3. Die Dateien im Einzelnen

3.1 `build.conf`

Liegt auf dem Builder und wird zusätzlich ins Paket kopiert, nach `/usr/share/scoreboard-system/build.conf`. Dadurch können die Maintainer-Skripte auf dem Zielrechner dieselben Werte lesen – Pfade und Listen stehen nur an dieser einen Stelle.

Die Datei ist von `set -a` … `set +a` umschlossen. Dadurch werden alle Zuweisungen exportiert, sodass nfpm als eigener Prozess sie in der `nfpm.yaml` als `${...}` einsetzen kann. Ohne das würde `source build.conf` nur Shell-Variablen setzen, die nfpm nie sieht.

Arrays exportiert `set -a` nicht – Bash kann das nicht. Das ist in Ordnung, denn Arrays braucht nur, wer die Datei sourct.

Aktiv genutzte Werte:

Variable	Wird gelesen von
`PACKAGE_NAME`	`nfpm.yaml`, `preparepackage.sh`
`PACKAGE_VERSION`	`nfpm.yaml`, `buildpackage.sh`
`PACKAGE_ARCH`	`nfpm.yaml`, `buildpackage.sh`
`JAVA_VERSION`, `JAVA_PACKAGE`	`postinstall.sh`, `nfpm.yaml`
`INSTALL_PATH`	alle Maintainer-Skripte, `preparepackage.sh`
`SERVICE_NAME`	`postinstall.sh`, `preremove.sh`, `preparepackage.sh`
`BUILD_ROOT`, `SOURCE_DIR`, `PACKAGE_ROOT`	`preparepackage.sh`, `buildpackage.sh`
die vier Verzeichnislisten	`preparepackage.sh`, `postinstall.sh`
`TEMPLATE_FILES`	`preparepackage.sh`, `postinstall.sh`

3.2 `preparepackage.sh`

Läuft auf dem Builder. Baut `package-root/` neu auf:

1. Prüft alle Pfade, bevor irgendetwas gelöscht wird
2. Löscht `package-root/` und legt es neu an
3. `REPLACE_DIRS` → `package-root/opt/scoreboard/`
4. `INITIAL_COPY_DIRS` → `package-root/usr/share/scoreboard-system/initial/`

5. `TEMPLATE_FILES` → `package-root/usr/share/scoreboard-system/templates/`
6. `systemd/scoreboard.service` → `.../systemd/`
7. Die vier Maintainer-Skripte → `.../scripts/`
8. `build.conf` → `.../`

Die Prüfungen in Schritt 1 sind der Grund, warum das Skript gefahrlos ein `rm -rf` enthalten darf:

- Alle Variablen müssen gesetzt sein
- `PACKAGE_ROOT` muss tatsächlich unterhalb von `BUILD_ROOT` liegen
- Weder `PACKAGE_ROOT` noch `BUILD_ROOT` dürfen im Live-Baum liegen
- `SOURCE_DIR` darf nicht `/opt/scoreboard` selbst sein
- Symlinks werden vorher aufgelöst, damit sie die Prüfung nicht aushebeln

Zusätzlich wird geprüft, ob bei jedem Maintainer-Skript der Shebang in Zeile 1 steht. Steht davor ein Kommentar, kann der Kernel die Datei nicht als Bash starten und dpkg fällt still auf `/bin/sh` zurück.

3.3 `nfpm.yaml`

Beschreibt das Paket. `name`, `version`, `arch` und `depends` kommen per `${...}` aus der Umgebung, also aus der `build.conf`.

Die Pfade unter `contents:` sind bewusst **relativ**. Stünde dort `${PACKAGE_ROOT}/opt` und die Umgebung fehlte, würde nfpm das zu `/opt` auflösen und das echte `/opt` der Maschine einpacken. Deshalb muss nfpm aus `BUILD_ROOT` heraus aufgerufen werden – `buildpackage.sh` erledigt das.

Zwei Einträge:

Aus `package-root`	Wird zu
`opt/`	`/opt` (also `/opt/scoreboard/{etc,lib,web}`)
`usr/share/scoreboard-system/`	`/usr/share/scoreboard-system`

3.4 `buildpackage.sh`

Klammer um den ganzen Build:

1. `build.conf` sourcen
2. `preparepackage.sh` aufrufen
3. Nach `BUILD_ROOT` wechseln und nfpm aufrufen
4. Prüfen, ob das erzeugte `.deb` wirklich die erwartete Version meldet, sonst abbrechen

5. `install-package.sh` mit nach `dist/` legen

Schritt 4 ist die Kontrolle gegen genau den Fehler, mit dem das alles angefangen hat: eine Versionsnummer in der `build.conf`, die nicht im Paket ankommt.

Wenn du ein eigenes Build-Skript verwenden willst, muss es mindestens die Schritte 1 bis 3 in dieser Reihenfolge tun.

3.5 `scripts/preinstall.sh` → `preinst`

Läuft auf dem Zielrechner **vor** dem Entpacken.

Einziger Zweck ist eine einmalige Migration. Früher gehörte `tokens` zum Paket. Jetzt nicht mehr – und beim Upgrade auf die erste Version ohne diesen Eintrag betrachtet dpkg das Verzeichnis als verwaiste Paketdatei und löscht es, bevor `postinstall.sh` überhaupt startet.

Das Skript legt deshalb vorher unter `/opt/scoreboard/.pre-upgrade/` eine Sicherung an. Sie arbeitet mit Hardlinks und kostet praktisch keinen Platz.
`postinstall.sh` holt sie zurück und räumt sie weg.

Sobald alle Zielrechner über diese Version hinaus sind, kann die Liste `MIGRATE_DIRS` oben im Skript geleert werden.

3.6 `scripts/postinstall.sh` → `postinst`

Läuft **nach** dem Entpacken und macht die eigentliche Einrichtung:

1. Prüft, ob `$1` überhaupt `configure` ist
2. Lädt `/usr/share/scoreboard-system/build.conf`
3. Prüft, ob Java vorhanden und alt genug ist
4. Legt die `CREATE_ONLY_DIRS` an
5. Legt die `INITIAL_COPY_DIRS` an – aber nur, wenn sie fehlen.
   Reihenfolge: vorhanden → nichts tun; Sicherung aus `preinst` vorhanden → zurückholen; sonst → Seed aus dem Paket
6. Kopiert fehlende `TEMPLATE_FILES`, vorhandene bleiben
7. Setzt die Ausführungsrechte auf die Start-/Stopp-Skripte
8. Installiert die systemd-Unit und startet den Dienst

Die `systemctl`-Aufrufe laufen nur, wenn systemd wirklich aktiv ist (`/run/systemd/system` existiert). In einem Container oder chroot würde sonst die ganze Installation daran scheitern.

3.7 `scripts/preremove.sh` → `prerm`

Läuft **vor** dem Entfernen von Dateien – und zwar auch bei jedem Upgrade. dpkg teilt den Grund über `$1` mit:

`$1`	Verhalten
`upgrade`	Nur den Dienst stoppen. Sonst nichts.
`remove`	Dienst stoppen und deaktivieren.
alles andere	Nichts tun.

Gelöscht wird in keinem Fall etwas. Die Paketdateien räumt dpkg selbst ab; alles andere sind Nutzdaten und bleiben stehen.

Das war die Stelle mit dem ursprünglichen Datenverlust: Das alte Skript wertete `$1` nicht aus und führte deshalb bei jedem Upgrade `rm -rf /opt/scoreboard` aus.

3.8 `scripts/postremove.sh` → `postrm`

Läuft **nach** dem Entfernen. Entfernt die systemd-Unit aus `/etc/systemd/system/` und lädt systemd neu – aber nur bei `remove` und `purge`. Bei `upgrade` muss die Unit liegen bleiben.

`/opt/scoreboard` wird auch bei `purge` nicht gelöscht, sondern nur ein Hinweis ausgegeben. Wer das anders möchte, findet die Stelle im Skript kommentiert.

Dieses Skript lädt die `build.conf` bewusst nicht: Zum Zeitpunkt von `postrm` hat dpkg die Paketdateien schon entfernt.

3.9 `install-package.sh`

Läuft auf dem **Zielrechner**, nicht auf dem Builder.

Der Grund für seine Existenz: Bei einem Upgrade führt dpkg das `prerm` des **bereits installierten** Pakets aus, nicht das aus dem neuen. Auf einer Maschine, auf der noch eine Version mit dem alten `preremove.sh` läuft, liefe also weiterhin `rm -rf /opt/scoreboard`, egal wie korrekt das neue Paket ist. Aus dem neuen Paket heraus lässt sich das nicht verhindern, weil zu diesem Zeitpunkt noch keine Zeile daraus gelaufen ist.

Ablauf:

1. Ist überhaupt etwas installiert? Wenn nein: direkt `dpkg -i`
2. Enthält das installierte `prerm` noch die gefährliche Zeile?

   Wenn nein: direkt `dpkg -i`
3. Wenn ja: Sicherung nach `/var/backups/` anlegen
4. Das `prerm` gegen die Fassung aus dem neuen `.deb` tauschen,
   die alte als `.vor-update` daneben legen
5. `dpkg -i`

Ab dem zweiten Update ist es ein reiner Durchläufer. Du kannst es deshalb dauerhaft verwenden und musst dir nicht merken, welche Maschine schon umgestellt ist.

Revision #2
Created 2026-08-20 10:53:57 UTC by dokumentation@bgrw-krefeld.de
Updated 2026-08-20 11:05:40 UTC by dokumentation@bgrw-krefeld.de