Squeeze-Release Arbeiten

Diese Seite beschreibt die **Git-Seite eines neuen SQUEEZE-Releases**: vom Abzweigen des Release-Branches bis zu dem Punkt, an dem der Release-Candidate auf Staging deployed werden kann und der `develop` wieder auf die Folgeversion zeigt.

Die Schritte sind bewusst als Checkliste formuliert – am besten von oben nach unten abarbeiten und nichts überspringen, da spätere Schritte auf früheren aufbauen.

Beteiligt sind zwei Repositories:

Repository	Inhalt	Branch	Tag
SQUEEZE	Backend	`release/2.XX`	`2.XX.0-rc.N`
DEXPRO Platform	Frontend	`release/squeeze/2.XX`	`squeeze-2.XX.0-rc.N`

In den Beispielen unten wird die Version **2.34** released, der `develop` läuft danach auf **2.35** weiter.

1. Release-Branch abzweigen

☐ **Abzweig-Commit auf `develop` bestimmen.** Der Commit muss so gewählt sein, dass genau die Features enthalten sind, die im Azure Sprint-Board zur Release-Nummer gehören – und **keine** Features, die erst für die nächste Version vorgesehen sind.

- Schaue auf Azure Commits Seite
- Schaue nach den Merge-Daten ungefähr auf Start des Sprints
- Schaue dir das Sprint-Board des neuen Sprints an
- Guckst du was da drauf ist und nicht "super-extra-wichtig-will-ich-dringend-noch-haben-sonst-geht-Welt-unter"
- Ermittle Schnittmenge daraus und finde den letzten Commit der nichts aus dem neuen Sprint behandelt

☐ Im **SQUEEZE**-Repo den Branch `release/2.34` von diesem Commit anlegen.

☐ Im **DEXPRO Platform**-Repo den passenden Branch `release/squeeze/2.34` anlegen.

> Der Abzweigpunkt ist die fehleranfälligste Stelle im gesamten Prozess. Ein zu spät abgezweigter Branch nimmt Features der Folgeversion mit ins Release. Im Zweifel vor dem Anlegen des Branches mit dem PO bzw. dem Team abgleichen, welche Work Items zum Release gehören.

2. Vorherigen Release-Branch mergen

Als Absicherung, falls ein Backmerge des letzten Releases in den `develop` vergessen wurde:

- ☐ Im SQUEEZE-Repo `release/2.33` in `release/2.34` mergen.
- ☐ Im DEXPRO Platform-Repo `release/squeeze/2.33` in `release/squeeze/2.34` mergen.

Bringt der Merge keine Änderungen, ist alles in Ordnung – der Schritt kostet nichts und verhindert, dass ein Hotfix aus dem letzten Release im neuen Release wieder fehlt.

3. Release-Candidate taggen

Getaggt wird nach Semver, beginnend mit `rc.1`; jeder weitere Kandidat zählt hoch (`rc.2`, `rc.3`, …).

- ☐ SQUEEZE-Repo (Backend): Tag `2.34.0-rc.1` auf dem Release-Branch setzen.
- ☐ DEXPRO Platform-Repo (Frontend): Tag `squeeze-2.34.0-rc.1` auf dem Release-Branch setzen.

> Das Frontend-Tag trägt zwingend das Präfix `squeeze-`, weil im DEXPRO Platform-Repo mehrere Produkte liegen.

4. Images bauen

Die Pipelines starten für Tags nicht automatisch, sie müssen in Azure DevOps **manuell auf dem jeweiligen Tag** gestartet werden:

- ☐ build and test - squeeze & worker (e2e) für das Tag im SQUEEZE-Repo → erzeugt die Docker-Images für Squeeze und Worker.
- ☐ push image - dexp-frontend für das Tag im DEXPRO Platform-Repo → erzeugt das Frontend-Image.
- ☐ Beide Pipelines sind grün durchgelaufen.

Damit ist der Release-Candidate fertig und kann auf Staging deployed werden.

5. `develop` auf die nächste Version heben

Direkt nach dem Abzweigen muss der `develop` im **SQUEEZE**-Repo auf die Folgeversion gestellt werden – sonst laufen die nächsten Entwicklungs-Builds weiter unter der alten Versionsnummer. Es gibt dafür im aktuellen Sprint eine Story für die Nacharbeiten, die in neues Sprints dupliziert wird, z.B. für 2.34 Release Nacharbeiten

- [ ] `config/versions.json` auf die neue Version setzen:

```
{
  "initialVersion": "2.35-develop"
}
```

- [ ] In `devops/swagger/ts-client.config.json` die Version erhöhen **und die Build-Nummer auf 1 zurücksetzen**:

```
{
  "npmName": "@dex/squeeze-client-ts",
  "npmVersion": "2.35.0-build.1"
}
```

- [ ] Beide Änderungen in **einem** PR nach `develop` bringen und mergen.

> Die Build-Nummer nicht weiterzählen lassen, sondern auf `build.1` zurücksetzen. Sie zählt innerhalb einer Version die veröffentlichten TS-Client-Pakete.

6. Neue TS-Client-Version verteilen

Nach dem Merge des PRs aus Schritt 5 wird automatisch eine neue Version des TypeScript-Clients `@dex/squeeze-client-ts` gebaut und veröffentlicht. Diese Version muss anschließend in den abhängigen Projekten eingetragen werden – in dieser Reihenfolge:

- [ ] **API-Tests** im SQUEEZE-Repo (`test/apitests/package.json`)
- [ ] SQUEEZE Viewer
- [ ] SQUEEZE Configframework
- [ ] **Frontend** im DEXPRO Platform-Repo

Die Reihenfolge ist wichtig: Viewer und Configframework werden vom Frontend eingebunden, das Frontend kommt deshalb zuletzt.

Weiterführend

Diese Seite deckt nur die Git- und Build-Schritte ab. Die weiteren Release-Tätigkeiten – finale Versionsnummer im Release-Branch, On-Premise-Artefakte, Testszenarien für den RC sowie Deployment und Ansible-Rollout – sind im DEXPRO-Platform-Wiki dokumentiert:

- Squeeze Releases
- Release und Deployment
- Versionierung

Zum grundsätzlichen Branch-Modell siehe Allgemein.

Revision #4
Created 2026-08-18 08:33:01 UTC by Florian Vick
Updated 2026-08-18 09:49:18 UTC by Florian Vick