i>	
Speicherplatz für Suitespot-Modul 5MB (14 bei NT)	
SunOS ab 4.1.2 & Motif 1.2 <u>oder</u> Solaris ab 2.3 & Openlook & Motif 1.2	
Speicherplatz auf / mind. 500kB	
Ist Port 94 frei?	
Pfad <i>/usr/tivoli</i> (für Binaries) & <i>/var/tivoli</i> (für DB) erzeugt. Es sollten für beide Pfade eigene Filesystem angelegt werden sonst mittels <i>ln -s</i> in ein gemeinsames Filesystem verweisen.	
User <i>tivoli</i> mit Benutzernummer 0 anlegen	

**Anmerkungen:**

Für das Datenbankverzeichnis wurde aus Speicherplatzgründen ein symbolischer Link auf */export/home/tivoli/var* gesetzt. Ebenso wurde für das Verzeichnis der Binaries (ausführbaren Tivoli Dateien) ein symbolischer Link auf */export/home/tivoli/usr* gesetzt. Ein weiterer Grund für dieses Vorgehen sind Bedenken der TME10 Administratoren, daß die Tivoli Applikation das Filesystem vollschreiben könnte (passiert ist das aber noch nie). Der zusätzliche Speicherplatz welchen das SuiteSpot Modul belegt geht hauptsächlich vom Binary Verzeichnis ab. Die Motif & Openlook Libraries benötigt man nur, wenn man die GUI auf dem jeweils lokalen Managed Node starten will. Hilfreich mag dies allemal sein, insbesondere bei überlastetem Netz bzw. TMR-Server. Port 94 ist der Standardport, auf welchem der *oserv* Daemon mit der TMR kommuniziert. Unter der Tivoli Kennung wird der *oserv* Daemon ausgeführt, sowie die *Monitore*. Sie kann auch benützt werden um die *Tasks* auszuführen. Aus Sicherheitsgründen sollte man das interaktive einloggen als User Tivoli sperren.

### **3.2. Integration eines Rechners in die TMR**

Die Integration eines neuen Rechners verläuft grundlegend immer gleich: Als erstes wird er als neuer Managed Node aufgesetzt, anschließend werden die Standard-

monitore des „System Management“ Projektteams aufgesetzt. Dies ist alles durch Skripten automatisiert.

Einzigste Besonderheit unserer Rechner, welcher extra Rechnung getragen werden muß, ist der Einsatz mehrerer Netzwerkkarten. Daher sollten, um einen späteren einwandfreien Betrieb zu gewährleisten, mittels des Tivoli Kommandos:

```
odadmin odlist add_hostname_alias od ipaddr hostname
```

bzw.

```
odadmin odlist add_IP_alias od ipaddr
```

alle physikalischen Interfaces an den Managed Node gebunden werden (Alias), da es ansonsten zu Problemen mit Tivoli kommen kann (wie der Testbetrieb gezeigt hat). Dieser Befehl kann nur direkt auf dem TME Server ausgeführt werden.

Parameter	Bedeutung
od	object dispatcher (kann mittels odadmin odlist ermittelt werden)
ipaddr	IP Adresse
hostname	Hostname alias

### **3.3. Integration des SuiteSpot Moduls in das TME10 Framework**

Die Installation des SuiteSpot Moduls bereitete anfangs einige größere Schwierigkeiten, welche jedoch nicht dem Modul zuzuschreiben sind, sondern an der sehr instabilen Testumgebung (bzw. Dev-Umgebung) lagen (z.B. Fehler im Bulletin Board; Probleme mit Backup).

Standardmäßig wird das SuiteSpot Modul sowohl auf dem *TMR (Tivoli Managed Region)* Server, als auch auf den jeweiligen *Managed Nodes* (auf welchen die WebServer überwacht werden sollen) installiert.

Bevor man mit der Integration eines neuen Moduls beginnt, sollte man immer ein Backup (*wbackupdb*) des bestehenden Systems anlegen, wobei sich die oben schon angesprochenen Probleme ergaben. Es verlief anfangs nur ein Teil der Installation erfolgreich, der Fehler lag wie sich später herausstellte, jedoch nicht an SuiteSpot, sondern an einer defekten Bulletin Board Datenbank.

Wir wählten die menügeführte Installation mittels GUI (*TME10 Desktop*). Für das Ausführen des Installations Task wird die Rolle *super* oder *install\_produk*t benötigt. Man wählt im „Install“ „Install Produkt“ Menü das SuiteSpot Modul aus, sollte es noch nicht aufgeführt sein, so muß zuerst mittels des „Select Media“ Buttons der Pfad der Installationsdateien bestimmt werden (in unserem Fall direkt von der CD). Anschließend wählten wir noch die Clients aus, auf welchen das Modul installiert werden sollte. In der Testumgebung waren das nur die „wwwtest.muc“ und der TMR Server (tics02).

Parallel dazu existiert noch die Möglichkeit einer Installation mittels des Command Line Befehls "winstall". Nach mehreren vergeblichen Versuchen stand letztendlich dann doch noch eine Grundinstallation von Tivoli zur Verfügung.

### 3.4. Grundlegende Anpassung der Installation

Tivoli richtet nach erfolgreicher Installation eine Grundkonfiguration des SuiteSpot Moduls ein. Da sich Tivoli nicht an die Namenskonventionen von BMW hält, sollte als erstes die Namensgebung aller benötigten Ressourcen an die BMW Konventionen (keine Leerzeichen, Pseudohierarchie mit „-“ getrennt) angepaßt werden. Dabei müssen die folgenden drei Ressourcen umbenannt werden:

#### Namensänderung

Vom SuiteSpot Modul vorgegeben	→	BMW Namenskonvention
Internet Manager/Web Manager		../central-web-manager
Internet Manager/Web Manager/Webserver Tasks		../central-web-manager-tl
Internet Manager/Web Manager/Webserver Monitors		../central-web-manager-sentries

Die *Policy Region* „Internet Manager“ wurde nicht umbenannt, da die „View“ des Administrators direkt „central-web-manager“ sein wird.

Nähere Informationen zur BMW Namenskonvention befinden sich in dem Papier „Architektur für die Einführung des Tivoli Management Environments bei der Bayerischen Motoren Werke AG“ welches von der System Management Gruppe herausgebracht wird.

#### Namensänderung von Policy Region oder Profile Managern

central-web-manager ist eine *Policy Region*, d.h. sie kann direkt mittels der GUI (rechte Taste auf Web Manager Icon dann „Properties“) umbenannt werden.

central-web-manager-sentries ist ein *Profile Manager*, d.h. auch er kann direkt mit der GUI umbenannt werden, allerdings muß er hierzu geöffnet werden, anschließend im Menü Edit „Profile Manager...“ aufrufen.

central-web-manager-tl ist eine *Task Library*, und damit ist alles anders, d.h. es ist keine Änderung direkt per GUI möglich! Als Alternative bietet sich hier nur das Command Line Interface an.

#### Labeländerung einer Ressource

Prinzipiell gibt es unter Tivoli keine Möglichkeit den Namen einer *Task Library* konsistent zu ändern. Zwar existiert ein direkter Aufruf auf IDL Ebene, welcher das Label jeder Ressource ändern kann z.B.:

```
wlookup -r TaskLibrary -a //Gibt alle Task Libraries mit
                          //ihrer OID aus
idlcall OID _get_label >muh //Gibt derzeitigen Namen aus
                          //Datei muh editieren
idlcall OID _set_label <muh //Weißt den Dateinamen aus muh
                          //der TaskLibrary zu
```

Allerdings führt das zu Inkonsistenzen, da die Elemente innerhalb der umbenannten *Task Library* (die eigentlichen Tasks und Jobs) die Namensänderung nicht nachvollziehen, und dadurch unbrauchbar werden (bzw. nicht mehr aufrufbar sind).

## Namensänderung einer Task Library

Die einzige Möglichkeit ist das komplette Exportieren und Importieren der gesamten *Task Library* (in eine neue nach BMW Norm Benannte). Das kann mit folgenden Befehlen geschehen:

```
wttl -F export-file-name -l library-name

//Erzeugt ein tar Archiv

tar -xvf export-file-name

//Die Datei tll ändern (Eintrag TaskLibrary "alt" durch neuen
//Namen Ersetzen ca. 4 Zeile)

wttl -r PolicyRegion -p Präprozessor tll

//Importieren der Task Library in die angegebene Policy Region
```

Parameter	Bedeutung
Export-file-name	Name des zu erstellenden TAR Archives
Library-name	Name der zu ex-/importierenden Task Library
PolicyRegion	Name der zugehörigen Policy Region
Präprozessor	C Präprozessor Pfad gewöhnlich /lib/cpp
Tll	Task Library Language File das von wttl im Archiv erstellt wird.

Aus technischen Gründen tritt derzeit ein Fehler beim reimportieren der *Task Library* auf, es fehlen sämtliche Jobs.

### 3.5. Integration eigener virtueller Server

Nachdem nun die wichtigsten Grundregeln für eine Integration in die BMW TMR befolgt wurden, kann mit der Integration der (virtuellen) Webserver begonnen werden. Da es sich hier ausschließlich um HTTP Server handelt werden die *Endpoints* in der *Policy Region* „Internet Manager/central-web-managers“ angelegt. Eine Automatisierung mittels Skripten ist hier weder sinnvoll noch notwendig, da die nötigen Eingaben sehr intuitiv und individuell sind. Eine Installation mit dem *Tivoli Desktop* wollte mir leider verwehrt bleiben (Bug in der Dev-Umgebung?), so daß als Alternative nur das CLI bleibt.

#### Installation eines HTTP-ENDPOINTS mittels CLI:

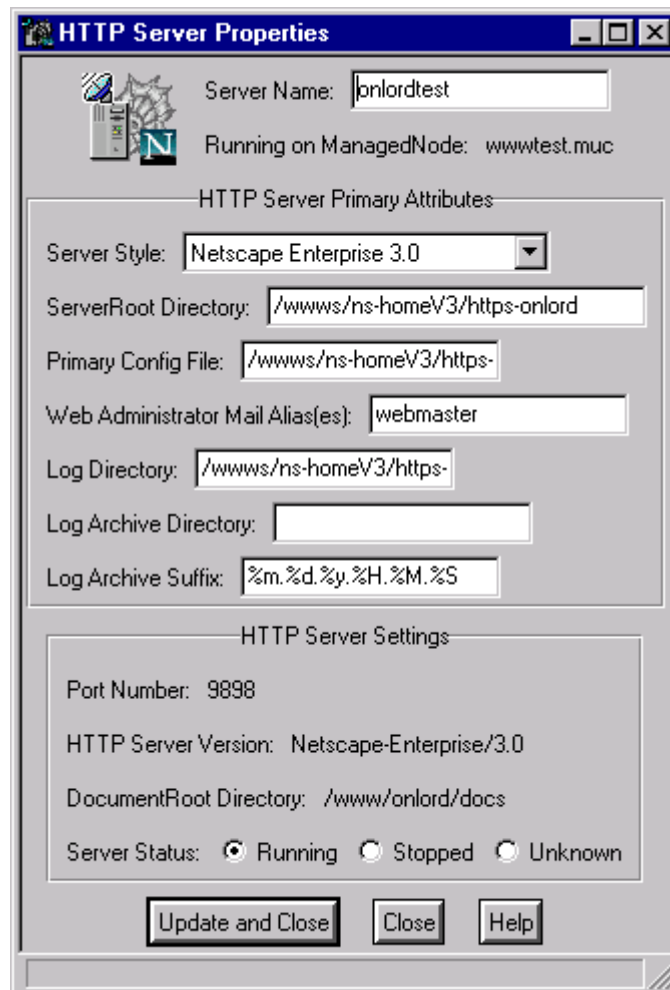
Beispiel:

Technical Settings		
Server Root		/wwws/ns-homeV3/https-carconftest
Hostname		Carconftest
IP address		135.143.150.90
Port		80
Log Files		/wwws/ns-homeV3/https-carconftest/logs/
Config File		/wwws/ns-homeV3/https-carconftest/config/magnus.conf
Managed Node (Host Rechner)		wwwtest.muc
Webmaster Email		Webmaster
Server Version		Netscape Enterprise 3.0

```
wcrthttp -n carconftest -p "central-web-managers"-h  
wwwtest.muc -s "Netscape Enterprise 3.0"-r /wwws/ns-  
homeV3/https-carconftest -c /wwws/ns-homeV3/https-  
carconftest/config/magnus.conf -m webmaster -l /wwws/ns-  
homeV3/https-carconftest/logs
```

Dieser Befehl richtet auf dem *Managed Node* „wwwtest.muc“ einen *HTTPServer-Endpoint* namens „carconftest“ in der *PolicyRegion* „central-web-managers“ ein. Die restlichen Informationen (z.B. IP Adresse, Port etc.) holt sich Tivoli direkt aus der Konfigurationsdatei. Anschließend ist eine weitere Konfiguration der Ressourcen mittels GUI ohne Probleme möglich (*Endpoint* mit rechter Maustaste selektieren, anschließend „Properties“ auswählen siehe Abbildung 2).





**Abbildung 2**

Für nähere Informationen verweise ich auf das Handbuch „Module for SuiteSpot User's Guide“ Kapitel 7-5.

### **3.6. Konfiguration der Monitore**

Nun kann mit dem Aufsetzen der Monitore begonnen werden. Neben den durch das Tivoli Team standardmäßig installierten kommen hier noch die speziell an die Webserver angepaßten hinzu. Da die Monitore von mir schon konfiguriert wurden kann man sie einfach mit folgendem Skript aufsetzen:

[/smic/dev/projects/suite/setup\\_mon.sh](/smic/dev/projects/suite/setup_mon.sh)

Dadurch werden alle benötigten Monitore in das Default *Profile* „Web Monitors“ installiert, sollen die Monitore in ein anderes *Profil* integrierte werden, so ist der Profilname als Command Line Parameter mit zu übergeben.

Genauere Informationen zu den installierten Monitoren finden Sie im [Anhang](#) oder in [Kapitel 2.3](#).

### **3.7. Konfiguration der Tasks**

Beim *SuiteSpot* Modul werden zwei *Task Libraries* mitgeliefert, welche für unsere Aufgaben interessant sind, zum einen die „All Managed Servers Tasks Task Library“, zum anderen die „central-web-manager-tl“. Sie decken hauptsächlich Auswertungsaufgaben ab, indem kleine Statistiken basierend auf den jeweiligen Logfile Daten erstellt werden. Sie können aber auch dazu verwendet werden die Server zentral zu starten/stoppen. Es existiert auch ein Task welcher explizit den Status (running/stopped) des Web-Servers überprüft (Get Webserver Status). Nähere Informationen zu den *SuiteSpot Tasks* finden Sie in [Kapitel 2.3](#).

Aus der *Task Library* „All Managed Server Tasks“ sollte der *Task* (und *Job*) „Configure TEC“ herausgelöscht werden. Da dieser i.d.R. nur einmalig aufgerufen wird, und keinesfalls ohne Absprache mit dem Tivoli Team (allerdings sollte die Rechtspolitik von Tivoli ein installieren ohne ausreichende Berechtigungen verhindern).

### **3.8. Indicator Collection**

Nach der Namensänderung konnte man leider nicht mehr auf die voreingestellte *Indicator Collection* („Web Indicators“) zugreifen. Als einzige Lösung viel uns hier das Ersetzen der bestehenden „Web Indicators“ durch eine neue, d.h. löschen der *Indicator Collection* in der *Policy Region* „central-web-manager“. Anschließend mittels Menüpunkt „Create/IndicatorCollection“ eine Neue erstellen. Gegebenenfalls muß noch im Sentry Profile „Web Monitors“ die neue *Indicator Collection* als Default eingestellt werden. Dies geschieht mittels des Menüpunktes „Sentry/Select Indicator Collection“.

### **3.9. Anbindung der SuiteSpot Monitore an die TEC**

Um einer zentralen Überwachung der Web-Server gerecht zu werden, müssen die einzelnen Events der bereits installierten SuiteSpot Sentry-Monitore an die TEC weitergereicht werden.

Um dies möglichst einfach und standardisiert zu halten, liefert das SuiteSpot Modul dafür einen speziellen Job genannt „Configure TEC“ mit. Der zugehörige Task führt das Skript „/usr/tivoli/bin/generic/TME/NETCMDR/GENSVR/configure\_tec“ auf dem TEC-Server (in unserer Testumgebung die TICS02) aus.

Was das Skript leisten sollte:

- Konfiguration der TEC für eine Optimale Zusammenarbeit mit dem SuiteSpot TME10 Modul.

- Einrichten der entsprechenden Klassen und Regeln, ohne bestehende zu verändern.
- Einrichten einer Event-Gruppe „Internet Management“.
- Zuweisen der Event-Gruppe an einen Vorbestimmten Administrator
- Automatische Einbindung des SuiteSpot *Logfile-Adapters* für Firewall Rechner (hier nicht nötig da keine Firewall Rechner installiert wurden)

Probleme:

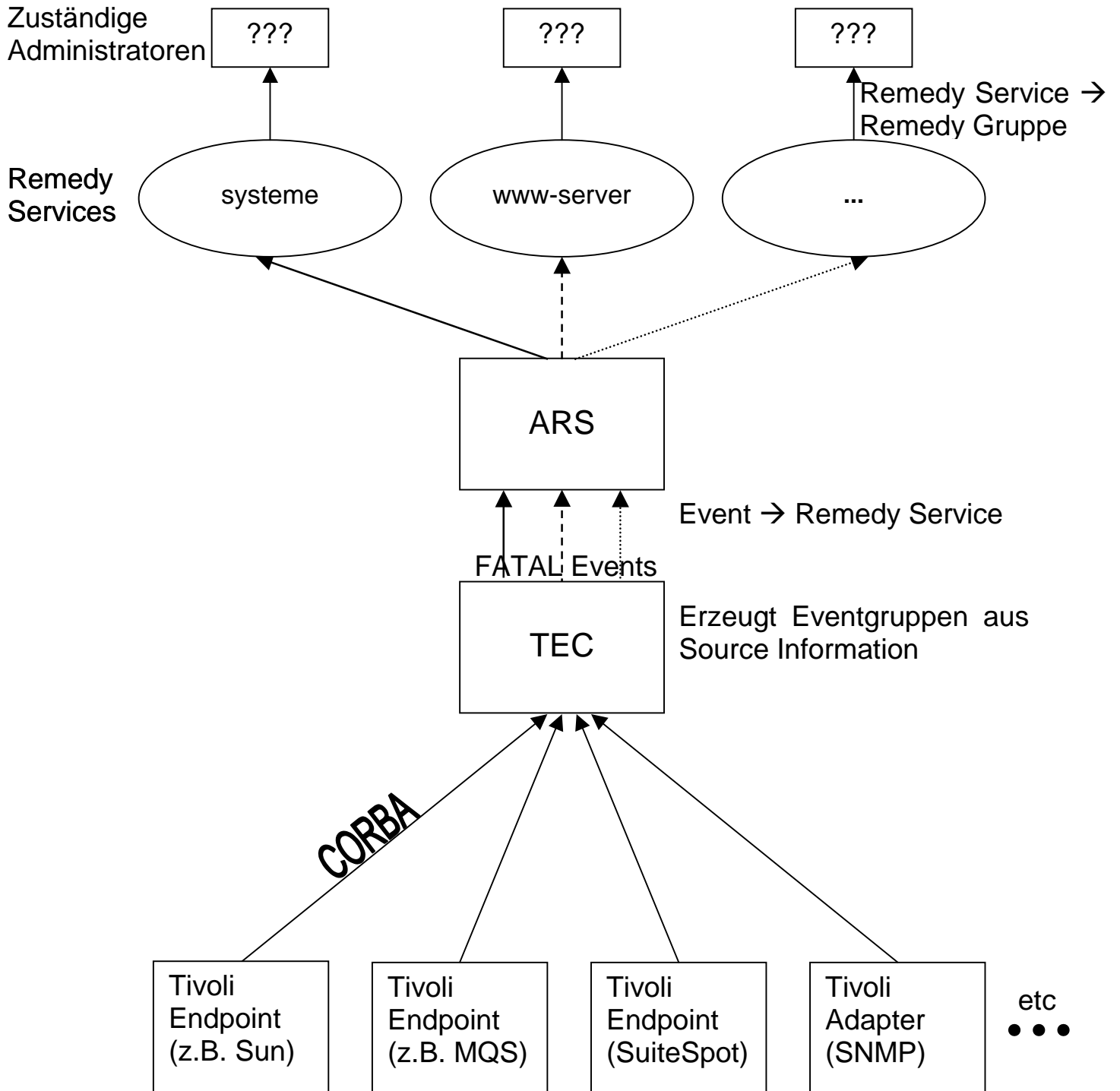
Das Skript führt einen *wcprb* (copy rule base) mit anschließendem *wcomprules* (compile rule base) aus, dabei zeigte der *wcprb* ein recht seltsames Verhalten. Er änderte die Reihenfolge der Einträge in der Datei *../TEC\_CLASSES/.load\_classes*. Indem er immer den Eintrag *tec.baroc* vor den Eintrag *root\_ext.baroc* setzte, was beim kompilieren der *Rulebase* Fehlermeldungen erzeugte da gewisse Klassen welche in *root\_ext.baroc* definiert sind *tec.baroc* noch nicht bekannt waren.

Bei *root\_ext.baroc* handelt es sich um eine für BMW entwickelte Erweiterung der standard *root.baroc*. Dieser BUG ist um so schmerzlicher, wenn man bedenkt, daß bei der Erstellung der *root\_ext.baroc* streng nach Tivoli Vorgaben vorgegangen wurde. Dieser Bug ist nicht SuiteSpot spezifisch!!!

### **3.10. Remedy Anbindung des SuiteSpot Moduls**

In der *TEC* treffen verschiedene Events (z.B. von Logfile-, Netview-, SNMP-Adapttern, Tivoli Managed Nodes, SuiteSpot Endpoints etc.) ein. Die *TEC* bietet mittels einer Prolog ähnlichen Sprache die Möglichkeiten die Events untereinander zu korrelieren bzw. auf sie zu reagieren. So ist es Beispielsweise möglich (mittels der *Remedy ARS* Schnittstelle) Trouble Tickets direkt an *Remedy ARS* zu schicken. Dazu müssen die eintreffenden *Events* auf sogenannte *Remedy Services* abgebildet werden. Vereinfacht gesprochen sind die *Remedy Services* die einzelnen Problemfelder (z.B. „rechnerbetrieb werk1 unix“ oder „desktop nt“). *Remedy Services* sind die Fehlerklassen , welche in *Remedy* definiert sind. Diese *Remedy Services* werden von *Remedy Gruppen* bearbeitet, die *Remedy Gruppen* bestehen aus den für die jeweiligen Problemfelder zuständigen Administratoren. Zum näheren Verständnis sei auf die nachfolgende Skizze und die *Remedy* bzw. *TEC* Administratoren verwiesen.

**Schaubild:**



**3.10.1. Vorgeschlagene Event-Gruppen:**

- WWW-Leitstand: enthält alle Ereignisse

- WWW-Systeme: enthält Ereignisse bis einschließlich zur BS-Ebene (Filterung nach Source und Subsource)
- WWW-Applikationen: enthält Ereignisse der Internetapplikationen (Filterung nach Source und Subsource)

Unterscheidung ist nötig, da für die Wartung/Betreuung der SuiteSpot Internetserver 2 verschiedene Gruppen zuständig sind. „Source“ ist hier *SENTRY* unter „Subsource“ werden die verschiedenen *Sentry Profiles* verstanden.

Die TEC filtert und korreliert die Events und leitet nur diejenigen welche mind. 15 min den Status fatal haben an Remedy weiter. In unserer Speziellen Umgebungen werden ausschließlich *Sentry-Events* und die des *Logfile Adapters* an die TEC weitergeleitet.

### 3.10.2. Vorschlag zur Erstellung der Remedy Services:

#### **Derzeitige Situation:**

Es existiert derzeit folgende Services:

- standard tools    intra- internet    server
- standard tools    intra- internet    zulassung

mit keinen weiteren Eintragungen. Derzeit werden beide Remedy Services von der Remedy Gruppe *Internet* bearbeitet, die hauptsächlich aus den ehemaligen Mitgliedern von FI-6 bestehen.

#### **Vorschlag:**

Einteilung in 2 Problembereiche

- Probleme von Hardware bis einschließlich zur Betriebssystemebene  
Rechnerbetrieb    intra- internet    systeme (BS etc.)
- Probleme mit den Webservern (Softwareseitig)  
Rechnerbetrieb    intra- internet    www-server

Diese Einteilung ist nötig, da abzusehen ist, daß die Betreuung von HW/BS und der darüberliegenden Applikationsebene von zwei unterschiedlichen Gruppen übernommen wird. Derzeit ist für beide Services die Gruppe „Internet“ zuständig.

### 3.10.3. Mapping der TEC Event-Gruppen auf Remedy Services:

#### **Vorschlag:**

- WWW-Leitstand → kein Durchreichen enthält alle Infos aus den beiden anderen Event Gruppen.
- WWW-Systeme → Rechnerbetrieb intra- internet systeme
- WWW-Applikationen → Rechnerbetrieb intra- internet systeme

## **4. Fazit/Zusammenfassung Abschlussbemerkung**

Meine Arbeit bestand darin mit einer Testinstallation (Webserver in Tivoli Testumgebung siehe Kapitel 2.1) zu bestimmen welche Möglichkeiten sich durch den Einsatz des TME10 Moduls für die SuiteSpot Server ergeben. Dabei beschränkte ich mich weitgehend auf die mitgelieferten Werkzeuge (Monitore, Tasks etc.). Das heißt es wurden nur die von Tivoli unterstützten Anwendungen eingebunden. Web basierte Systeme, in welchen der HTTP Server nur die Rolle einer Teilkomponente übernahm wurden nicht als einheitliches System integriert. Hier müßten viele systemspezifische Parameter überwacht werden, für welche aus verständlichen Gründen keine Standardwerkzeuge bereit stehen. D.h. je nach Anwendung ist viel Handarbeit (entwerfen und implementieren von Skripten, Monitoren etc.) nötig. Beispiele hierfür wären die DB-Schnittstelle, CGI-Skripten, sowie weitere Webserver basierte Anwendungen; wenn verfügbar das HA System.

Tivoli hat sich in ihrer Roadmap viel vorgenommen, wie z.B. die Migration des einstigen UNIX Tools TME auf die NT Plattform (ich meine hier den TMR und TEC Server). Des weiteren versucht man immer mehr Produkte zu unterstützen (fast im Monatstakt erscheinen neue TME10 Module), so daß für einzelne „vollausgereifte“ Lösungen nicht mehr viel Spielraum bleibt. Wie man stellenweise auch an dem „SuiteSpot“ Modul erkennen kann, so ist es eigentlich nur eine Weiterentwicklung des älteren NetCommander Moduls, was sich unschwer an den Skripten und Variablennamen erkennen läßt (fast alle beginnen mit NETCMDR). Was zur Folge hatte, daß es sehr allgemein gehalten ist (NetCommander diente dem Management UNIX basierten Internet Server), so läßt sich auch das SuiteSpot Modul u.a. für den Apache und andere einsetzen. Eine negative Folge davon war das spezielle Netscape Fähigkeiten/Eigenschaften nicht berücksichtigt werden bzw. nur unter NT unterstützt werden (z.B. Performance Monitoring). Die mitgelieferten Tasks und Monitore konnten jedoch in ihrer Funktionsweise und Zuverlässigkeit überzeugen.

## 5. Anhang

### Interfaces

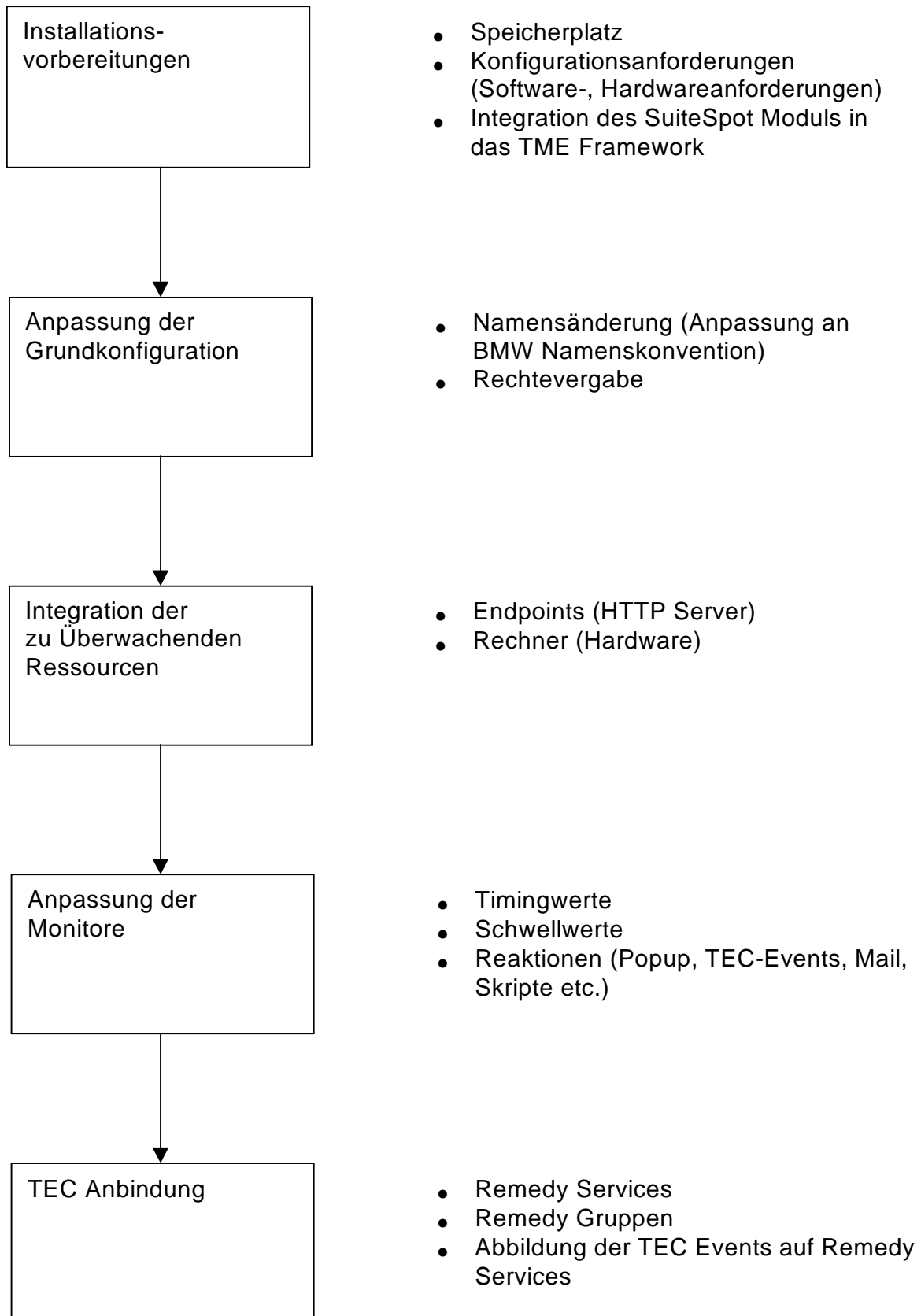
Rechnername	1. FDDI Interface	2. FDDI Interface	Ethernet Interface
wwwa	wwwa	wwwb	Intra20
wwwc	wwwc	wwwd	Intra21
wwwe	wwwe	wwwf	Inter1
intra3	Intra3	nicht vorhanden	Intra1
Watertest	Intra4	nicht vorhanden	Intra2

### Hardwarebeschreibung

wwwa	spiegelt	wwwc
	Sun E3000	
	512 MB RAM	
	2 * 250 MHZ Sun Sparc 1MB Cache	
	2 * FDDI Interface	
	1 * Ethernet Interface	
	4*9 GB HD im Network Array (Sun Photon) mittels Fibre Channel an beide Rechner angebunden.	
6 GB HD (lokal)		6 GB HD (lokal)

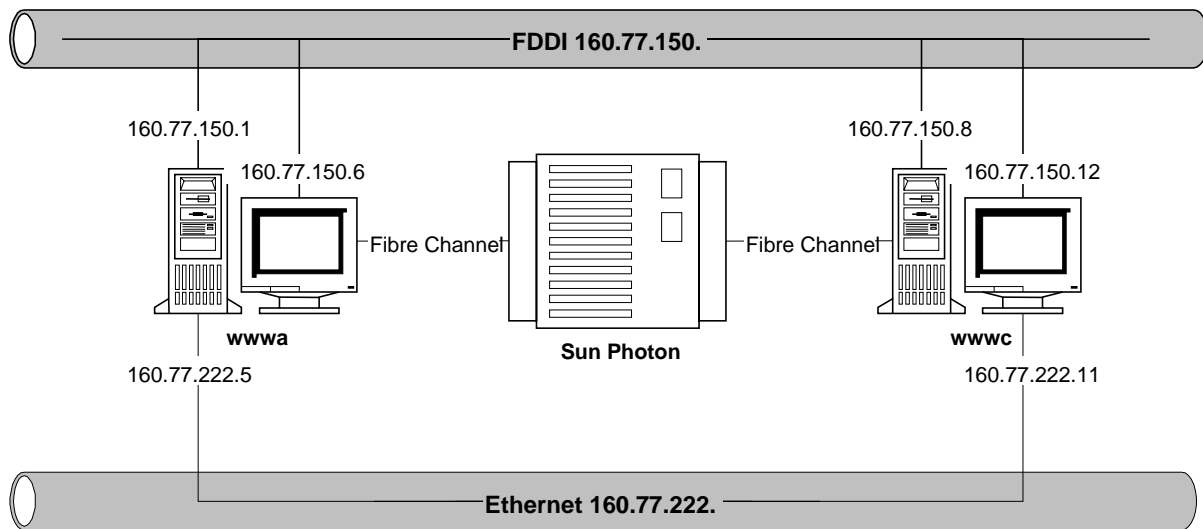
Wwwe	intra3	watertest
Sun E3000	Sun Ultra 2	Sun Ultra 2
512 MB RAM	256 MB RAM	256 MB RAM
2 * 250 MHZ Sun Sparc 1MB Cache	1 * 170 MHZ	1 * 170 MHZ
2 * FDDI Interface	1 * FDDI Interface	1 * FDDI Interface
1 * Ethernet Interface		
12 GB HD in externem Multipack mittels SCSI	24 GB HD gespiegelt extern im Multipak mittels SCSI	26 GB HD gespiegelt extern im Multipak mittels SCSI
	2 * 2 GB gespiegelt (lokal)	2 * 2 GB gespiegelt (lokal)

## Flowchart Installationsablauf





## Interfaceskizze



## Kornshellskript

```
#!/bin/ksh

#Standardvorgaben nach Till Schnupp:
TIMEOUT=20
SLEEP=30

STDOUT=/tmp/STDOUT
STDERR=/tmp/STDERR
LOGFILE=/tmp/LOGFILE

COUNT=1

#Informationen ueber zu ueberwachenden ENDPOINT mittels objcall
#auf $ENDPOINT abfragen:
CONFIG_FILE=`objcall $ENDPOINT_OID getattr config_file` #Sendet Anfrage
an den ORB und liefert den Pfad der Server Konfigurationsdatei
CONFIG_FILE=/${CONFIG_FILE#???????} #Dies ist nötig, da Tivoli Junk Bytes
in die Nutzinformation einfüegt

#Infos aus Konfigfile (bei Netscape magnus.conf) extrahieren:
PORT=`sed -n "s/^Port[ ]//p" $CONFIG_FILE`
IP=`sed -n "s/^Address[ ]//p" $CONFIG_FILE`
IP_NAME=`sed -n "s/^ServerName[ ]//p" $CONFIG_FILE`
RECHNER=$IP_NAME

#echo "Telnet auf $RECHNER mit Port $PORT"

if /usr/sbin/ping $RECHNER 5 2>/dev/null >/dev/null ; then
    SECONDS=1
    { echo GET / HTTP/1.0\r; echo \r; sleep $TIMEOUT; } |
    telnet $RECHNER $PORT >$STDOUT 2>$STDERR #Anfrage ob
Dokumentroot verfuegbar
    RTIME=$SECONDS
    if [ $RTIME -ge $TIMEOUT ] ; then
        echo 100 #Anfrage ergab ein timeout
```

```
elif [ "$(<$STDERR)" != 'Connection closed by foreign host.' ]
; then
    echo 104
    else
        case $(<$STDOUT) in
            *'<TITLE>Not Found</TITLE>') echo 99 ;; #Angefragtes
Dokument wurde nicht gefunden, aber Server reagiert
            *) echo $RTIME ;; #Anfrage erfolgreich, liefert
Antwortzeit zurueck, da die Anfrage meist lokal ist der Wert unbedeutend
        esac
        fi
        #wait
    else
        echo 105 #ping war nicht moeglich
    fi
exit 0
```