

Service Management Facility (smf)

September 2006

Wolfgang Ley
Technology Consultant MCSC
Sun Services

Die Überraschung

**SunOS Release 5.10 Version Generic_118822-27 64-bit
Copyright 1983-2005 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
Hostname: dummy**

dummy console login:

Service Management Facility (smf)

- Limitierungen der gewohnten init-Scripte
- Grundidee von smf
- Definition: Services, Instanzen, Milestones...
- Übersicht der einzelnen Befehle
- Konfiguration
- Migration existierender Scripte
- Fehlerbehebung, Notfall-Mechanismen
- Ausblick

Limitierung der alten init Scripte

- Keine echten Abhängigkeiten
 - > Service A nur starten wenn Service B läuft
- Abhängigkeiten nur über Startreihenfolge
 - > Sortierung nach Dateinamen in `/etc/rc?.d/`
- Keine Fehlerbehandlung
 - > Fehler in einem Script wird von nachfolgenden nicht erkannt
 - > Fehler im init-Script kann zu Hängern beim Boot führen
- Probleme bei Update oder Patch
 - > Eigene Änderungen werden ggf. überschrieben

Grundidee von smf

- Abhängigkeiten klar definieren und managen
- Fehler vermeiden, erkennen und beheben
 - > Fehler in Startscripten sollen keine Hänger auslösen
 - > Fehlerursache anzeigen (nicht die Auswirkung)
 - > Automatischer Service (Re-)Start
 - > Soweit sinnvoll und möglich (z.B. nach coredump)
 - > Sobald alle Abhängigkeiten erfüllt sind
- ... und das ganze noch schneller

Definition eines Services

- Services können vielfältig sein
- Ein Service ist nicht (immer) ein Daemon
- Klassen von Services
 - > Einmaliger Aufruf eines Config-Befehls
 - > z.B. bei “coreadm”
 - > Start eines Daemons
 - > z.B. bei “sshd”
 - > Start mehrerer Prozesse zur Erbringung eines Services
 - > z.B. bei Oracle Datenbank

Definition einer Instanz

- Ein Service besteht aus einer oder mehreren Instanzen
- In der Regel gibt es eine “default” Instanz
- Beispiel Webserver
 - > Der gesamte Service lautet z.B. “apache”
 - > Es kann mehrere Instanzen (z.B. für verschiedene Hostnamen oder für verschiedene Portnummern) geben

Definition Milestone

- Ein Milestone wird erreicht, wenn alle hierfür notwendigen Services laufen
- Milestones sind vergleichbar mit runlevels
 - > single-user, multi-user, multi-user-server
 - > Spezielle milestones: all, none
- Neue Bootoption -m
 - > Mit “-m milestone=none” kann ohne Services gebootet werden
 - > Ein “boot -s” ist ähnlich “boot -m milestone=single-user”

Bestandteile eines Services

- **Manifest**
 - > XML-Datei mit Beschreibung des Services und seiner Komponenten
- **Method**
 - > Information für SMF wie ein Service gestartet/gestoppt wird
 - > z.B. ein “init Script” oder gleich der Start eines Daemons
- **Properties**
 - > Konfigurationsparameter, Privileges, Timeouts, etc.
- **Logfile**
 - > Ausgaben gehen (per default) nicht auf die Console oder nach `/var/adm/messages`

Identifikation eines Services

- Fault Management Ressource Identifier: FMRI
 - > `svc://<hostname>/<servicename>:<instance>`
- Beispiele
 - > `svc://localhost/network/smtp:sendmail`
 - > `svc:/network/smtp:sendmail`
 - > `network/smtp:sendmail`
- Derzeit beschränkt auf localhost
 - > Auch Kontrolle und Administration derzeit nur auf dem eigenen Rechner lokal möglich

Befehlsübersicht zu SMF

- **svcs**
 - > Statusanzeige der Services
- **svcadm**
 - > Services managen (aktivieren, deaktivieren etc.)
 - > Milestones wechseln
- **svccfg**
 - > Services importieren oder löschen
 - > Konfiguration eines Services anzeigen oder verändern

Servicestatus

- **uninitialized**
 - > Konfiguration noch nicht eingelesen
- **disabled**
 - > Service abgeschaltet und läuft nicht
- **offline**
 - > Service wartet auf andere Abhängigkeiten
- **online**
 - > Service läuft

Servicestatus

- degraded
 - > Service läuft nur teilweise (z.B. nur 3 von 4 Webservern)
- maintenance
 - > Service hat Probleme die ein Eingreifen des Administrators notwendig machen
- legacy_run
 - > Kein SMF Service sondern ein altes /etc/rc?.d/ Script
 - > Notwendig um Kompatibilität zu gewährleisten

Implementation von SMF

- Kernel startet init Prozess
- init liest /etc/inittab und startet svc.startd
 - > svc.startd startet und überwacht Services
- svc.startd startet svc.configd
 - > svc.configd verwaltet die SMF Konfiguration
- svc.startd startet Services für den geforderten Milestone
 - > Incl. Fehlerbehandlung (z.B. sulogin an der Konsole starten)

SMF Konfigurationsdateien

- Die Service Definitionen werden als Manifest (XML Datei) ausgeliefert
 - > üblicherweise in /var/svc/manifest/
 - > Gruppierungen der Services: application, device, milestone, network, platform, site, system
- Manifeste werden in eine Repository importiert
 - > Implementation als sqlite DB in /etc/svc/
 - > Existierende Einträge werden dabei nicht überschrieben oder verändert

SMF Konfigurationsdateien

- Dokumentation der Manifest Syntax in `service_bundle(4)` Manpage
 - > Verweist aber im wesentlichen nur direkt auf die Service DTD
 - > Siehe `/usr/share/lib/xml/dtd/service_bundle.dtd.1`
- Syntaxcheck eines Manifests per `svccfg`
 - > `svccfg validate <file>`
- Import des Manifests in die Repository
 - > Manuell mittels “`svccfg import`”
 - > Automatisch beim Boot (wenn in `/var/svc/manifest/`)
 - > Metadaten Check (uid, gid, size, mtime)
 - > MD5 Checksumme des Manifests

SMF Repository Backup

- **Automatisches Backup der Repository**
 - > Bei jedem Reboot vor der ersten Änderung
 - > /etc/svc/repository-boot-<timestamp>
 - > Nach jeder Änderung durch ein Manifest Import
 - > /etc/svc/repository-manifest_import-<timestamp>
 - > Es werden je Typ bis zu 4 Backups vorgehalten
- **Manuelles Recovery via Script**
 - > /lib/svc/bin/restore_repository
 - > Bei Problemen mit der Repository wird auf dieses Script und auf /lib/svc/share/README hingewiesen (Consolen Output)

Konfiguration von smf

- Konfiguration mittels svccfg
 - > Zugriff auf die gesamte aktuelle Konfiguration
 - > Kann auch auf anderen Repositories (z.B. Backup oder Repository einer Zones) arbeiten
- Keine Änderungen in den Manifesten!
 - > Manifeste können z.B. bei Update oder Patch überschrieben werden
- Defaults aus den Manifesten und aus Profiles
 - > Manifeste liefern default Einstellungen
 - > Profiles wählen Services aus, die für das System verwendet werden sollen

Konfiguration von smf

- Drei automatische Profiles (XML Dateien)
 - > /var/svc/profile/generic.xml
 - > /var/svc/profile/platform.xml
 - > /var/svc/profile/site.xml
 - > Diese drei werden einmalig eingelesen
- Weitere Profiles möglich
 - > Siehe z.B. in /var/svc/profile/ die ns_*.xml Profiles
- Manuelle Aktivierung anderer Profiles
 - > svccfg apply <filename>
 - > Beispiel generic_limited_net.xml

Migration nach SMF

- Existierende init-Scripte können als Grundlage für SMF Methods genutzt werden
 - > Kopieren von `/etc/init.d/` nach `/lib/svc/method/`
 - > Einbindung von `/lib/svc/share/smf_include.sh`
 - > Returnwerte `SMF_EXIT_*` benutzen (siehe `smf_method(5)`)
 - > Script sollte sich erst beenden, wenn Service wirklich zur Verfügung steht
 - > Häufig kann das Script vereinfacht werden
 - > Ist eine spezielle Stop-Methode notwendig?
 - > Vorsicht bei Umstieg nach Solaris 10 (unabhängig von SMF)
 - > Zones beachten (z.B. `-z` bei `pkill`)

Migration nach SMF

- Alte init-Scripte laufen auch weiterhin
 - > Notwendig für Kompatibilität (“legacy run scripte”)
 - > /etc/rc?.d/ Scripte werden nach allen SMF Services gestartet
 - > Für bestimmte Dienste ist daher eine Konvertierung zwingend notwendig (z.B. falls ein Filesystem zur Verfügung gestellt werden soll, das andere Services benötigen)
 - > Kein Abhängigkeiten zwischen SMF Diensten und rc-Scripten möglich
 - > Keine Überwachung der rc-Scripte möglich
 - > Keine Fehlererkennung, Kein automatischer Restart
 - > Kein paralleler Start der Scripte

Erstellung eines Manifests

- Auswahl des Service Namens
 - > 7 Default Kategorien in `/var/svc/manifest/`
 - > application (high level, z.B. apache)
 - > milestone (Sammlung anderer Services, z.B. multi-user)
 - > platform (platformspezifisch, z.B. DR)
 - > system (Solaris Systemdienste, z.B. syslog)
 - > device (Support für Devices und Treiber)
 - > network (Netzwerkdienste, z.B. NFS)
 - > site (speziell für eine Organisation)
 - > Servicetypen: service, restarter, milestone

Erstellung eines Manifests

- Beispiel

```
<service  
  name='application/dummyd'  
  type='service'  
  version='1'>
```

Erstellung eines Manifests

- Anzahl der Instanzen
 - > Kann der Service mit mehreren Instanzen parallel laufen?
 - > Beispielsweise kann nur ein syslog laufen
 - > Es können aber z.B. mehrere Webserver (auf unterschiedlichen Ports) laufen
 - > Falls nein, dann einschränken mittels XML tag `<single_instance>`
 - > Dadurch wird ein Start im Fall einer Fehlkonfiguration verhindert

Erstellung eines Manifests

- Auswahl der start/stop Methoden
 - > Start Methode kann direkt einen Daemon starten oder aber ein komplexeres Script von `/lib/svc/method/`
 - > Stop Methode kann ggf. vereinfacht werden falls keine spezielle Behandlung notwendig
 - > Spezielle Methoden “:kill” und “:true”
 - > Optional eine refresh Methode
 - > Achtung: refresh darf nicht den Prozess killen, da das als Servicefehler erkannt werden würde
 - > Angabe eines Timeouts ('0' für endloses Timeout)
 - > Optional Kontext
 - > credentials, privs, working dir, ressource pools etc.

Erstellung eines Manifests

- Beispiel

```
<exec_method
  type='method'
  name='start'
  exec='/lib/svc/method/appl-dummy %m'
  timeout_seconds='60'>
  <method_context>
    <method_credential user='nobody' group='other' />
  </method_context>
</exec_method>
```

Erstellung eines Manifests

- Spezifikation der Abhängigkeiten
 - > <dependency> Attribut
- Hier liegt die eigentliche Arbeit
 - > Welche anderen Services werden benötigt?
 - > Typen “require_all”, “require_any”, “optional_all”
 - > Mit welchen anderen Services bestehen Inkompatibilitäten?
 - > Type “exclude_all”
- Abhängigkeiten können Services aber auch Dateien (z.B. Existenz einer Config-Datei) sein

Erstellung eines Manifests

- Für jede Abhängigkeit muß Fehlerbehandlung spezifiziert werden
 - > none: lediglich zum Start einmalig notwendig
 - > error: Restart wenn Abhängigkeit Fehler hat (z.B. coredump)
 - > restart: Restart wenn Abhängigkeit restarted
 - > refresh: Restart wenn Abhängigkeit refreshed
- Reverse Abhängigkeiten
 - > <dependent> Attribut falls ein Service vor einem anderen Service gestartet werden soll
 - > Dient auch zur Eingliederung in Milestones

Erstellung eines Manifests

- Beispiel

```
<!--  
  This service requires /var which might not be on /  
-->  
<dependency  
  name='filesystem'  
  grouping='require_all'  
  restart_on='error'  
  type='service'  
  <service_fmri value='svc:/system/filesystem/local' />  
</dependency>
```

Erstellung eines Manifests

- **Optional Erstellung einer Default-Instanz**
 - > Z.B. wenn keine zusätzliche SMF Config vom Admin für diesen Service notwendig ist
 - > Instanz sollte per Default auf “disabled” stehen
 - > Aktivierung über Profiles
- **Optional Hinweis auf Dokumentation**
 - > Hinweise auf Manpages oder Webseiten
 - > Zumindest aber ein vollständiger Name des Services für die Anzeige in “svcs” etc.
- **Optional Restart bei core/signal vermeiden**
 - > Achtung: gilt für alle Prozesse des Dienstes

Erstellung eines Manifests

- Existierende Manifeste als Beispiele nutzen
- Konfigbefehl (transient Service)
 - > Bsp. `/var/svc/manifest/system/coreadm.xml`
- Einfacher Daemon
 - > Bsp. `/var/svc/manifest/system/cron.xml`
- Inetd basierter Service
 - > Bsp. `/var/svc/manifest/network/ftp.xml`

Migration von inetd Diensten

- **Spezieller Support für inetd Migration**
 - > Der Befehl “inetconv” konvertiert /etc/inetd.conf in SMF
 - > Wird beim OS Upgrade automatisch ausgeführt
 - > Danach nur noch manuell
 - > Hinweis in /var/adm/messages falls inetconv notwendig ist
- **Vereinfachte Konfiguration via inetadm**
 - > Der Befehl “inetadm” zeigt alle Parameter eines inetd Services
 - > Einstellungen global (alle inetd Services) oder lokal (ein einzelner Service) möglich
 - > Z.B. Aktivierung des TCP Wrappers

Fehlersuche und -behebung

- Statuscheck mittels “svcs”
 - > Anzeige der Probleme mit “svcs -x” bzw. “svcs -xv”
 - > Anzeige aller Services mit “svcs -a”
 - > Abhängigkeiten prüfen mit “svcs -d” und “svcs -D”
- Normalerweise kein Output an der Console
 - > Services werden parallel gestartet (wenn möglich)
 - > Dadurch würden Ausgaben gemischt
- Pro Service eigenes Logfile mit Output
 - > Für Services vor Milestone single-user unter /etc/svc/volatile/
 - > Danach unter /var/svc/log/
 - > Anzeige des Pfades zum Logfile via “svcs -l”

Fehlersuche und -behebung

- **Vorsicht bei manuellem Eingriff**
 - > Prozesse können nicht einfach mit “kill” beendet werden
 - > svc.startd wird dieses als Fehler erkennen und den Service automatisch neu starten
 - > Dieses kann sich auch auf andere Services auswirken
 - > Bei Bedarf Service disablen (“svcadm disable <service>”) und dann die Daemons/Dienste manuell starten (z.B. mit Debug Optionen)
- **Tests am besten nur temporär ausführen**
 - > Siehe -t Option bei svcadm

Fehlersuche und -behebung

- **Verbose Boot**
 - > **boot -m verbose**
 - > Zeigt pro Service Statusinformation (z.B. beim Start)
 - > **boot -m debug**
 - > Zeigt alle SMF Aktionen (Lesen der Config, Auswahl der Services gemäß Abhängigkeiten etc.): viel Output
 - > Serieller Start der Services (keine Parallelisierung)
- **Besonderheit beim Boot von Zonen**
 - > Der “zoneadm reboot” Befehl unterstützt noch kein -m
 - > Alternative: options/logging SMF Property setzen
 - > Siehe Beispiel in svc.startd Manpage

Fehlerquellen

- **Abhängigkeiten nicht richtig definiert**
 - > Zugriff auf Filesysteme nicht beachtet (z.B. /usr oder /var)
 - > Abhängigkeit von Netzwerk oder Nameservices?
 - > Zyklische Abhängigkeiten
 - > Werden von SMF erkannt und gemeldet
- **Service Typ falsch definiert**
 - > Daemon vs. transient Service
- **Restart bei core/signal von Childs ok?**
- **Fehler im Service selber**
 - > Siehe jeweiliges Logfile

Notfallsituationen

- Keine Systemhänger mehr durch defekte Init-Scripte oder fehlschlagende Services
- Im Notfall root login an der Console
 - > Deutlich zuverlässiger als bislang
 - > Auch bei fehlschlagendem `svc.startd`
- Repository Recovery via Script
 - > Interaktives Script `/lib/svc/bin/restore_repository`
 - > Achtung: ggf. vorher Root-Filesystem writable remounten
- Detaillierte Informationen im Notfall
 - > Hinweis auf `/lib/svc/share/README`

Ausblick

- SMF Konfiguration und Administration kann per RBAC delegiert werden
 - > Z.B. Privileg `solaris.smf.manage` um Services zu starten oder zu stoppen (Operator Tätigkeiten)
 - > Details in Manualpage `smf_security(5)`
- Library für Entwickler
 - > Zugriff auf alle SMF Funktionen via `libscf`
- GUI in Planung
 - > Siehe OpenSolaris Portal zu `vpanels` (visual panels)

Referenzen

- Solaris Dokumentation zu SMF
 - > <http://docs.sun.com/app/docs/doc/817-1985/6mhm8o5rh?a=view>
- Sun Blueprints
 - > <http://www.sun.com/blueprints/0206/819-5150.html>
 - > <http://www.sun.com/blueprints/0605/819-2887.html>
- BigAdmin Webseite
 - > <http://www.sun.com/bigadmin/content/selfheal/>
- OpenSolaris Community
 - > <https://www.opensolaris.org/os/community/smf/>

Danke....

