SCCL2

SCript CLuster
Version 2

Das SCript CLuster 2

SCCL-0008DE-05W

SCRIPT CLUSTER

SCript CLuster ist eine eigenständige Entwicklung und steht auf der Webseite des Universal Logging System zur Verfügung.

SCRIPT CLUSTER	http://www.universal-logging-system.org/dokuwiki/doku.php?id=sccl
Projekt:	SCript CLuster

Dies Dokument wurde erstellt und ist veröffentlicht in der Hoffnung, dass es von Nutzen für den Einsatz des SCript CLusters ist. Es schließt jedoch eine Garantie auf Vollständigkeit oder den Einsatz für einen bestimmten Zweck aus.

Änderungsübersicht

Änderung	Datum	Grund / Bemerkung / Änderungen	Bearbeiter
1	2009-09-29	Erstellung SCCL	roveda
2	2016-06-28	Erstellung SCCL2	stz
3	2019-09-19	Anpassung für SCCL 2.6	stz
4	2024-01-06	Anpassung für SCCL 2.10	stz

Inhaltsverzeichnis

Abbildungsverzeichnis:

Tabellenverzeichnis:

1 Management Summary

Komplexe Anwendungen, die ggf. über mehrere Rechner und Betriebssysteme verteilt betrieben werden, erfordern einen hohen Aufwand beim Aufbau und Betrieb des resultierenden Rechnerverbundes.
Gerade Abhängigkeiten einzelner Anwendungskomponenten untereinander (z.B. des Webservice vom Datenbankservice) oder Abhängigkeiten einzelner Anwendungskomponenten von bereitzustellenden Ressourcen (z.B. Datenbankservice vom Festplattenplatz) sind in jeder Anwendung unterschiedliche und potentielle Störungsursachen für den zuverlässigen Betrieb.

SCript CLuster ist ein Programmpaket, das alle Teilkomponenten einer komplexen Anwendung in Konfigurationsdateien logisch beschreibt.
Basierend auf diesen Konfigurationsdateien wird mit den SCript CLuster Programmen die Anwendung gesteuert. Hierbei übernimmt SCript CLuster die Berücksichtigung der definierten Abhängigkeiten.

SCript CLuster bietet somit eine Transformation unterschiedlichster Anwendungen in eine einheitliche Administrations- und Steuerungsumgebung unter Auflösung der Betriebssystem-abhängigkeiten. Ein Mischbetrieb zwischen Linux und Windows ist möglich.

Die Kommunikation der Komponenten des SCript CLuster Programmpaketes erfolgt verschlüsselt.

Optional besteht die Möglichkeit, die Konfiguration, relevanten Zustandsdaten und Zustands-änderungen an das Universal Logging System (ULS) zu senden.
(siehe: http://www.universal-logging-system.org)

2 Gegenstand des Dokuments

Dieses Dokument gibt einen Überblick zum SCript CLuster und beschreibt die Installation und Konfiguration. Einsatzmöglichkeiten werden anhand von Beispielen dargestellt.

Dieses Dokument bezieht sich auf folgende Versionen:

Betriebssystem	SCript CLuster Version
Linux	2.10.
Windows (mit Cygwin)	2.10.

Tabelle 1: Referenz Betriebssystem zu SCript CLuster Version

2.1 Vereinbarungen zur Notation

Terminal Ein- und Ausgaben werden durch das Format

```
Terminal Ein- und Ausgabe
```

gekennzeichnet.

Lange Kommandos mit Zeilenumbruch werden gekennzeichnet mit:

```
Terminal Ein- und Ausgaben werden bei langen Zeilen derart dargestellt,
  dass die folgende zugehörige Zeile eingerückt dargestellt wird.
```

Ausgaben können zur besseren Darstellung gekürzt, angepasst oder gar nicht aufgeführt werden.

Kommandozeilen, die mit

```
$ ...
```

beginnen, zeigen an, dass dieses Kommando von jedem Benutzer mit Standard Betriebssystemrechten ausgeführt werden kann.

Kommandozeilen, die mit

```
# ...
```

beginnen, zeigen an, dass dieses Kommando nur vom Benutzer root oder Mitgliedern der Gruppe „sccl“ ausgeführt werden kann.

Kommandozeilen, die mit

```
C:\Program Files\sccl>
```

beginnen, müssen als Administrator auf einem Windows Server ausgeführt werden

Wird in einer Kommandozeile ein „$$“ genutzt, so wird auf den Parameter in der Datei sccl.conf verwiesen.

```
# vi $$PACKAGES
```

Mit TODO werden Dokumentabschnitte gekennzeichnet, die noch in Bearbeitung sind oder technisch noch geprüft werden müssen.

Besondere Hinweise sind farblich markiert.

3 Überblick

Komplexe Anwendungen, die ggf. über mehrere Rechner verteilt betrieben werden, erfordern einen hohen Aufwand beim Aufbau und Betrieb des resultierenden Rechnerverbundes (=Cluster). Um dies zu vereinfachen, wurde das SCript CLuster (SCCL) entwickelt.

Beim Aufbau eines SCript CLusters werden die beteiligten Rechner (=Clusternodes) und die Clusterpakete definiert. Zu den Clusterpaketen wird festgelegt, auf welchen Clusternodes sie laufen können und welche Ressourcen und Start- und Stoppskripte dazugehören.

Die Administration und der Betrieb des Clusters werden durch Kommandos vereinfacht, mit denen Clusterpakete gestartet und gestoppt werden können. Durch diese Vereinfachung sind auch „projektfremde“ Administratoren in der Lage, das Cluster zu bedienen. Definierte Abhängigkeiten von Clusterpaketen zu anderen Clusterpaketen werden dabei entsprechend berücksichtigt.

Durch die Vereinfachungen beim Einsatz des SCript CLusters ergibt sich eine effektive Verbesserung der Verfügbarkeit des gesamten Rechnerverbundes mit seinen Anwendungen.

Das SCript CLuster

- definiert in Konfigurationsdateien die Clusterkomponenten
- bildet Abhängigkeiten von Clusterkomponenten untereinander ab
- kann virtuelle IP-Adressen starten und stoppen
- kann SAN-, NAS-, NFS- und FS-Partitionen an- und abdocken
- startet und stoppt Clusterkomponenten über Kommandoskripte
- verhindert Ressourcenkonflikte beim Start von Clusterkomponenten
- kommuniziert zwischen den Clusternodes über verschlüsselte https Verbindungen
- führt jedoch selbstständig KEINE Überwachung im laufenden Betrieb durch und somit erfolgt auch KEIN Neustart noch automatischer Schwenk von Clusterkomponenten. Für diese Anforderung stellt SCript CLuster aber geeignete Clusterressourcen zur Verfügung, die genutzt werden können (siehe Kapitel 5.3.2.8, „Testskripte für die Clusterpaketüberwachung“).

Voraussetzung für den Einsatz von SCript CLuster

- benötigt den Unix2Web Server (u2webtools) zur Kommunikation im Cluster (siehe http://www.universal-logging-system.org/dokuwiki/doku.php?id=unix2web)
- die beteiligten Server müssen sich über einen definierten Port untereinander erreichen
- die Namensauflösung muss auf allen Clusternodes möglich sein
- (optional) kann SCript CLuster die relevanten Informationen an das Universal Logging System (ULS) schicken (siehe http://www.universal-logging-system.org/dokuwiki/doku.php?id=uls)

Siehe auch Kapitel 8, „Anwendungsbeispiele“.

3.1 Architektur

Schematische Übersicht der Architektur eines SCript CLusters.

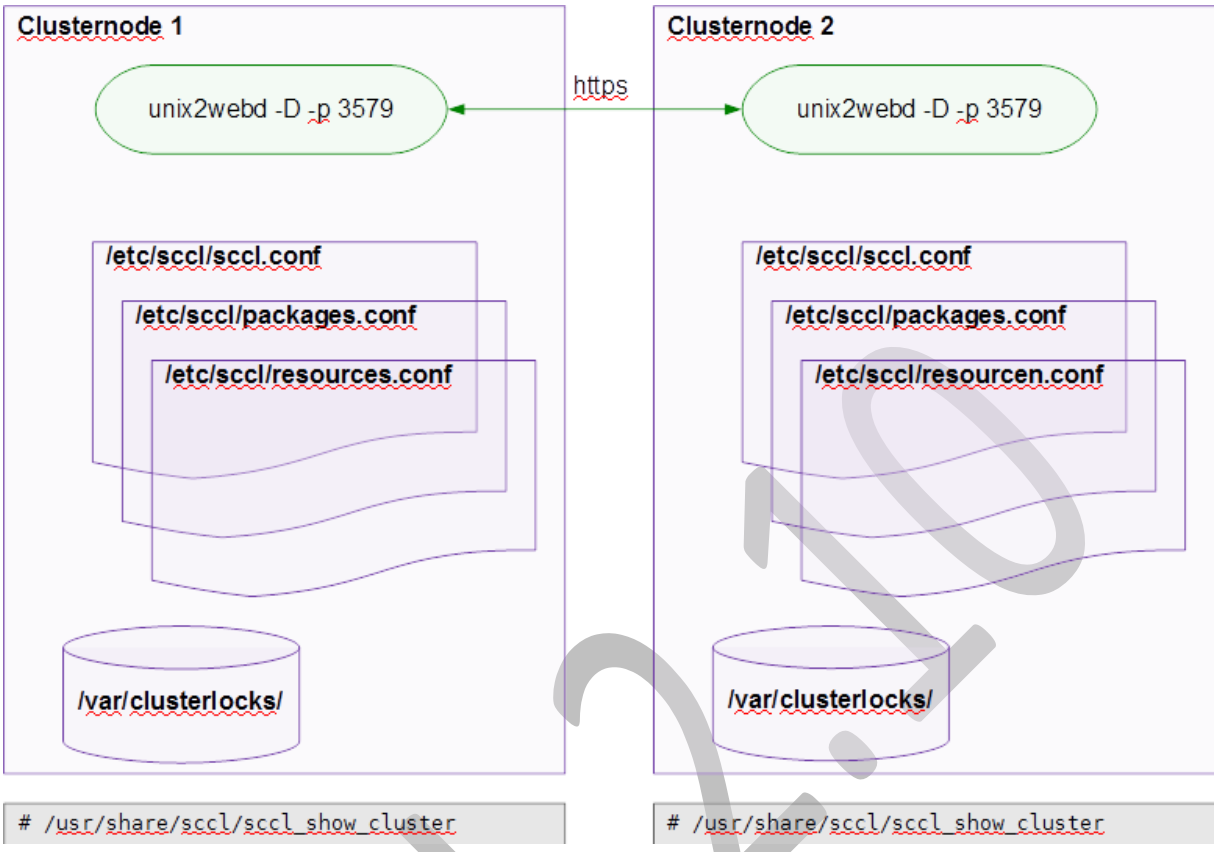

Abbildung 1: Architekturüberblick

Es werden zwei Clusternodes mit

- den Prozessen zur Clusterkommunikation,
- den Standard-Konfigurationsdateien,
- dem Verzeichnis, in dem die aktiven Clusterpakete und Ressourcen registriert werden und
- eine Kommandozeile

dargestellt.

Hinweis: Die dargestellten Pfadangaben sind die Default Pfade und können geändert werden.

3.2 Begriffe des SCript CLusters

Eine kurze Übersicht der verwendeten Ausdrücke für die verschiedenen Clusterkomponenten.

3.2.1 Cluster

Das Cluster definiert einen Rechnerverbund aus einem oder mehreren Rechnern.

3.2.2 Clusternode, Node, Knoten

Ein am Cluster beteiligter Rechner ist ein Clusternode.

3.2.3 Clustermasternode, Master

Ist der erste Clusternode eines Clusters, es ist ein normaler Clusternode, zusätzlich hält er aber die Schlüssel für die verschlüsselten Verbindungen.

3.2.4 Clusterpaket, Paket

Ein Clusterpaket ist die kleinste logische Einheit der im Cluster abzubildenden Anwendung.

Ein Clusterpaket wird immer einem (oder mehreren) Clusternodes zugeordnet, auf denen es ausgeführt werden kann.

Beispiele für Clusterpakete sind „Datenbankservice“ oder „Webservice“.

3.2.5 Clusterressource, Ressource

Clusterressourcen sind beschreibende Objekte für Clusterpakete und können wie folgt zusammengefasst werden:

- physikalische Komponenten eines Clusterpakets (z.B. Partitionen)
- logische Komponenten eines Clusterpakets (z.B. IP Adressen)
- Verweise auf andere Clusterpakete inkl. Abbildung der Abhängigkeiten
- Programme oder Skripte zum Starten und Stoppen des Clusterpakets

3.2.6 Clusterkommunikation

Die Kommunikation zwischen den Clusternodes erfolgt über die unix2webd Prozesse, dafür wird ein dedizierter, im Cluster einheitlicher TCP-Port verwendet.

Die Kommunikation erfolgt nur bei Bedarf, d.h. wenn explizit Kommandos ausgeführt werden. Es erfolgt keine permanente Kommunikation, die Kommunikation im Cluster erfolgt grundsätzlich verschlüsselt.

3.2.7 Singlenodecluster

Das SCCL Cluster besteht nur aus einem Knoten.

3.3 Dateiablage

Die ausführbaren Kommandos des SCript CLusters befinden sich im Pfad

Linux	/usr/share/sccl
Windows	Die Dateien liegen in dem Verzeichnis, in dem das Programmpaket entpackt wurden.

Tabelle 2: Dateiablage der Skripte

Die Konfigurationsdateien des SCript CLusters befinden sich im Pfad

Linux	/etc/sccl
Windows	\etc\sccl ist ein Unterverzeichnis zu dem Verzeichnis, in dem das Programmpaket Installiert wurde, default: C:\Program Files\sccl

Tabelle 3: Dateiablage der Konfigurationsdateien

Die Logbücher des letzten Start/Stop eines Paketes befinden sich im Pfad

Linux	/var/tmp
Windows	\var\tmp ist ein Unterverzeichnis zu dem Verzeichnis, in dem das Programmpaket Installiert wurde, default: C:\Program Files\sccl

Tabelle 4: Logbücher Paket Start/Stop

In der nachfolgenden Dokumentation wird nur die LINUX Variante beschrieben.

4 Installation

In diesem Kapitel ist die Installation des SCript CLusters unter verschiedenen Betriebssystemen beschrieben.

4.1 Voraussetzungen

Es wird das Paket Unix2Web zum Betrieb des SCript CLusters vorausgesetzt.
Siehehttp://www.universal-logging-system.org/dokuwiki/doku.php?id=unix2web für weitere Informationen und Download.

SCCL2 ist nicht kompatibel zu älteren SCCL Version, vor einer Installation muss die ältere SCCL Version deinstalliert werden, die alte Konfigurationsdatei sccl.conf muss manuell gelöscht werden. Ggf. vorhandene Konfigurationsdateien für Clusterpakete und Clusterressourcen können übernommen werden.

4.2 Installationsanleitungen

4.2.1 Downloadquelle

SCCL2 wird als Datei bereitgestellt,
siehe http://www.universal-logging-system.org/dokuwiki/doku.php?id=sccl

4.2.2 Linux (rpm)

Die rpm-Datei des SCript CLusters (sccl2-*.noarch.rpm) wird auf den Zielrechner kopieren, anschließend erfolgt die Installation des Paketes als root:

```
# rpm -Uvh sccl2*.rpm
```

4.2.3 Linux (debian based systems)

Die deb-Datei des SCript CLusters (sccl2-*_all.deb) wird auf den Zielrechner kopieren, anschließend erfolgt die Installation des Paketes als root:

```
# dpkg -i sccl2*.deb
```

4.2.4 Windows (mit Cygwin)

Nach der Bereitstellung des Paketes „WIN-sccl2-*.zip“ kann dieses in ein beliebiges Verzeichnis des Zielrechners entpackt werden. Für die Installation des Paketes muss nur das Kommando „setup.cmd“ ausgeführt werden, welches dann SCCL nach „C:\Program Files\sccl“ kopiert.

Hinweis: Unter Windows wird SCript CLuster in einer CYGWIN Umgebung ausgeführt. Diese wird gemeinsam mit dem SCript CLuster bereitgestellt.

4.3 Betriebsvoraussetzungen

Bei der Nutzung von SCCL muss sichergestellt sein, dass die Services, die im SCCL verwaltet werden, nicht beim Starten der Rechner automatisch gestartet werden.

Somit muss

- bei Windows der Dienst auf „manuell“ oder „disabled“ gestellt werden.
- bei Linux mit init der Dienst im init.d ausgetragen werden.
- bei Linux mit systemd der Dienst disabled werden.

4.4 Einrichten des Clusters

Die Kommunikation zwischen den Clusternodes des Clusters erfolgt grundsätzlich verschlüsselt.

Hieraus ergibt sich, dass auf dem ersten Clusternode (Master) der private Schlüssel und (optional) das Passwort des CA-Keys der CA vorhanden sind.

Nachfolgend wird beispielhaft die normale Einrichtung eines Clusters beschrieben, Details finden sich unter 7.2, „allgemeines Kommando sccl“

4.4.1 Der erste Clusternode

Zum Erzeugen eines Clusters muss auf dem ersten Clusternode das Kommando

```
# sccl create_cluster -P <cluster_name>
```

Windows:

```
C:\Program Files\sccl> create_cluster -P <cluster_name>
```

ausgeführt werden.

Beim Anlegen des Clusters wird der Clusternode zum Clustermasternode und es wird die CA für die Zertifikate für die verschlüsselte Kommunikation angelegt.

4.4.2 Alle weiteren Clusternodes

Jeder Clusternode muss für den Beitritt zum Cluster das Kommando

```
# sccl join_cluster
```

Windows:

```
C:\Program Files\sccl> join_cluster
```

ausführen.

Hierdurch wird ein Webservice für den Empfang der Clusterkonfiguration gestartet.

Auf dem Clustermasternode wird mit dem Kommando

```
# sccl add_node <Name des neuen Nodes>
```

Windows:

```
C:\Program Files\sccl># sccl add_node <Name des neuen Nodes>
```

der neue Clusternode dem Cluster hinzugefügt.

Der neue Clusternode empfängt anschließend die Clusterkonfiguration und die Zertifikate.
Dieses Vorgehen muss für jeden neuen Clusternode wiederholt werden.

4.5 (optional) Anbindung SCCL an ULS

ULS ist die Abkürzung für Universal Logging System.

Details siehe http://www.universal-logging-system.org/dokuwiki/doku.php?id=uls

Optional kann das SCript CLuster die Konfigurationen und Statusberichte an ULS übergeben

```
# sccl 2_uls enable
C:\Program Files\sccl\bin> sccl 2_uls enable
```

Voraussetzung

- unter Linux muss ein konfigurierter ULS Client eingesetzt werden.
- unter Windows muss der im SCCL2 Paket enthaltene ULS Client konfiguriert werden.

siehe auch: 7.6, „Monitoring mit ULS“

5 Konfiguration des SCript Clusters

Hinweis: Für alle Programme/Services, die unter die Kontrolle von SCript CLuster gestellt werden, muss das automatische Starten verhindert werden.

Die Konfiguration erfolgt in drei Konfigurationsdateien (siehe „Tabelle 3: Dateiablage der Konfigurationsdateien“).

Linux:

```
# cd /etc/sccl/
```

Windows:

```
C:\Program Files\sccl> cd C:\Program Files\sccl\etc\sccl
```

5.1 Grundkonfiguration (sccl.conf)

Die Grundkonfiguration in der Datei sccl.conf legt u.a. den Namen des Clusters und die beteiligten Rechnernamen fest.

Achtung: Diese Datei wird von Kommandoskripte gesourced, beim Editieren also bitte aufpassen.

```
# vi sccl.conf
```

```
#!/bin/bash
# Definition der Variablen fuers Script-Cluster

# Clustername
CLUSTER=MYCLUSTER

# Physische Nodes im Cluster
NODES="clnode1 clnode2"
HBNODE_EXT="_hb"
#HBNODES="clnode1_hb clnode2_hb"
HBNODES=""
#
#
# Master, die diesen Knoten steuern, aber nicht von
# diesem gesteuert werden.
# MASTERNODES
MASTERNODES=""
#
#
# Slaves, die von diesen Knoten gesteuert werden, aber
# diesen Knoten nicht steuern duerfen.
# SLAVENODES
SLAVENODES=""
#
#
# Wartezeit in Sekunden, die nach einem Reboot gewartet
# wird, bevor die Cluster-Pakete gestartet werden.
# (gilt nicht fuer Windows)
```

```
# SLEEP_AFTER_REBOOT
SLEEP_AFTER_REBOOT=30
#
#
# Pakete bei sccl_show_cluster sortieren
SHOWSORTED=1#
# Start und Stopp-Zeiten ans ULS liefern?
#ULS="irgendwas" -> Datenlieferung an ULS aktiv
#ULS=""          -> Datenlieferung an ULS nicht aktiv
ULS=""
#
#############################################################
# Verzeichnis zum Speichern der Cluster-Lockdateien
LOCKDIR=/var/clusterlocks
#
# Datei mit den Paketen des Clusters
PACKAGES=/etc/sccl/packages.conf
#
# Datei mit den Resourcen des Clusters
RESOURCES=/etc/sccl/resources.conf
#
# Directory mit den Start-Stop und Test-Skripten der Ressourcen
STARTSTOPDIR=/etc/sccl/scripts
#
# Installationsverzeichnis
SCCLBASEDIR=/usr/share/sccl
#
# Einstellungen fuer die Clusterkommunikation ueber unix2web
U2WPORT=3579
U2WPWDDAT=/etc/sccl/u2w_pwd.dat
U2WHOSTSDAT=/etc/sccl/u2w_hosts.dat
#
# SSL
U2WCA=/etc/sccl/certs/${CLUSTER}-ca.pem
U2WCERT=/etc/sccl/certs/certs/${CLUSTER}-$(hostname | cut -f 1 -d . | tr
   '[:upper:]' '[:lower:]')-cert.pem
U2WKEY=/etc/sccl/certs/private/${CLUSTER}-$(hostname | cut -f 1 -d . | tr
   '[:upper:]' '[:lower:]')-key.pem
```

Parameter	Bedeutung
CLUSTER	Der Name des Clusters, wird bei der Ausgabe von sccl show_cluster angezeigt Anmerkung: im laufenden Betrieb darf der Clusternamen nicht geändert werden, weil sich daraus die Namen der Zertifikate ableitet
NODES	Liste der am Cluster beteiligten Rechnernamen. Alle Clusternodes müssen in der Lage sein, die Rechnernamen zu IPs aufzulösen. Anmerkung: die Zeile wird durch sccl add_node gepflegt und sollte nicht händisch verändert werden, weil dann die Zertifikate fehlen
HBNODE_EXT	optional Erweiterung für die Rechnernamen aus NODES. Über diese Rechnernamen erfolgt die interne Clusterkommunikation. Alle Clusternodes müssen in der Lage sein, die Rechnernamen zu IPs aufzulösen. Sinnvoll z.B. wenn die Clusternodes über zwei Netzwerkkarten verbunden sind.
HBNODES	optional, aber zwingend, wenn HBNODE_EXT gesetzt ist. Liste der Rechnernamen, resultierend aus NODES mit HBNODE_EXT.
SLEEP_AFTER_REBOOT	Wartezeit, bevor das SCCL Cluster nach einem Reboot gestartet wird (nur Linux Server)
ULS	Wenn nicht "", übergibt SCCL seine Informationen an ULS. Wird durch sccl 2_uls verwaltet
LOCKDIR	Verzeichnis, in dem die Lock-Dateien für die Ressourcen des Clusternodes angelegt werden. Das Verzeichnis ist clusterweit einheitlich.
PACKAGES	Konfigurationsdatei der Clusterpakete zu dem Cluster.
RESOURCES	Konfigurationsdatei der Ressourcen zu den Clusterpaketen.
STARTSTOPDIR	Verzeichnis, in dem Startskripte für die Ressource PRG abgelegt werden, wenn der Aufruf nicht mit absolutem Pfad erfolgt. Liegt das Verzeichnis nicht unter /etc/sccl, wird es von sccl dist_config nicht verteilt.
SCCLBASEDIR	Installationsverzeichnis für SCript CLuster Default /usr/share/sccl
U2WPORT	U2W Port für die Clusterkommunikation, Default 3579
U2WPWDDAT	Passwortdatei für die unix2web Kommunikation
U2WHOSTSDAT	Hosttabelle (IPs) für die unix2web Kommunikation
U2WCA	Cluster CA, die beim sccl create_cluster erzeugt wird
U2WCERT	Cluster Zertifikat für den aktuellen Clusternode. Wird vom sccl add_node erzeugt.
U2WKEY	privater Key für das Cluster Zertifikat für den aktuellen Clusternode. Wird vom sccl add_node erzeugt.

Tabelle 5: Parameter des Clusters (sccl.conf)

5.2 Konfiguration der Clusterpakete (default: packages.conf)

Die Clusterpakete werden in der Datei, die durch PACKAGES in der sccl.conf angegeben ist, definiert. Als Standard ist /etc/sccl/packages.conf festgelegt. Sie beinhaltet eine Tabelle mit allen Clusterpaketen des Clusters und die Clusternodes, auf denen sie gestartet werden dürfen.

Beispiel:

```
# vi $$PACKAGES
package1   clnode1 clnode2
package2 - clnode2 clnode1
package3 + clnode3
```

Jede Zeile hat die Form:

<CLUSTERPACKAGE> { - | + | <stdnode> } [<altnode1> <altnode2> ...]

Das Clusterpaket <CLUSTERPACKAGE> darf auf <stdnode> und allen <altnodeX> laufen.

Hierbei ist <stdnode> der Knoten, auf dem das Paket defaultmäßig laufen soll. Das Paket wird beim Booten des Clusternodes gestartet, wenn es nicht auf <altnodeX> läuft.

Ein '-' anstelle von <stdnode> verhindert das automatische Starten des Clusterpakets.

Ein ‚+' anstelle von <stdnode> startet das Paket, falls es vor dem Reboot aktiv war.

Soll ein Paket auf mehreren Servern automatisch gestartet werden, so sind diese durch ein Komma getrennt als <stdnode> einzutragen.

Hinweis: Die Pakete werden von oben nach unten gestartet und von unten nach oben gestoppt, wenn ein Clusternode bootet

Es gibt die Möglichkeit, Pakete in Gruppen zusammenzufassen. Eine Gruppe beginnt mit [<Gruppenname>].

5.3 Konfiguration der Ressourcen (default: resources.conf)

Die Ressourcenkonfiguration befindet sich in der Datei, die durch RESOURCES in der sccl.conf angegeben ist, als Standard ist /etc/sccl/resources.conf festgelegt. Sie beinhaltet eine Tabelle mit allen Clusterressourcen zu den Clusterpaketen.

Beispiel:

```
# vi $$RESOURCES
package1  package1_beschreibung PRG:startstop_package1
package2  package2_beschreibung PRG:startstop_package2
```

Es können auch Unterresourcen mit Ressourcen definiert werden. Diese können wiederum als Ressource in Clusterpaketen verwendet werden. Sie können nicht alleinstehend gestartet oder gestoppt werden, sie werden nicht im Cluster (z.B. mit sccl show_cluster) angezeigt und deren Status kann nicht abgefragt werden.

Allgemeiner Aufbau der Definitionszeile für ein Clusterpaket:

<CLUSTERPACKAGE> <description_no_spaces> [MULTI]|[EXECUTE]|[SWITCH] <resource> SETSTATE <resource> ...

Beim Starten werden die Ressourcen in der Reihenfolge, also von links nach rechts, "aktiviert".

Beim Stoppen werden die Ressourcen in umgekehrter Reihenfolge, also von rechts nach links, wieder "deaktiviert".

<resource> kann auch ein Unterpaket sein:

<subpackage> <description_no_spaces> <resource> <resource> ...

siehe hierzu Kapitel 5.3.2.7.1, „Unterressource verwenden (Ressource: RS)“ und Kapitel 8.5.3, „Realisierung (per RS)“

<CLUSTERPACKAGE>

Eindeutiger Name für ein Clusterpaket. Der Name korrespondiert mit dem in der $$PACKAGES definierten Clusterpaket. Für jedes Clusterpaket darf nur eine Zeile verwendet werden.

<description_no_spaces>

Ausführliche Beschreibung für das Clusterpaket. Darf keine Leerzeichen enthalten, dafür können '_' verwendet werden.

<resource>

Definition einer Clusterressource, die möglichen Parameter werden unter 5.3.2, mögliche Ressourcen beschrieben.

<subpackage>

Eindeutiger Name für eine Unterressource.

MULTI

Das Clusterpaket darf gleichzeitig auf mehreren Clusternodes laufen.

MULTI muss als erste Ressorce angegeben werden

EXECUTE

Das Paket wird nur ausgeführt, aber nicht gestartet oder gestoppt. Es erlangt also nie den Status „running“.

EXECUTE muss als erste Ressource angegeben werden

SWITCH

Wenn der Knoten mit dem laufenden Paket gestoppt wird, wird das Paket auf dem nächsten Knoten gestartet.

SWITCH muss als erste Ressource angegeben werden

SETSTATE

Beim Abarbeiten der Ressourcen wird SCript CLuster angewiesen, den Status „Running“ zu schreiben.

Hierdurch können gegenseitige Abhängigkeiten von Paketen aufgelöst werden.

SETSTATE sollte hinter der letzten wesentlichen Ressource angegeben werden, spätestens aber vor der Ressource „RST“.

TEARDOWN

Das Paket wird wieder beendet, liefert aber OK, als ob das Paket gestartet wurde.

Wie EXECUTE, aber es wird gestartet und wieder gestoppt.

TEARDOWN muss als letzte Ressource angegeben werden.

Beispiel siehe 8.7 „Datenübernahme“

5.3.1 Hinweise zur Bearbeitung von $$RESOURCES

Hinweis: Änderungen an der Konfiguration einer Clusterressource gelten immer erst ab dem nächsten Start des betroffenen Clusterpaketes.

Hinweis: bei der Bearbeitung der $$RESSOURCES ist die Großschreibung zu beachten

5.3.2 mögliche Ressourcen

Wegen der Komplexität der Ressourcen sei an dieser Stelle auf die Beispiele verwiesen:

- 8.1, „Eine Datenbank auf zwei Servern (Cold Standby)“
- 8.2, „Datenbank und Applikation auf verschiedenen Servern“
- 8.3, „Datenbank und Applikation auf einem Server“
- 8.4, „Anbindung Windows Share an Oracle Datenbankservice“
- 8.5, „Datenbankserver mit mehreren Applikationsservern“
- 8.6, „Zwei Datenbankinstanzen alternativ nutzen“
- 8.7 „Datenübernahme“
- 8.8 „Galera Cluster“

5.3.2.1 Skript ausführen (Ressource: PRG)

Definition	**PRG:[<path>]<file>**
Aktion beim Start	Skript <file> in <path> wird mit dem Parameter "start" aufgerufen. Wird <path> nicht angegeben, wir das Verzeichnis $$STARTSTOPDIR aus der sccl.conf verwendet.
Aktion beim Stopp	Skript <file> in <path> wird mit dem Parameter "stop" aufgerufen. Wird <path> nicht angegeben, wir das Verzeichnis $$STARTSTOPDIR aus der sccl.conf verwendet.
Beispiel	PRG:/bin/echo
Hinweis	liefert das Skript beim Start einen Exitcode ungleich 0, ist der Start des Clusterpaketes fehlgeschlagen und die Ressourcen werden wieder abgebaut.

Tabelle 6: Ressourcen: Skript ausführen

5.3.2.2 Skript mit Parametern ausführen (Ressource: PRGP)

Wie Ressource PRG; aber mit Aufrufparametern

Definition	**PRGP:[<path>]<file>[:Parameter1][:Parametern]**
Aktion beim Start	Skript <file> in <path> wird mit dem Parameter "start" und den nachfolgenden Parametern aufgerufen. Ist kein Parameter angegeben, wird der Paketname übergeben. Wird <path> nicht angegeben, wir das Verzeichnis $$STARTSTOPDIR aus der sccl.conf verwendet.
Aktion beim Stopp	Skript <file> in <path> wird mit dem Parameter "stop" und den nachfolgenden Parametern aufgerufen. Wird <path> nicht angegeben, wir das Verzeichnis $$STARTSTOPDIR aus der sccl.conf verwendet.
Beispiel	PRGP:/bin/echo:Textausgabe
Hinweis	liefert das Skript beim Start einen Exitcode ungleich 0, ist der Start des Clusterpaketes fehlgeschlagen und die Ressourcen werden wieder abgebaut.

Tabelle 7: Ressourcen: Skript mit Parametern ausführen

5.3.2.3 Netzwerkparameter

5.3.2.3.1 IP-Adresse setzen (Ressource: IP)

Definition	**IP:{<interface>|AUTO}:<ip_address>:{<netmask>|<netbits>}[:ROUTE]**
Aktion beim Start	Setzt auf der Schnittstelle <interface> eine sekundäre IP Adresse <ip_address> mit der Netzmaske <netmask>. Wird als Schnittstelle AUTO angegebene, dann wird anhand der IP-Adresse <ip_address> und der Netzmaske <netmask> eine passende Schnittstelle gesucht. Ist [:ROUTE] angegeben, dann wird die virtuelle IP-Adresse <ip_address> als Source-Adresse in den Routingtabellen eingetragen.
Aktion beim Stopp	Ist [:ROUTE] angegeben, dann wird die sekundäre IP-Adresse <ip_address> als Source-Adresse in den Routingtabellen ausgetragen. Die IP-Adresse <ip_address> wird auf der verwendeten Schnittstelle deaktiviert.
Beispiel	IP:eth0:10.1.17.103:255.255.255.0 IP:AUTO:10.1.12.122:24

Tabelle 8: Ressourcen: IP Adresse setzen

Hinweis: diese Ressource steht unter Windows nicht zur Verfügung

5.3.2.4 Paketkontrolle

5.3.2.4.1 Restart eines Pakets (Ressource: RST)

Definition	**RST:<package1>[,<package2>]...[:CLUSTER]**
Aktion beim Start	Die aufgeführten Clusterpakete <packageX> werden auf dem aktuellen Clusternode gestoppt und gestartet, wenn sie zu dem Zeitpunkt gestartet sind. Ist CLUSTER angegeben, dann werden diese Pakete im Cluster neu gestartet (da, wo sie zuletzt liefen).
Aktion beim Stopp	Diese Pakete <packageX> werden auf dem aktuellen Clusternode gestoppt oder auf allen Clusternodes, wenn CLUSTER angegeben ist.
Beispiel	RST:Applikationsserver,Webserver:CLUSTER

Tabelle 9: Ressourcen: Restart eines Paketes

Die <packageX> werden beim Stoppen von links nach rechts gestoppt und beim Starten von rechts nach links gestartet.

Anmerkung: wenn das Cluster aufgebaut ist, und erstmalig das Clusterpaket, welches RST verwendet, gestoppt wird, wird RST nicht sofort angewendet. Dieses erfolgt beim nächsten Start des Clusterpaketes, welches RST verwendet.

Ursache: Änderungen an der Konfiguration einer Clusterressource gelten immer erst ab dem nächsten Start des betroffenen Clusterpaketes.

Empfehlung: Vor dem Bearbeiten der Clusterressourcen die beteiligten Clusterpakete stoppen.

5.3.2.4.2 Clusterpaket muss im Cluster aktiv sein (Ressource: CPKG)

Definition	**CPKG:<other_package>[:WAIT[:<m>]|START|WAITSTART[:<m>]]**
Aktion beim Start	Nur wenn das Clusterpaket <other_package> im Cluster gestartet ist, wird das aktuelle Clusterpaket gestartet. Mit [:WAIT] wird auf das Clusterpaket gewartet. <m> ändert Wartezeit in Minuten, Default 5. Mit [:START] wird ggf. des Clusterpakets <other_package> gestartet. Mit [:WAITSTART], wird auf das Clusterpaket gewartet. <m> ändert Wartezeit in Minuten, Default 5. Anschließend wird ggf. das Clusterpaket gestartet.
Aktion beim Stopp	keine
Beispiel	CPKG:database:WAIT

Tabelle 10:Ressourcen: Clusterpaket muss im Cluster laufen

Hinweis: die Nutzung von „START“ sollte sehr sparsam eingesetzt werden! WAITSTART sollte vorgezogen werden. Am Besten ist WAIT (und sich dann überlege, was man da macht ;-)

5.3.2.4.3 Clusterpaket darf im Cluster nicht aktiv sein (Ressource: !CPKG)

Definition	**!CPKG:<other_package>[:STOP]**
Aktion beim Start	Nur wenn das Clusterpaket <other_package> im Cluster nicht gestartet ist, wird das aktuelle Clusterpaket gestartet. Mit [:STOP] wird das Clusterpaket <other_package> gestoppt.
Aktion beim Stopp	keine
Beispiel	!CPKG:database_backup:STOP

Tabelle 11:Ressourcen: Clusterpaket darf nicht im Cluster laufen

5.3.2.4.4 Eines der Clusterpakete muss im Cluster laufen (Ressource: CPKGO)

Definition	**CPKGO:<name1>,<name2>,...[:WAIT[:<m>]]**
Aktion beim Start	Nur wenn mindestens eins der Clusterpakete <namex> im Cluster gestartet ist, wird das aktuelle Clusterpaket gestartet. Mit [:WAIT] wird gewartet, bis eines der Clusterpakete gestartet wurde. <m> ändert Wartezeit in Minuten, Default 5.
Aktion beim Stopp	
Beispiel	Tomcat Tomcat CPKGO:Apache1,Apache2 PRG tomcat.sh

Tabelle 12: Ressourcen: eines der Clusterpakete muss laufen

5.3.2.4.5 Clusterpaket muss n Sekunden im Cluster aktiv sein (Ressource: CPKGW)

Definition	**CPKGW:<name>:<sek>[:<m>]**
Aktion beim Start	Es wird gewartet, bis Paket <name> mindestens <sek> Sekunden läuft. Das Warten wird nach <m> Minuten abgebrochen. Ist <m> nicht angegeben, wird nach 90 Minuten abgebrochen.
Aktion beim Stopp	Keine
Beispiel	CPKGW:Datenbank:30

Tabelle 13:Ressourcen: Clusterpaket muss im Cluster x Sekunden laufen

5.3.2.4.6 Clusterpaket ist Service für andere Pakete (Ressource: CSRV)

Definition	**CSRV:<name1>,<name2>,...**
Aktion beim Start	
Aktion beim Stopp	Clusterpaket wird nicht gestoppt, wenn eines der Pakete <namex> noch läuft
Beispiel	Apache Apache :PRG:apache.sh CSRV:TOMCAT1,TOMCAT2

Tabelle 14: Ressourcen: Service für andere Services

5.3.2.4.7 Clusterpaket muss auf diesem Clusternode aktiv sein (Ressource: PKG)

Definition	**PKG:<other_package>[:WAIT[:<m>]]**
Aktion beim Start	Nur wenn das Clusterpaket <other_package> auf diesem Clusternode gestartet ist, wird das aktuelle Clusterpaket gestartet. Mit [:WAIT], es wird auf das Paket gewartet, <m> ändert Wartezeit in Minuten, Default 5.
Aktion beim Stopp	keine
Beispiel	PKG:apache2:WAIT

Tabelle 15: Ressourcen: Clusterpaket muss lokal laufen

5.3.2.4.8 Clusterpaket darf auf diesem Clusternodes nicht aktiv sein (Ressource: !PKG)

Definition	**!PKG:<other_package>[:STOP|SWITCH]**
Aktion beim Start	Das Clusterpaket <other_package> darf auf diesem Clusternode nicht gestartet sein. Ist STOP angegeben, wird es gestoppt (und bleibt gestoppt). Ist SWITCH angegeben, wird Clusterpaket <other_package> gestoppt, beim Stoppen dieses Clusterpakets jedoch automatisch wieder gestartet.
Aktion beim Stopp	Ein geSWITCHtes Clusterpaket wird wieder gestartet.
Beispiel	!PKG:disturber:STOP

Tabelle 16: Ressourcen: Clusterpaket darf lokal nicht laufen

5.3.2.4.9 Clusterpaket muss mindestens n mal laufen (Ressource MINPKGS)

Definition	**MINPKGS:<pkgs>:<num>[:<m>]**
Aktion beim Start	keine
Aktion beim Stopp	Von den <pkgs> (regexp) müssen mindestens <num> laufen. MINPKGS sollte als letztes angegeben werden, damit vor dem Stoppen geprüft wird. Beim Clusterstop bzw. Reboot des Systems wird das Paket nach <m> (default:30) Minuten trotzdem heruntergefahren Hinweis: Will man das Stoppen eines Paketes mit MINPKGS erzwingen, so muss sccl stop mit --force aufgerufen werden.
Beispiel	MINPKGS:GALERA_HUGO_.*:1

Tabelle 17: Ressourcen: Clusterpaket muss mindestens n mal laufen

Beispiel siehe 8.8 „Galera Cluster (MariaDB)“

5.3.2.5 Filesystemressourcen

5.3.2.5.1 File System (Ressource: FS)

Definition	**FS:<mountpoint>[:FSCK]**
Aktion beim Start	Filesystem <mountpoint> mounten Filesystem kann nur an einem Clusternode gemountet sein. [:FSCK] führt vor dem Mounten einen Filesystemcheck aus
Aktion beim Stopp	Filesystem <mountpoint> entmounten.
Beispiel	FS:/opt/informix
Hinweise	Das verwendete Filesystem <mountpoint> muss in der Datei /etc/fstab mit der Option „noauto“ eingetragen sein. Die Zahlen für fs_passno sollten „0“ sein, damit kein file system check ausgeführt wird. Im Filesystem wird eine Datei fslock angelegt.

Tabelle 18: Ressourcen: File System

Hinweis: diese Ressource steht unter Windows nicht zur Verfügung

5.3.2.5.2 File System Group (Ressource: FSG)

Definition	**FSG:<fsgruppe>[:FSCK]**
Aktion beim Start	Filesystemgroup <fsgruppe> mounten Filesystemgroup kann nur an einem Clusternode gemountet sein. [:FSCK] führt einen Filesystemcheck aus
Aktion beim Stopp	Filesystem <fsgruppe> entmounten.
Beispiel	FS:/opt/informix
Hinweis	Bei FSG wird jedem Filesystem der Gruppe <Gruppenname> in der /etc/fstab folgendes vorangestellt: # SCCL GRP: <Gruppenname> #
Hinweis	Im Filesystem wird eine Datei fslockgp angelegt.

Tabelle 19: Ressourcen: File System Group

Hinweis: diese Ressource steht unter Windows nicht zur Verfügung

5.3.2.5.3 Network File System (Ressource: NFS)

Definition	**NFS:<nfs_file_system>**
Aktion beim Start	NFS-Filesystem <nfs_file_system> mounten. Kann an mehreren Clusternode gleichzeitig angedockt sein.
Aktion beim Stopp	NFS-Filesystem <nfs_file_system> entmounten.
Beispiel	NFS:/backup/oracle
Hinweis	Das zu verwendete Filesystem muss in der Datei /etc/fstab mit der Option „noauto“ eingetragen sein. Im Unterschied zu FS: kann ein NFS: an mehreren Clusternodes gleichzeitig gemountet werden.

Tabelle 20: Ressourcen: Network File System

Hinweis: diese Ressource steht unter Windows nicht zur Verfügung

5.3.2.5.4 Network File System (Ressource: NFSG)

Definition	**NFSG:<nfsgruppe>**
Aktion beim Start	NFS-Filesystem <nfsgruppe> mounten. Kann an mehreren Clusternode gleichzeitig angedockt sein.
Aktion beim Stopp	NFS-Filesystem <nfsgruppe>entmounten.
Beispiel	NFS:/backup/oracle
Hinweis	Bei NFSG wird jedem Filesystem der Gruppe <Gruppenname> in der /etc/fstab folgendes vorangestellt: # SCCL GRP: <Gruppenname> # Im Unterschied zu FSG: kann ein NFSG: an mehrere Clusternodes gleichzeitig gemountet werden.

Tabelle 21: Ressourcen: Network File System Group

Hinweis: diese Ressource steht unter Windows nicht zur Verfügung

5.3.2.5.5 Volume Gruppe aktivieren (nur HP-UX) (Ressource: VG)

Definition	**VG:<vg>**
Aktion beim Start	Volume Gruppe <vg> wird aktiviert.
Aktion beim Stopp	Volume Gruppe <vg> wird deaktiviert.
Beispiel	VG:vgdatabase
Hinweis	Nach dem VG muss noch ein FS erfolgen, um das Filesystem zu mounten

Tabelle 22: Ressourcen: Volume Gruppen (nur HP-UX)

5.3.2.6 Kontrollmechanismen

5.3.2.6.1 Prozess vorhanden (Ressource: PROC)

Definition	**PROC:<expression>[:WAIT]**
Aktion beim Start	Es wird geprüft, ob auf dem Clusternode ein bestimmter Prozess läuft. Dies wird mit "ps -efa | grep <expression>" bestimmt. Ist [WAIT] angegeben, wird auf den Prozess gewartet.
Aktion beim Stopp	keine
Beispiel	PROC:java:WAIT

Tabelle 23: Ressourcen: Prozess vorhanden

Hinweis: diese Ressource steht unter Windows nicht zur Verfügung

5.3.2.6.2 Prozess nicht vorhanden (Ressource: !PROC)

Definition	**!PROC:<expression>**
Aktion beim Start	Es wird geprüft, ob im Clustern ein bestimmter Prozess nicht läuft. Dies wird mit "ps -efa | grep <expression>" bestimmt.
Aktion beim Stopp	keine
Beispiel	!PROC:backup

Tabelle 24: Ressourcen: Prozess nicht vorhanden

Hinweis: diese Ressource steht unter Windows nicht zur Verfügung

5.3.2.6.3 Sperren setzen (Ressource: LOCK)

Definition	**LOCK:<Name>**
Aktion beim Start	Erzeugt einen Lockeintrag (siehe Kapitel 9, „Die Sperrmechanismen des SCript CLusters“
Aktion beim Stopp	Entfernt einen Lockeintrag (siehe Kapitel 9, „Die Sperrmechanismen des SCript CLusters“
Beispiel	LOCK:irgendwas
Hinweis	Jeder LOCK:<Name> kann nur einmal im Cluster aktiv sein. Versucht ein weiteres Paket den gleichen <Name>n noch einmal zu setzten, schlägt LOCK fehl und Paket wird wieder beendet.

Tabelle 25: Ressourcen: Lock

5.3.2.7 Unterressorcen

5.3.2.7.1 Unterressource verwenden (Ressource: RS)

Definition	**RS:<subressource>**
Aktion beim Start	Die Ressourcen des Clusterunterpakets werden aktiviert.
Aktion beim Stopp	Die Ressourcen des Clusterunterpakets werden deaktiviert.
Beispiel	RS:myfilesystems myfilesystems Kommentar_my_Filesystem FS:/myFileSystem

Tabelle 26: Ressourcen: Unterressourcen

5.3.2.8 Testskripte für die Clusterpaketüberwachung

5.3.2.8.1 Testskript zur automatischen Clusternodeumschaltung (Ressource: TST)

Definition	**TST:<sek>:[<path>]<file>**
Aktion beim Start	Es wird auf dem nächsten erlaubten Clusternode ein Testskript gestartet. Dieses Skript ruft alle <sek> Sekunden das Programm <path><file> auf, das die Verfügbarkeit der Clusterressource prüft. Wenn das Skript einen Fehlerwert liefert, dann wird vom Clusternode versucht, das Clusterpaket zu stoppen und anschließend wieder auf dem nächsten erlaubten Clusternode aus der $$PACKAGES zu starten. Wird <path> nicht angegeben, wir das Verzeichnis $$STARTSTOPDIR aus der sccl.conf verwendet.
Aktion beim Stopp	Das Testskript wird gestoppt.
Beispiel	TST:60:/usr/share/sccl/test_apache

Tabelle 27: Ressourcen: automatische Clusternodeumschaltung

Anmerkung: Wenn das Paket auf dem letzten Clusternode aus der $$PACKAGES gestartet wird, dann wird kein Testskript gestartet, um eine Endlosschleife der Start- und Stop Aktionen zu verhindern. Wenn eine Endlosschleife gewünscht ist, dann wird der erste Clusternode zusätzlich als letzter Clusternode eingetragen.

5.3.2.8.2 Testskript zum automatischen Paket Neustart (Ressource: STST)

Definition	**STST:<sek>:<num>:[<path>]<file>**
Aktion beim Start	Es wird auf dem aktuellen Clusternode ein Testskript gestartet. Dieses Skript ruft alle <sek> Sekunden das Programm <path><file> auf, das die Verfügbarkeit der Clusterressource prüft. Wenn das Skript einen Fehlerwert liefert, dann wird versucht, das Clusterpaket zu stoppen und anschließend wieder zu starten. <num> gibt hierbei die Anzahl der zulässigen Versuche für den Start an. Wird <path> nicht angegeben, wir das Verzeichnis $$STARTSTOPDIR aus der sccl.conf verwendet.
Aktion beim Stopp	Das Testskript wird gestoppt.
Beispiel	TST:60:5:/usr/share/sccl/test_apache

Tabelle 28: Ressourcen: automatischer Paket Neustart

5.4 Gruppen

Dienen nur der Übersichtlichkeit im sccl show_cluster.

In der Datei $$PACKAGES können Pakete zu Gruppen zusammengefasst werden.

Auf den Servern sind bei einem sccl show_cluster nur die Pakete zu sehen, in deren Gruppen sich der ausführende Server befindet.

6 Start des SCript CLusters

Für die Clusterkommunikation wird auf allen Clusternodes ein unix2webd-Prozess gestartet. Zu beachten ist ggf. eine Freischaltung des verwendeten Ports (siehe 5.1 „Grundkonfiguration (sccl.conf)“) in der jeweiligen Firewall der Clusternodes.

6.1 Linux

Das SCript CLuster ist so konfiguriert, dass es beim nächsten Reboot des Rechners automatisch gestartet wird. Die Konfiguration des SCript CLusters (siehe Kapitel 5, „Konfiguration des SCript Clusters“) muss vor dem Start des SCript CLusters erfolgen.

6.2 Windows

Durch die Programme create_cluser oder join_cluster werden die unter Windows benötigten Dienste eingerichtet. Die Konfiguration des SCript CLusters (siehe Kapitel 5, „Konfiguration des SCript Clusters“) muss vor dem Start des SCript CLusters erfolgen.

7 Kommandos des SCript CLusters

Hinweis: da $$SCCLBASEDIR in PATH aufgenommen ist, wird hier auf die Pfadangabe verzichtet, wo möglich.

7.1 Ausführungsberechtigung sccl auf Linux

Grundsätzlich ist auf Linux Systemen nur dem Benutzer root das Ausführungsrecht an sccl gewährt.

Sollen andere Benutzer sccl Kommandos ausführen, so müssen die Benutzer der Gruppe „sccl“ zugeordnet werden. Anschließend ist ein „sccl dist_config“ auszuführen, um die Berechtigung an den Zertifikaten anzupassen.

7.2 allgemeines Kommando sccl

Zur Herstellung der Kompatibilität zwischen Windows und Linux wurde einheitlich das Kommando „sccl“ eingeführt.

Zur Vereinfachung der Nutzung unterstützt das Kommando „sccl“ eine tab completion.

Somit kann z.B. die Eingabe von „sccl sho“ mit betätigen der <Tab> Taste vervollständigt werden zu „sccl show_cluster“.

Die tab completion Funktion ist auch für folgende Parameter wie z.B. Paketnamen oder Clusternodes verfügbar.

Zu den meisten sccl Komandos gibt es man-Pages, dann aber als z.B. sccl_show_cluster statt sccl show_cluster.

7.3 Clustermanagement

7.3.1 sccl create_cluster

```
# sccl create_cluster [--force] {-P | -p <CA-KEY-PWD>} [-c] <CLUSTERNAME>
```

Windows:

```
C:\Program Files\sccl> create_cluster [--force] {-P | -p <CA-KEY-PWD>} [-
   c] <CLUSTERNAME>
```

Das Kommando dient zum Erzeugen des Clusters inkl. der benötigten Zertifikate.

Der aktuelle Clusternode wird Clustermasternode.

Parameter:

-c <CLUSTERNAME>	Optional, aus Kompatibilitätsgründen
<CLUSTERNAME>	Name des Clusters
--force	force ggf. vorhandene Clusterkonfiguration überschreiben Anmerkung: im laufenden Betrieb darf der Clusternamen nicht geändert werden, weil sich daraus die Namen der Zertifikate ableiten
-p <CA-KEY-PWD>	Passwort <CA-KEY-PWD> für den CA Key Das Passwort bitte merken!
-P	Zufälliges Passwort für den CA Key erzeugen und im Filesystem speichern

Tabelle 29: sccl create_cluster, Parameter

7.3.2 sccl join_cluster

```
# sccl join_cluster [--force | --merge] [-m <Master IP>] [-p <Port>]
```

Windows:

```
C:\Program Files\sccl> join_cluster [--force | --merge] [-m <Master IP>]
   [-p <Port>]
```

Der Clusternode wird angewiesen, sich für den Beitritt zu einem Cluster vorzubereiten.

Hierzu wird ein Webserver auf dem Clusternode gestartet, Default Port: 3579

Parameter:

--force	force ggf. vorhandene Clusterkonfiguration überschreiben
--merge	Der Knoten wird dem Cluster hinzugefügt und die Konfiguration wird auch übernommen. Geht nur, wenn der neue Node ein Singlenodecluster ist ist die Paket- und Ressourcenamen noch nicht vorhanden sind.
-m <Master IP>	IP des Clustermasternodes festlegen nur dieser Clustermasternode darf den Clusternode hinzufügen.
-p <Port>	abweichender Port des Clusters, auf dem die Clusterkommunikation erfolgt, Default 3579

Tabelle 30: sccl join_cluster, Parameter

7.3.3 sccl add_node

```
# sccl add_node [--force] [--merge] [-l] [-p <CA-KEY-PWD> | -P
   <Passwortfile>] <neuer Clusternode>
```

Windows:

```
C:\Program Files\sccl> sccl add_node [--force] [--merge] [-l] [-p <CA-
   KEY-PWD> | -P <Passwortfile>] <neuer Clusternode>
```

Der Clustermasternode nimmt einen Clusternode in das Cluster auf.

Parameter:

--force	force sollte der Clusternode im Cluster schon vorhanden sein, wird dieser überschrieben
--merge	Der Node wird dem Cluster hinzugefügt und die Konfiguration wird auch übernommen Geht nur, wenn der neue Node ein Singlenodecluster ist ist die Paket- und Ressourcenamen noch nicht vorhanden sind Voraussetzung: sccl join_cluster muss auch mit „--merge" aufgerufen werden
-l	für Aufruf aus sccl create_cluster, wird nur intern verwendet und sollte nie, nie, wirklich nie verwendet werden
-p <CA-KEY-PWD>	Passwort <CA-KEY-PWD> des CA Keys Kann entfallen, wenn das Cluster mit sccl create_cluster -P erzeugt wurde.
-P <Passwordfile>	<Passwortfile> für den CA Key Kann entfallen, wenn das Cluster mit sccl create_cluster -P erzeugt wurde.
<neuer Clusternode>	Name des neuen Clusternodes, der Name muss vom Clustermasternode zu einer IP aufgelöst werden können

Tabelle 31: sccl add_node, Parameter

Nach der Aufnahme des neuen Clusternodes in das Cluster wird die Clusterkonfiguration übertragen.

7.3.4 sccl dist_config

```
# sccl dist_config [-m] [-n <CLUSTERNODE>] [-x <EXCLUDENODE>] [files…]
```

Windows:

```
C:\Program Files\sccl> sccl dist_config [-m] [-n <CLUSTERNODE>] [-x
  <EXCLUDENODE>] [files…]
```

Der Clustermasternode verteilt die Clusterkonfiguration auf die Clusternodes.

Parameter:

-m	Überträgt die Masterfunktion auf einen zweiten Clusternode kann nur mit -n <CLUSTERNODE> verwendet werden
-n <CLUSTERNODE>	Die Konfiguration wird nur an <CLUSTERNODE> geschickt
-x <EXCLUDENODE>	Die Konfiguration wird an alle Clusternodes außer <EXCLUDENODE> geschickt
[files…]	Files unter /etc/sccl

Tabelle 32: sccl dist_config, Parameter

7.3.5 sccl check_config

```
# sccl check_config [-v] [-q] [-C] [-i <node>]
```

Windows:

```
C:\Program Files\sccl> sccl check_config [-v] [-q] [-C] [-i <node>]
```

Prüft die Clusterkonfiguration sccl.conf, $$PACKAGES und $$RESOURCES im Cluster ab und gibt das Ergebnis aus.

Parameter:

-v	verbose
-q	quiet
-C	prüft nur die eigene Config
-i <node>	ein Knoten, der von der Prüfung ausgenommen wird

Tabelle 33: sccl check_config, Parameter

Sollte ein RST ohne SETSTATE angewendet werden, so warnt sccl check_config davor.

7.3.6 sccl remove_node

```
# sccl remove_node <CLUSTERNODE>
```

Windows:

```
C:\Program Files\sccl> sccl remove_node <CLUSTERNODE>
```

Knoten <CLUSTERNODE> aus der sccl.conf und u2w-Hosts-Liste entfernen.

7.3.7 sccl disable_node

```
# sccl disable_node [<CLUSTERNODE>]
```

Windows:

```
C:\Program Files\sccl> sccl disable_node [<CLUSTERNODE>]
```

verhindert, dass die Clusterpakete beim Booten des Clusternodes gestartet werden.

Setzt in den Sperrmechanismen einen Eintrag, der beim Booten des Clusternodes die Ausführung der Clusterpakete verhindert (siehe Kapitel 9, „Die Sperrmechanismen des SCript CLusters“).

Wird [<CLUSTERNODE>] nicht angegeben, ist es der lokale Server.

Es werden keine Clusterpakete gestoppt.

7.3.8 sccl enable_node

```
# sccl enable_node [<CLUSTERNODE>]
```

Windows:

```
C:\Program Files\sccl> sccl enable_node [<CLUSTERNODE>]
```

ermöglicht, dass die Clusterpakete beim Booten des Clusternodes gestartet werden.

Löscht in den Sperrmechanismen einen Eintrag, der beim Booten des Clusternodes die Ausführung der Clusterpakete verhindert (siehe Kapitel 9, „Die Sperrmechanismen des SCript CLusters“).

Wird [<CLUSTERNODE>] nicht angegeben, ist es der lokale Server.

Es werden keine Clusterpakete gestartet.

7.3.9 sccl reset_passwords

```
# sccl reset_passwords
```

Windows:

```
C:\Program Files\sccl> sccl reset_passwords
```

User und Admin Passwort der u2w Clusterkommunikation in der Datei $$U2WPWDDAT neu setzen. Sollte gemacht werden, wenn ein Knoten das Cluster verlassen hat (sccl remove_node führt sccl reset_passwords aus). Alle Clusternodes müssen erreichbar sein!

7.3.10 sccl update_ips

```
# sccl update_ips [-l]
```

Windows:

```
C:\Program Files\sccl> sccl update_ips [-l]
```

IP-Adressen der U2W-Kommunikation in der Datei $$U2WHOSTSDAT aktualisieren.

Parameter:

-l	verhindert den sccl dist_config

Muss immer dann ausgeführt werden, wenn ein Clusternode eine neue IP bekommen hat. Muss auf einem Clusternode ausgeführt werden, dessen IP sich nicht geändert hat. Alle Clusternodes müssen erreichbar sein! Es wird ein sccl dist_config ausgeführt!

Hinweis: haben sich alle IP-Adressen geändert, dann muss sccl update_ips -l auf allen Knoten ausgeführt werden.

7.4 Paketmanagement

7.4.1 sccl show_cluster

```
$ sccl show_clsuter [-a] [-e] [-F] [-G] [-l|-L] [-m] [-v] [-V]
   [<groups> ...]
```

Windows:

```
C:\Program Files\sccl> sccl show_cluster [-a] [-e] [-F] [-G] [-l|-L] [-m]
   [-v] [-V] [<groups> ...]
```

Gibt eine Übersicht aller Clusternodes und Clusterpakete im Cluster.

Die Übersicht wird in Form einer Tabelle ausgegeben, an den Schnittpunkten befinden sie dann die Statusinformationen.

Folgende Statusinformationen werden verwendet:

R	running
-	not running, start on startup
.	not running, allowed to start
+	not running, start on last node
;	not running, execute on startup
,	not running, allowed to execute
Rs	stopping
R!	error on stopping package
-s	starting
.s	starting
;e	executing
,e	executing
N	Node stopped
B	Node booting
D	Node disabled
?	no connection

Tabelle 34: sccl show_cluster, Statusinformationen

Parameter:

-a	zeigt alle Gruppen an
-e	Zeigt start/stop Fehler an
-F	zeigt FQDN statt Servernamen an
-G	Zeigt Gruppennamen mit an
-h	zeigt die Kurzhilfe an
-l|-L	-l zeigt lange Paketnamen zusätzlich an -L zeigt nur lange Paketnamen an
-m	zeigt den Clustermasternode mit an
-v	zeigt die Beschreibung der Statusinformationen mit an
-V	zeigt die SCCL2 Versionen der Clusternodes mit an
<groups> ...	Wenn Gruppen angegeben werden, dann werden nur die Pakete der Gruppen angezeigt

Tabelle 35: sccl start, Parameter

Beispiel:

```
$ sccl show_cluster
MYCLUSTER   clnode1  clnode2
----------------------------
package1      -          .
----------------------------
package2                 .
----------------------------
package3      R
----------------------------
```

7.4.2 sccl start

```
# sccl start [--simulate] [--force] [--norst] [-p <startparmeters>] [-i
  <initignore>] [-x <extrapars>] <CLUSTERPACKAGE> [<CLUSTERNODE>]
```

Windows:

```
C:\Program Files\sccl> sccl start [--simulate] [--force] [--norst] [-p
  <startparmeters>] [-i <initignore> ] [-x <extrapars> ]
  <CLUSTERPACKAGE> [<CLUSTERNODE>]
```

Startet ein Clusterpaket

Parameter:

--simulate	Es wird gezeigt, was alles gemacht würde (wird aber tatsächlich nicht ausgeführt)
--force	erzwingt den Start des Paketes
--norst	Alles hinter Ressource „RST“ wird nicht ausgeführt
-p	übergibt diesen Parameter an das Start Skript des PRG oder PRGP. Hierbei ist zu beachten, dass diese Parameter direkt nach dem „start“ vor den Parametern eines PRGP an das entsprechende Skript übergeben werden.
-i	Parameter für die Auflösung zyklischer Verkettungen, wird nur intern genutzt und sollte nie, nie, wirklich nie verwendet werden
-x	Parameter für die Auflösung zyklischer Verkettungen, wird nur intern genutzt und sollte nie, nie, wirklich nie verwendet werden
<CLUSTERPACKAGE>	Name des Paketes
<CLUSTERNODE>	Das Paket wird auf <CLUSTERNODE> gestartet. Ist der Parameter nicht angegeben, wird der lokale Clusternode verwendet.

Tabelle 36: sccl start, Parameter

Anmerkung: schlägt der Start einer Ressource fehl, werden bisher gestartete Ressourcen wieder gestoppt. Selbst erstellte Startskripte müssen hierzu einen Rückgabewert ungleich 0 zurückgeben.

7.4.3 sccl start_node

```
# sccl start_node -r [--cleanstart] [<CLUSTERNODE>]
```

Windows:

```
C:\Program Files\sccl> sccl start_node -r [--cleanstart] [<CLUSTERNODE>]
```

Startet alle Clusterpakete auf dem aktuellen oder dem angegebenen Clusternode <CLUSTERNODE>.

Parameter:

-r	wird nur für den Fall des Bootens des Servers verwendet und sollte nie, nie, wirklich nie verwendet werden.
[--cleanstart]	Clusterlocks bereinigen und Pakete starten, wenn "skip_start_node" nicht gesetzt (nur lokal) (zu skip_start_node siehe 9, „Die Sperrmechanismen des SCript CLusters“)
<CLUSTERNODE>	Die Clusterpakete des <CLUSTERNODES> werden gestartet. Ist der Parameter nicht angegeben, wird der lokale Clusternode verwendet.
Hinweis	Es werden nur die Clusterpakete gestartet, bei denen der Clusternode der Standard Clusternode ist.

Tabelle 37: sccl start_node, Parameter

7.4.4 sccl stop

```
# sccl stop [--simulate|--force|--clear|--tstkill] [--clusterstop]
  <CLUSTERPACKAGE> [<CLUSTERNODE>]
```

Windows:

```
C:\Program Files\sccl> sccl stop [--simulate|--force|--clear|--tstkill]
  [--clusterstop] <CLUSTERPACKAGE> [<CLUSTERNODE>]
```

Stoppt ein Clusterpaket

Parameter:

--simulate	Es wird gezeigt, was alles gemacht würde (wird aber tatsächlich nicht ausgeführt)
--force	Erzwingt das Beenden aller Ressourcen des Paketes
--clear	Erzwingt das Bereinigen der Clusterlocks benötigt <CLUSTERNODE>
--tstkill	Beendet das Testskript
--clusterstop	verhindert, dass Ressourcen, die mit SWITCH konfiguriert sind, neu gestartet werden
<CLUSTERPACKAGE>	Name des Paketes
<CLUSTERNODE>	Das Clusterpaket des <CLUSTERNODES> wird gestoppt. Ist der Parameter nicht angegeben, wird der lokale Clusternode verwendet.

Tabelle 38: sccl stop, Parameter

7.4.5 sccl stop_node

```
# sccl stop_node [--force|--clear] [--enable-traps] [-r] [-s]
  <CLUSTERNODE>]
```

Windows:

```
C:\Program Files\sccl> sccl stop_node [--force|--clear] [--enable-traps]
  [-r] [-s] <CLUSTERNODE>]
```

Stoppt auf einem Clusternode alle Clusterpakete.

Parameter:

--force	erzwingt das Stoppen aller Clusterpakete auf dem aktuellen oder dem angegebenen Clusternode <CLUSTERNODE>. Dies sollte nur in Notfällen verwendet werden, wenn bekannt ist, warum die Clusterpakete nicht normal herunterfahren.
--clear	hat die gleich Wirkung wie --force mit dem Unterschied, dass immer davon ausgegangen wird, dass das Start-Stopp-Skript der Anwendung erfolgreich die Anwendung beendet hat.
--enable-traps	wird nur intern genutzt und sollte nie, nie, wirklich nie verwendet werden
-r	wird nur für den Fall des Boots des Servers verwendet und sollte nie, nie, wirklich nie verwendet werden.
-s	Ist -s angegeben, dann werden die zu stoppenden Clusterpakete auf einem anderen erlaubten Clusternode geswitcht/gestartet, siehe Kapitel 5.2, „Konfiguration der Clusterpakete“.
<CLUSTERNODE>	Die Clusterpakete des <CLUSTERNODES> werden gestoppt. Ist der Parameter nicht angegeben, wird der lokale Clusternode verwendet.

Tabelle 39: sccl stop_node, Parameter

7.4.6 sccl reload

Ruft alle PRG und PRGP Ressourcen mit dem Parameter „reload“ auf

```
# sccl reload [-p <prgpars>] <CLUSTERPACKAGE> [<CLUSTERNODE>]
```

Windows:

```
C:\Program Files\sccl> sccl reload [-p <prgpars>] <CLUSTERPACKAGE>
  [<CLUSTERNODE>]
```

-p	übergibt diesen Parameter an das Start Skript des PRG oder PRGP. Hierbei ist zu beachten, dass diese Parameter direkt nach dem „reload“ vor den Parametern eines PRGP an das entsprechende Skript übergeben werden.
<CLUSTERPACKAGE>	Name des Paketes
<CLUSTERNODE>	Das Clusterpaket des <CLUSTERNODES> wird reloaded. Ist der Parameter nicht angegeben, wird der lokale Clusternode verwendet.

Tabelle 40: sccl reload, Parameter

7.4.7 sccl restart

Stoppt und startet ein oder mehrere Clusterpakete im Cluster.

```
# sccl restart [--simulate|--force|--clear] [--norst]
  <package1>[,<package2>,...] [<CLUSTERNODE>]
```

Windows:

```
C:\Program Files\sccl> sccl restart [--simulate|--force|--clear] [--
  norst] <package1>[,<package2>,...] [<CLUSTERNODE>]
```

Clusterpakete <packageX> wird auf dem aktuellen oder dem angegebenen Clusternode <CLUSTERNODE> gestoppt und wieder gestartet, jedoch nur wenn sie zu dem Zeitpunkt aktiv sind.

Parameter:

--simulate	Es wird gezeigt, was alles gemacht würde (wird aber tatsächlich nicht ausgeführt)
--force	Erzwingt den Neustart
--norst	Alles hinter Ressource „RST“ wird nicht ausgeführt
packageX	Clusterpaket, das restartet werden soll
CLUSTERNODE	Das Clusterpaket des <CLUSTERNODES> wird restartet. Ist der Parameter nicht angegeben, wird der lokale Clusternode verwendet.

Tabelle 41: sccl restart, Parameter

7.4.8 sccl status

Ruft alle PRG und PRGP Ressourcen mit Parameter „status“ auf

```
# sccl status [-p <prgpars>] <CLUSTERPACKAGE> [<CLUSTERNODE>]
```

Windows:

```
C:\Program Files\sccl> sccl status [-p <prgpars>] <CLUSTERPACKAGE>
  [<CLUSTERNODE>]
```

-p	übergibt diesen Parameter an das Start Skript des PRG oder PRGP. Hierbei ist zu beachten, dass diese Parameter direkt nach dem „status“ vor den Parametern eines PRGP an das entsprechende Skript übergeben werden.
<CLUSTERPACKAGE>	Name des Paketes
<CLUSTERNODE>	Das Clusterpaket des <CLUSTERNODES> wird geprüft. Ist der Parameter nicht angegeben, wird der lokale Clusternode verwendet.

Tabelle 42: sccl status, Parameter

7.4.9 sccl disable

```
# sccl disable <CLUSTERPAKET>
```

Windows:

```
C:\Program Files\sccl> sccl disable <CLUSTERPAKET>
```

Der Autostart des Clusterpaketes <CLUSTERPAKET> wird verhindert, indem vor den ersten Clusternode ein "-" eingetragen wird. Somit wird der <stdnode> der erste eingetragenen Clusternode. Anschließend wird ein sccl dist_config durchgeführt.

7.4.10 sccl enable

```
# sccl enable <CLUSTERPAKET>
```

Windows:

```
C:\Program Files\sccl> sccl enable <CLUSTERPAKET>
```

Der Autostart des Clusterpaketes <CLUSTERPAKET> erzeugt, indem der ersten Clusternode als <stdnode> eingetragen wird. Anschließend wird ein sccl dist_config durchgeführt.

7.5 Clusterkontrolle

7.5.1 sccl test_package

```
# sccl test_package <CLUSTERPAKET>
```

Windows

```
C:\Program Files\sccl> sccl test_package <CLUSTERPAKET>
```

Prüft, ob ein Clusterpaketes auf dem aktuellen Clusternode läuft.

Rückgabewert: 0=läuft, 1=läuft nicht (Fehlerfall;-)

<CLUSTERPACKAGE>	Clusterpaket, dessen Status geprüft werden soll

Tabelle 43: sccl status, Parameter

7.5.2 sccl get_aktnode

```
# sccl get_aktnode <CLUSTERPACKAGE>
```

Windows:

```
C:\Program Files\sccl> sccl get_aktnode <CLUSTERPACKAGE>
```

Es werden die Namen der Clusternodes ausgegeben, auf denen das <CLUSTERPACKAGE> gestartet ist. Ist das <CLUSTERPACKAGE> auf keinem Clusternode aktiv, dann wird

```
0
```

ausgegeben.

Beispiel:

```
# sccl get_aktnode package3
clnode1 clnode2
```

7.5.3 sccl list_cluster

Erstellt eine Liste der Clusternodes und deren aktive Clusterpakete.

```
# sccl list_cluster
```

Windows:

```
C:\Program Files\sccl> sccl list_cluster
```

gibt eine Tabelle mit zwei Spalten aus. In der ersten Spalte stehen die Clusternodes und in der zweiten die Clusterpakete, die auf den Clusternodes aktiv sind.

Beispiel:

```
# sccl list_cluster
clnode1 package1
```

```
clnode1 package3
clnode2 package3
```

7.5.4 sccl list_cluster_nodes

Liste der Namen aller Clusternodes.

```
# sccl list_cluster_nodes [<package>]
```

Windows:

```
C:\Program Files\sccl> sccl list_cluster_nodes [<package>]
```

Listet alle Clusternodes auf. Mit <package> werden die Clusternodes ausgegeben, auf denen <package> laufen darf.

Beispiel:

```
# sccl list_cluster_nodes
clnode1 clnode2
```

7.5.5 sccl list_packages_on_node

```
# sccl list_packages_on_node [-a|-s] [-u|-U] [-L] [<node>]
```

Windows

```
C:\Program Files\sccl> sccl list_packages_on_node [-a|-s] [-u|-U] [-L]
   [<node>]
```

Listet gestartete Clusterpaket auf, die auf dem Clusternode <node> läuft.

Parameter:

-a	Auch die nicht gestarteten Pakete werden ausgegeben.
-s	Zu den Paketen wird der Status gemäß sccl show_cluster ausgegeben
-u	unsortierte Ausgabe (= wie in der $$PACKAGES definiert)
-U	unsortierte Ausgabe (rückwärts) (= wie in der $$PACKAGES definiert, aber von unten nach oben)
-L	wird nur intern genutzt und sollte nie, nie, wirklich nie verwendet werden
<node>	optional Clusternode, dessen Clusterpakete ausgegeben werden sollen.

Tabelle 44: sccl list_packeges_on_node, Parameter

7.6 Monitoring mit ULS

7.6.1 Voraussetzungen

Auf einem Linux Server muss ein ULS Client installiert und konfiguriert sein, da dieser mit genutzt wird.

Auf einem Windows Server enthält SCCL einen abgespeckten ULS Client, der ULS Server muss nur mit dem Kommando

```
C:\Program Files\sccl> set_uls_server.cmd ULS-Server:ULS-Port
```

angegeben werden.

7.6.2 sccl 2_uls

```
# $$SCCLBASEDIR/bin/sccl 2_uls {enable | disable | status | config}
```

Monitoring, Steuerung und Ausführung

Parameter:

enable	setzt den Parameter ULS in der sccl.conf auf einen Wert ungleich ""
disable	setzt den Parameter ULS in der sccl.conf auf den Wert ""
status	sendet den Status des aktuellen Clusternodes an ULS
config	sendet die Konfiguration sccl.conf, $$PACKAGES und $$RESOURCES des aktuellen Clusternodes an ULS

Tabelle 45: sccl 2_uls, Parameter

8 Anwendungsbeispiele

In diesem Kapitel werden unterschiedliche Einsatzmöglichkeiten des SCript-CLusters anhand von Beispielen aufgezeigt.

Voraussetzungen für alle Beispiele:

- das SCript CLuster ist erstellt, alle Clusternodes sind Mitglied des Clusters
- die Auflösung von Servernamen zu IPs funktioniert (DNS oder Host Table)

8.1 Eine Datenbank auf zwei Servern (Cold Standby)

8.1.1 Aufgabenstellung

Zwei Clusternodes sollen eine Datenbank über eine virtuelle IP-Adresse zur Verfügung stellen. Die Datenbankdateien liegen auf einem NFS-Laufwerk, das zwischen den beiden Clusternodes bei Bedarf hin und her geschaltet werden kann.

8.1.2 Realisierung

- Name des Custers ist „DB_CLU“
- Servernamen lauten: „rechner1“ und „rechner2“

```
# vi sccl.conf
CLUSTER=DB_CLU
NODES="rechner1 rechner2"
```

1. Schritt: Paket konfigurieren

- der Paketname für die Oracle Datenbank soll sein: „datenbank“

```
# vi $$PACKAGES
datenbank - rechner1 rechner2
```

Das Paket wird nicht automatisch gestartet, es kann auf beiden Clusternodes gestartet werden.

2. Schritt: Ressourcen konfigurieren

- die Datenbank soll über die IP 10.1.111.20 angesprochen werden,
- der Mountpoint für die Datenfiles sei /oracle/admin
- das StartStoppSkript für die Datenbank sei /datenbank/admin/dbstartstop

Die Ressourcen zu dem Clusterpaket definieren

```
# vi $$RESOURCES
datenbank Oracle_Datenbank IP:AUTO:10.1.111.20:255.255.255.0
  NFS:/oracle/admin PRG:/datenbank/admin/dbstartstop
```

Beim Starten des Clusterpakets „datenbank“ wird

- die IP-Adresse 10.1.111.20 mit der Netzmaske auf einer automatisch bestimmten Schnittstelle gestartet,
- das NFS-Laufwerk /oracle/admin gemountet
- das StartStoppSkript /datenbank/admin/dbstartstop start ausgeführt.

Beim Stoppen erfolgt die Ausführung in umgekehrter Reihenfolge:

- das StartStoppSkript /datenbank/admin/dbstartstop stop wird ausgeführt.
- das NFS-Laufwerk /oracle/admin wird entmounted
- die IP-Adresse 10.1.111.20 wird gestoppt

Die /etc/fstab kontrollieren. Dort muss auf beiden Clusternodes ein Eintrag für /oracle/admin vorhanden sein, z.B.:

```
# vi /etc/fstab
nfs_server:/oradb/files /oracle/admin nfs
  noauto,intr,hard,bg,noexec,nosuid 0 0
```

Ein

```
# sccl show_cluster
```

liefert dann schon mal

```
DB_CLU     rechner1  rechner2
------------------------------
datenbank     .         .
------------------------------
```

Zum Starten und Stoppen der Datenbank benötigt man noch ein Skript.

```
$ vi /datenbank/admin/dbstartstop
```

```
#!/bin/bash

# This script is not suitable for a production environment!

case $1 in
    start)
        echo
        echo "Starting Database Server..."
        su - oracle  -c "echo 'startup;' | sqlplus / as sysdba"
        # you might need to start also the listener

        # Place a check of Oracle running correctly here
        # exit with 1, if Oracle does not run correctly

        exit 0
        ;;
    stop)
        echo
        echo "Stopping Database Server..."

        su - oracle -c "echo 'shutdown immediate;' | sqlplus / as sysdba"

        exit 0
```

```
        ;;
    *)
        echo $0 '{start|stop}'
        ;;
esac
```

Nun kann das Clusterpaket gestartet werden:

```
# sccl start datenbank rechner1
IP-Adresse 10.1.111.20 setzen
Das Filesystem /oracle/admin mounten

Starting Database Server...
...
Das Paket datenbank wurde auf rechner1 gestartet.
```

Prüfung des Clusterstatus

```
# sccl show_cluster
DB_CLU     rechner1  rechner2
-----------------------------
datenbank      R        .
-----------------------------
```

Hier eine schematische Darstellung des Clusters.

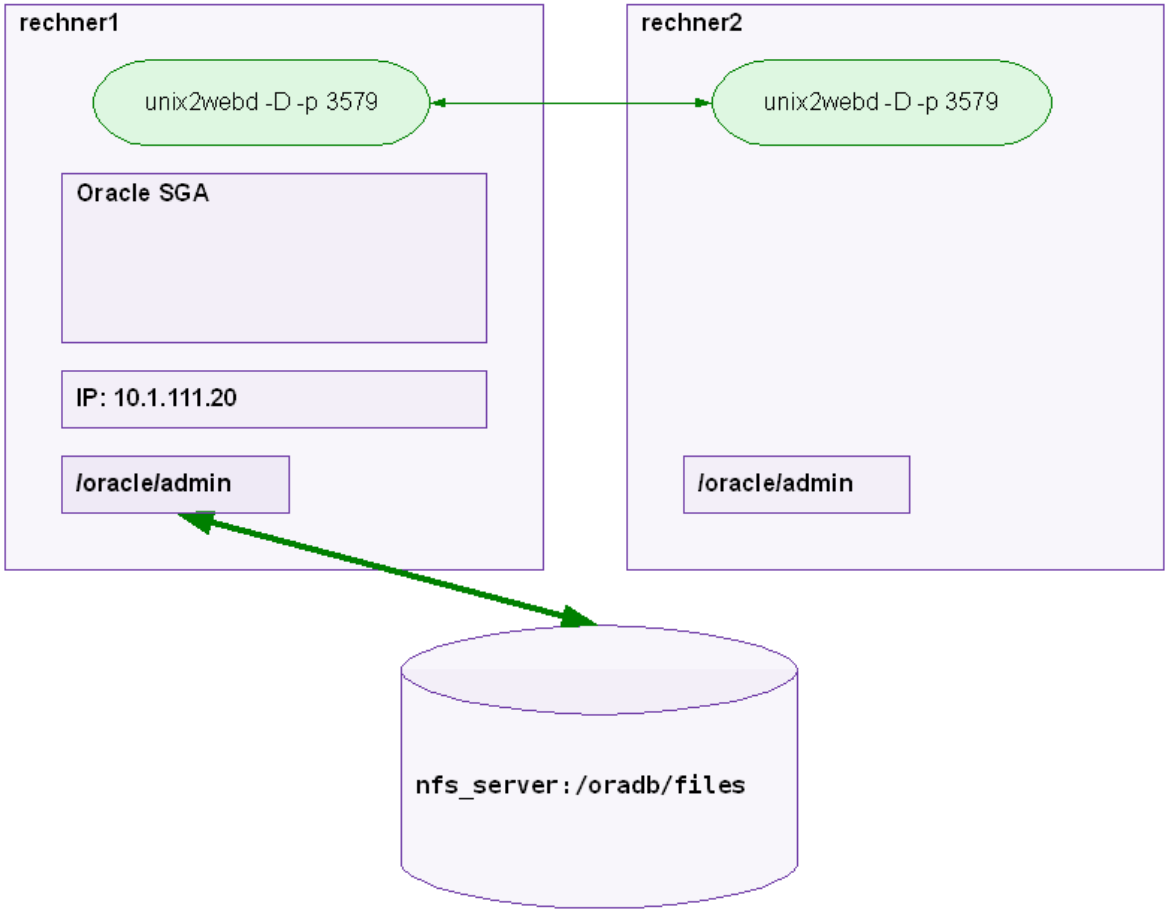

Abbildung 2: Datenbank-Cluster mit zwei Clusternodes, rechner1 aktiv

Mit

```
# sccl stop datenbank rechner1
```

und nachfolgendem

```
# sccl start datenbank rechner2
```

wird die Datenbank auf dem anderen Clusternode gestartet.

Hier eine schematische Darstellung des Clusters.

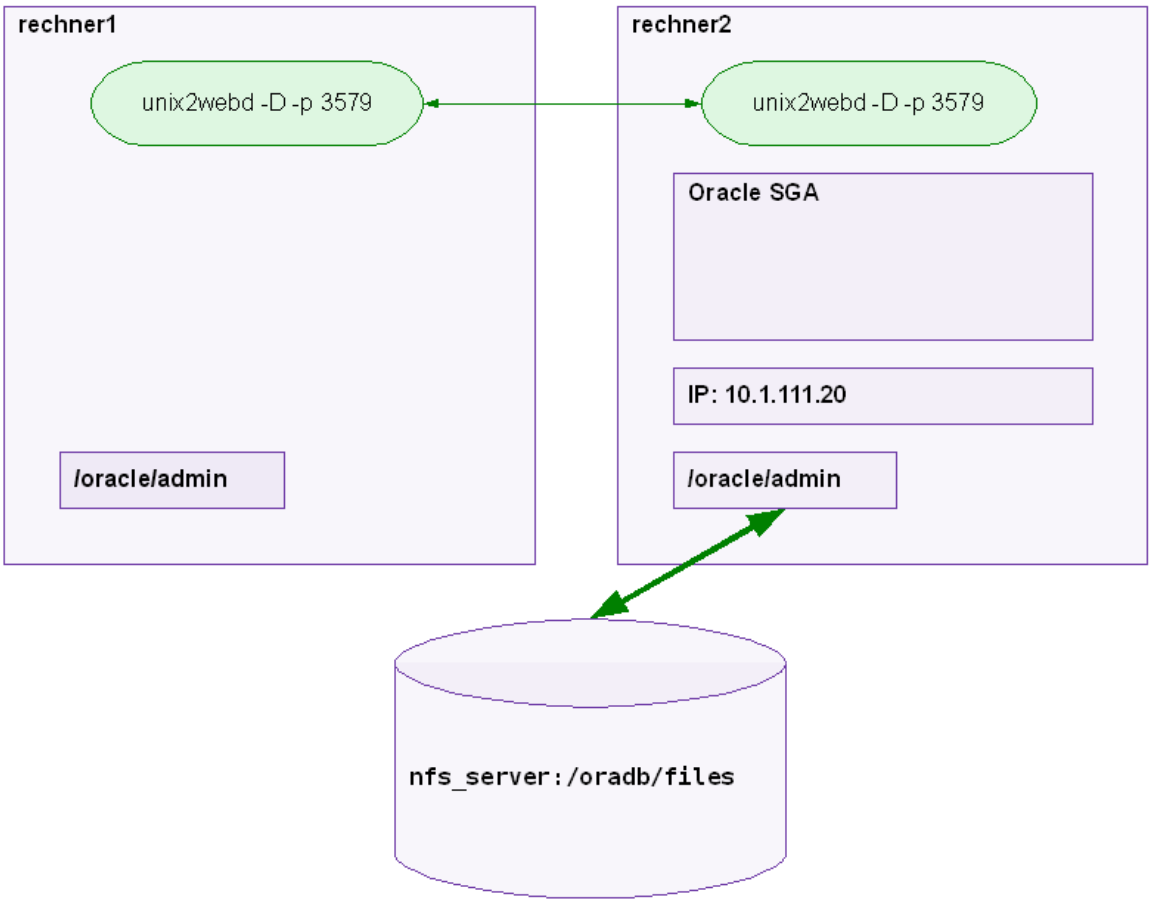

Abbildung 3: Datenbank-Cluster mit zwei Clusternodes, rechner2 aktiv

8.2 Datenbank und Applikation auf verschiedenen Servern

8.2.1 Aufgabenstellung

In einem Cluster sind zwei Clusternodes aufgenommen

- ein Clusternode mit Datenbankservice
- ein Clusternode mit Applikationsservice

Wird der Datenbankservice neu gestartet, soll der Applikationsservice auch neu gestartet werden.

8.2.2 Realisierung

- Name des Custers ist „DB_APP_CLU“
- der Rechnernamen für den Datenbankservice ist „db“
- der Rechnernamen für den Applikationsservice ist „app“

```
-- Auf dem Applikationsserver „app“:
# sccl join_cluster
-- Auf dem Datenbankserver „db“:
# sccl create_cluster -P DB_APP_CLU
# sccl add_node app
```

1. Schritt: Paket konfigurieren

- der Paketname für den Datenbankservice ist „dbservice“
- der Paketname für den Applikationsservice ist „appservice“

```
# vi $$PACKAGES
dbservice db
appservice app
```

Beide Pakete werden auf beiden Clusternodes beim Booten aktiviert.

2. Schritt: Ressourcen konfigurieren

- das StartStoppSkript für den Datenbankservice sei /datenbank/admin/dbstartstop
- das StartStopSkript für den Applikationsservice sei /app/admin/appstartstop

```
# vi $$RESOURCES
dbservice Oracle_Datenbank PRG:/datenbank/admin/dbstartstop SETSTATE
  RST:appservice:CLUSTER
appservice Apache_Tomcat CPKG:dbservice:WAIT PRG:/app/admin/appstartstop
```

Beim Starten des Clusterpakets „dbservice“ wird

- das StartStoppSkript /datenbank/admin/dbstartstop start ausgeführt.
- SETSTATE erzwingt, dass das Paket „dbservice“ seinen Status auf „Running“ setzt. Dieses ist wichtig für den nachfolgenden RST, um zu verhindern, dass Pakete gegenseitig aufeinander warten.
- anschließend wird das Paket „appservice“ im Cluster neu gestartet

Beim Stoppen des Clusterpakets „dbservice“ wird

- das Paket „appservice“ im Cluster gestoppt
- das StartStoppSkript /datenbank/admin/dbstartstop stop ausgeführt

Beim Starten des Clusterpakets „appservice“ wird

- geprüft, ob das Paket „dbservice“ im Cluster läuft. Wenn nicht, wird auf den Start des Paketes gewartet (WAIT)
- das StartStoppSkript start ausgeführt.

Beim Stoppen des Clusterpakets „appservice“ wird

- das StartStoppSkript /app/admin/appstartstop stop ausgeführt.

8.3 Datenbank und Applikation auf einem Server

Hinweis: Hierbei handelt es sich um eine Abwandlung des Beispiels 8.2, „Datenbank und Applikation auf verschiedenen Servern“.

Es können auch verschiedene Clusterpakete auf nur einem Server ausgeführt werden.

```
# sccl create_cluster -P SINGLENODE
```

```
# vi $$PACKAGES
dbservice server
appservice server
```

Die Datei $$RESOURCES aus 8.2, „Datenbank und Applikation auf verschiedenen Servern“ bleibt unverändert!

8.4 Anbindung Windows Share an Oracle Datenbankservice

8.4.1 Aufgabenstellung

In einem Cluster ist ein Clusternode vorhanden

- es läuft ein Oracle Datenbankservice unter Linux
- der Datenbankservice soll auf der IP 192.168.1.28/27 konfiguriert werden
- auf dem Windows Server „WServ“ gibt es eine Freigabe „PROD$“
- der Datenbankservice soll Dateien auf einem Windows Share //WServ/Prod$ ablegen.

8.4.2 Realisierung

- Name des Custers ist „DB_WIN_CLU“
- der Rechnernamen für den Datenbankservice ist „db“

```
# sccl create_cluster -P DB_WIN_CLU
```

1. Schritt: Paket konfigurieren

- der Paketname für den Datenbankservice ist „dbservice“

```
# vi $$PACKAGES
dbservice db
```

Das Paket wird beim Booten des Servers „db“ aktiviert

2. Schritt: Ressourcen konfigurieren

- das StartStoppSkript für den Datenbankservice sei /oracle/admin/dbstartstop
- der Mountpoint für den Windows Share sei /Transfer_Prod

```
# vi $$RESOURCES
dbservice Oracle_Datenbank IP:AUTO:192.168.1.28:27 NFS:/Transfer_Prod
  PRG:/oracle/admin/dbstartstop
```

Die /etc/fstab kontrollieren. Dort muss ein Eintrag für /Transfer_Prod vorhanden sein, z.B.:

```
# vi /etc/fstab
#Freigabe eines Windows Server (Dateiablage des Verfahrens) einbinden
//WServ/Prod$ /Transfer_Prod cifs
  noauto,uid=oracle,credentials=/etc/Prod.cred 0 0
```

Für die Einbindung des Windows Shares wird in der (nur für root auf rw gesetzten) Datei /etc/Prod.cred die Windows Kennung und das Passwort hinterlegt:

```
# vi /etc/Prod.cred
username=<Kennung>
password=<Passwort>
workgroup=<Domain>
```

Eintragung im Listener.ora anpassen (IP statt Servername)

```
LISTENER_db =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.1.28)(PORT = 3456))
    )
  )
```

8.5 Datenbankserver mit mehreren Applikationsservern

8.5.1 Aufgabenstellung

In einem Cluster sind mehrere Clusternodes aufgenommen

- ein Clusternode mit Datenbankservice
- mehrere Clusternodes mit Applikationsservices
- die Lastverteilung zwischen den Applikation Clusternodes erfolgt durch einen vorgeschalteten Loadbalancer.

Wird der Datenbankservice neu gestartet, soll der Applikationsservice auch neu gestartet werden.

8.5.2 Realisierung (per MULTI)

- Name des Custers ist „DB_APP_N_CLU“
- der Rechnernamen für den Datenbankservice ist „db“
- der Rechnernamen für die Applikationsservice ist „app1“, „app2“ und „app3“
- beim Booten der Server „app1 und „app2“ soll der Applikationsservice gestartet werden
- der Server „app3“ soll im Cold Standby für Hochlastphasen zur Verfügung stehen

```
-- Auf den Servern „app1“, „app2“ und „app3“
# sccl join_cluster
-- Auf dem Server „db“
# sccl create_cluster -P DB_APP_N_CLU
# sccl add_node app1
# sccl add_node app2
# sccl add_node app3
```

1. Schritt: Clusterpakete konfigurieren

- der Paketname für den Datenbankservice ist „dbservice“
- der Paketname für den Applikationsservice ist „appservice“

```
# vi $$PACKAGES
dbservice db
```

```
appservice app1,app2 app3
```

Das Paket dbservice wird beim booten des Servers db mit gestartet.

Das Paket appservice wird auf den Clusternodes app1 und app2 beim Booten aktiviert, auf „app3“ muss „appservice“ manuell gestartet werden.

2. Schritt: Ressourcen konfigurieren

- das StartStoppSkript für den Datenbankservice sei /datenbank/admin/dbstartstop
- das StartStoppSkript für den Applikationsservice sei /app/admin/appstartstop

```
# vi $$RESOURCES
dbservice Oracle_Datenbank PRG:/datenbank/admin/dbstartstop SETSTATE
  RST:appservice:CLUSTER
appservice Apache_Tomcat MULTI CPKG:dbservice:WAIT
  PRG:/app/admin/appstartstop
```

Beim Starten des Clusterpakets „dbservice“ wird

- das StartStoppSkript /datenbank/admin/dbstartstop start ausgeführt.
- SETSTATE erzwingt, dass das Paket „dbservice“ seinen Status auf „Running“ setzt. Dieses ist wichtig für den nachfolgenden RST, um zu verhindern, dass Pakete gegenseitig aufeinander warten.
- anschließend wird das Paket „appservice“ im Cluster neu gestartet, ggf. auf mehreren Clusternodes

Beim Stoppen des Clusterpakets „dbservice“ wird

- das Paket „appservice“ im Cluster auf allen Clusternodes gestoppt
- das StartStoppSkript /datenbank/admin/dbstartstop stop ausgeführt

Beim Starten des Clusterpakets „appservice“ wird

- durch die Ressource MULTI darf das Paket mehrfach gestartet werden
- geprüft, ob das Paket „dbservice“ im Cluster läuft. Wenn nicht, wird auf den Start des Paketes gewartet (WAIT)
- das StartStoppSkript /app/admin/appstartstop start auf dem angegebenen Clusternode oder lokal ausgeführt

Beim Stoppen des Clusterpakets „appservice“ wird

- das StartStoppSkript /app/admin/appstartstop stop auf dem angegebenen Clusternode ausgeführt. Wenn das Paket appservice nur einmal läuft, muss der Clusternode nicht angegeben werden.

Prüfung des Clusterstatus

```
# sccl show_cluster
DB_APP_N_CLU  db    app1    app2    app3
----------------------------------------
dbservice      R
```

```
----------------------------------------
appservice            R       R       .
----------------------------------------
```

Hinweis: wird „dbservice“ das erste Mal gestoppt und wieder gestartet, kann es sein, dass „appservice“ erst beim Start von „dbservice“ gestoppt und gestartet wird. Dann hatte „dbservice“ die Information in den Lockdateien (siehe Kapitel 9, „Die Sperrmechanismen des SCript CLusters“), dass es beim letzten Start den „appservice“ noch nicht mit „RST“ kannte.

8.5.3 Realisierung (per RS)

- Name des Custers ist „DB_APP_N_CLU“
- der Rechnernamen für den Datenbankservice ist „db“
- der Rechnernamen für die Applikationsservice ist „app1“, „app2“ und „app3“
- beim Booten der Server „app1 und „app2“ soll der Applikationsservice gestartet werden
- der Server „app3“ soll im Cold Standby für Hochlastphasen zur Verfügung stehen
- alle Clusternodes können die Rechnernamen zu IPs auflösen.

```
-- Auf den Server „app1“, „app2“ und „app3“
# sccl join_cluster
-- Auf dem Server „db“
# sccl create_cluster -P DB_APP_N_CLU
# sccl add_node app1
# sccl add_node app2
# sccl add_node app3
```

1. Schritt: Paket konfigurieren

- der Paketname für den Datenbankservice ist „dbservice“
- der Paketname für den Applikationsservice ist „appservice“

```
# vi $$PACKAGES
dbservice db

appservice1 app1
appservice2 app2
appservice3 - app3
```

Zwei von drei Paketen werden auf den Clusternodes beim Booten aktiviert.

2. Schritt: Ressourcen konfigurieren

- das StartStoppSkript für den Datenbankservice sei /datenbank/admin/dbstartstop
- das StartStoppSkript für den Applikationsservice sei /app/admin/appstartstop

```
# vi $$RESOURCES
dbservice Oracle_Datenbank PRG:/datenbank/admin/dbstartstop SETSTATE
  RST:appservice1,appservice2,appservice3:CLUSTER
appservice1 Apache_Tomcat RS:app
appservice2 Apache_Tomcat RS:app
appservice3 Apache_Tomcat RS:app
```

```
app Tomcat_für_alle MULTI CPKG:dbservice:WAIT PRG:/app/admin/appstartstop
```

Beim Starten des Clusterpakets „dbservice“ wird

- das StartStoppSkript /datenbank/admin/dbstartstop start ausgeführt.
- SETSTATE erzwingt, dass das Paket „dbservice“ seinen Status auf „Running“ setzt. Dieses ist wichtig für den nachfolgenden RST, um zu verhindern, dass Pakete gegenseitig aufeinander warten.
- anschließend wird das Paket „appservice1“, „appservice2“ und „appservice3“ im Cluster neu gestartet

Beim Stoppen des Clusterpakets „dbservice“ wird

- die Pakete „appservice1“, „appservice2“ und „appservice3“ gestoppt
- das StartStoppSkript /datenbank/admin/dbstartstop stop ausgeführt

Beim Starten des Clusterpakets „appserviceX“ wird

- die Ressource „app“ gestartet
- da diese MULTI gestartet werden darf, wird
    - geprüft, ob das Paket „dbservice“ im Cluster läuft. Wenn nicht, wird auf den Start des Paketes gewartet (WAIT)
    - das StartStoppSkript /app/admin/appstartstop start ausgeführt

Beim Stoppen des Clusterpakets „appservice1“, „appservice2“ oder „appservice3“ wird

- das StartStoppSkript /app/admin/appstartstop stop auf ausgeführt.

Prüfung des Clusterstatus

```
# sccl show_cluster
DB_APP_N_CLU  db    app1    app2    app3
----------------------------------------
dbservice      R
----------------------------------------
appservice1           R
----------------------------------------
appservice2                  R
----------------------------------------
appservice3                           -
----------------------------------------
```

Hinweis: wird „dbservice“ das erste Mal gestoppt und wieder gestartet, kann es sein, dass „appserviceX“ erst beim Start von „dbservice“ gestoppt und gestartet wird. Dann hatte „dbservice“ die Information in den Lockdateien (siehe Kapitel 9, „Die Sperrmechanismen des SCript CLusters“), dass es beim letzten Start den „appserviceX“ noch nicht mit „RST“ kannte.

8.6 Zwei Datenbankinstanzen alternativ nutzen

8.6.1 Aufgabenstellung

In einer Schulungsumgebung sind zwei Datenbankinstanzen vorhanden

- eine für den Grundlehrgang
- eine für den Fortgeschrittenenlehrgang

Die beiden Instanzen sollen über einen gemeinsamen Webserver angesprochen werden, daher dürfen die Instanzen nicht gleichzeitig laufen.

8.6.2 Realisierung

- Name des Custers ist SCHULUNG
- Datenbank- und Applikationsserver ist gemeinsam der Server „schul_server“

```
# sccl create_cluster -P SCHULUNG
```

1. Schritt: Paket konfigurieren

- die Instanz für den Grundlehrgang heißt „oracle_grund“
- die Instanz für den Fortgeschrittenen heißt „oracle_fort“
- der Paketname für den Applikationsservice ist webservice

```
# vi $$PACKAGES
oracle_grund - schul_server
oracle_fort - schul_server
webservice schul_server
```

2. Schritt: Ressourcen konfigurieren

- das StartStoppSkript für den Datenbankservice „oracle_grund“ sei /datenbank/admin/oracle_grund_startstop
- das StartStoppSkript für den Datenbankservice „oracle_fort“ sei /datenbank/admin/oracle_fort_startstop
- das StartStoppSkript für den Webservice sei /app/admin/webstartstop

```
# vi $$RESOURCES
oracle_grund Oracle_Grundkurs !CPKG:oracle_fort:STOP
  PRG:/datenbank/admin/oracle_grund_startstop SETSTATE
  RST:webservice:CLUSTER

oracle_fort Oracle_Aufbaukurs !CPKG:oracle_grund:STOP
  PRG:/datenbank/admin/oracle_fort_startstop SETSTATE
  RST:webservice:CLUSTER

webservice Apache_Tomcat PRG:/app/admin/webstartstop
```

Beim Starten des Clusterpakets „oracle_grund“ wird

- wenn es läuft, das Clusterpaket „oracle_fort“ gestoppt
- das StartStoppSkript /datenbank/admin/oracle_grund_startstop start ausgeführt.
- SETSTATE erzwingt, dass das Paket „oracle_grund“ seinen Status auf „Running“ setzt. Dieses ist wichtig für den nachfolgenden RST, um zu verhindern, dass Pakete gegenseitig aufeinander warten.
- anschließend wird das Paket „webservice“ im Cluster neu gestartet

Beim Starten des Clusterpakets „oracle_fort“ wird

- wenn es läuft, das Clusterpaket „oracle_grund“ gestoppt
- das StartStoppSkript /datenbank/admin/oracle_fort_startstop start ausgeführt.
- SETSTATE erzwingt, dass das Paket „oracle_fort“ seinen Status auf „Running“ setzt. Dieses ist wichtig für den nachfolgenden RST, um zu verhindern, dass Pakete gegenseitig aufeinander warten.
- anschließend wird das Paket „webservice“ im Cluster neu gestartet

8.7 Datenübernahme

8.7.1 Aufgabenstellung:

Es soll eine Datenübername in eine Datenbank per SCCL durchgeführt werden können.

8.7.2 Realisierung

Name des Custers ist KASSE

- Datenbankserver ist der Server „datenbank“
- Applikationsserver ist der Server „app“

```
-- Auf dem Applikationsserver „app“:
# sccl join_cluster
-- Auf dem Datenbankserver „datenbank“:
# sccl create_cluster -P KASSE
# sccl add_node app
```

1. Schritt: Paket konfigurieren

- Das Paket für die Datenbank heißt „dbservice“
- Das Paket für die Applikation heißt „Applikation“
- Das Paket für die Datenübername heißt „Import“

```
# vi $$PACKAGES
dbservice datenbank
Applikation app
Import - datenbank
```

Hinweis: das „-“ vor dem Servernamen des Paketes bedeutet, dass das Paket nicht standardmäßig beim Booten des Servers „datenbank“ mit gestartet wird. Es kann aber per SCCL gestartet werden. Siehe auch: 5.2 „Konfiguration der Clusterpakete (default: packages.conf)“

2. Schritt: Ressourcen konfigurieren

- das StartStoppSkript für den Datenbankservice sei /datenbank/admin/dbstartstop
- das StartStoppSkript für den Applikationsservice sei /app/admin/appstartstop
- das StartStoppSkript für den Import sei /datenbank/admin/import
- wird die Datenbank gestoppt, wird auch die Applikation gestoppt

```
# vi $$RESOURCES
dbservice Oracle_Datenbank PRG:/datenbank/admin/dbstartstop SETSTATE RST:
  Applikation:CLUSTER
Applikation Apache_Tomcat CPKG:dbservice:WAIT PRG:/app/admin/appstartstop
Import Datenimport CPKG:dbservice:WAIT PRG:/datenbank/admin/import
  TEARDOWN
```

Beim Starten des Clusterpakets „dbservice“ wird

- das StartStoppSkript /datenbank/admin/dbstartstop start ausgeführt.
- SETSTATE erzwingt, dass das Paket „dbservice“ seinen Status auf „Running“ setzt. Dieses ist wichtig für den nachfolgenden RST, um zu verhindern, dass Pakete gegenseitig aufeinander warten.
- anschließend wird das Paket „Applikation“ im Cluster neu gestartet, wenn es vorher durch das Paket dbservice gestoppt wurde.

Beim Stoppen des Clusterpakets „dbservice“ wird

- das Paket „appservice“ im Cluster auf allen Clusternodes gestoppt, falls es läuft.
- das StartStoppSkript /datenbank/admin/dbstartstop stop ausgeführt

Beim Starten des Clusterpakets „Applikation“ wird

- geprüft, ob das Paket „dbservice“ im Cluster läuft. Wenn nicht, wird auf den Start des Paketes gewartet (WAIT)
- das StartStoppSkript /app/admin/appstartstop start ausgeführt

Beim Stoppen des Clusterpakets „Applikation“ wird

- das StartStoppSkript /app/admin/appstartstop stop ausgeführt.

Beim Starten des Clusterpakets „Import“ wird

- geprüft, ob das Paket dbservice“ im Cluster läuft. Wenn nicht, wird auf den Start des Paketes gewartet (WAIT)
- das StartStoppSkript /datenbank/admin/import start ausgeführt.
- Anschließend wird das Paket wieder beendet (TEARDOWN). Das Paket kann also nie den Status „Running“ erreichen.

8.8 Galera Cluster (MariaDB)

8.8.1 Aufgabenstellung

Es soll ein Galera Cluster über SCCL gemanagt werden. Es ist sicherzustellen, dass immer mindestens ein Knoten läuft.

8.8.2 Realisierung

Name des Clusters ist GALERA

- die Datenbanken des Galera Clusters laufen auf den Server „dbg1“, „dbg2“ und „dbg3“

```
-- Auf den Servern „dbg2“ und „dbg3“:
# sccl join_cluster
-- Auf dem Server „dbg1“:
# sccl create_cluster -P GALERA
# sccl add_node dbg2
# sccl add_node dbg3
```

1. Schritt: Paket konfigurieren

- Die Paket für die Datenbanken des Galera Clusters heißen „mariadb_prod_x“ mit x =1..3 entsprechend der Nummer des Servers

```
# vi $$PACKAGES
mariadb_prod_1 dbg1
mariadb_prod_2 dbg2
mariadb_prod_3 dbg3
```

2. Schritt: Ressourcen konfigurieren

- das StartStoppSkript für den Datenbankservice sei /datenbank/admin/mariadbstartstop

```
# vi $$RESOURCES
mariadb_prod_1 Galera_DB1 PRG:/datenbank/admin/mariadbstartstop MINPKGS:
  mariadb_prod_[1-3]:1
mariadb_prod_2 Galera_DB2 PRG:/datenbank/admin/mariadbstartstop MINPKGS:
  mariadb_prod_[1-3]:1
mariadb_prod_3 Galera_DB3 PRG:/datenbank/admin/mariadbstartstop MINPKGS:
  mariadb_prod_[1-3]:1
```

Beim Starten der Clusterpakete „mariadb_prod_x“ wird

- das StartStoppSkript /datenbank/admin/mariadbstartstop start ausgeführt

Beim Stoppen der Clusterpakete „mariadb_prod_x“ wird

- geprüft, ob noch mindestens ein Paket des regular expressions „mariadb_prod_[1-3]“ läuft
- wenn ja, wird das StartStoppSkript /datenbank/admin/mariadbstartstop stop ausgeführt

9 Die Sperrmechanismen des SCript CLusters

Für die Ablage von Sperrinformationen der Pakete wird das Verzeichnis $$LOCKDIR aus der sccl.conf verwendet.

```
# grep LOCKDIR sccl.conf
LOCKDIR=/var/clusterlocks
```

Sollten sich im Betrieb des SCript CLusters, Probleme ergeben, so ist die Ursache üblicherweise hier zu suchen.

Es wird aber dringend von einer manuellen Bearbeitung der Dateien abgeraten!

Statt einer manuellen Bearbeitung sollte das betroffene Clusterpaket mit

```
# sccl stop --clear <CLUSTERPAKET> [<CLUSTERNODE>]
```

bereinigt werden.

Der Rest fehlt noch…

Lose Sammlung

Datei	Bedeutung
savestate.LASTSTATE	Status vor dem Booten, wird nach dem Booten ausgewertet
node.DISABLED	verhindert die Ausführung des SCript Cluster, wird mit sccl disable_node gesetzt und mit sccl enable_node wieder gelöscht.
Bla	fasel
PRG_Paketname_Pfad(/=_)_lock	Sperrdatei für das unter PRG ausgeführte Programm
PRGP_Paketname_Pfad(/=_)_lock	