# Update und Upgrade auf Webdesk 4.x

Diese Seite beschreibt, wie ein Update bzw. Upgrade von **Webdesk 4.x** abläuft und wie es sicher durchgeführt wird. Sie erklärt, was Webdesk beim Start einer neuen Programmversion automatisch erledigt (Datenbankschema, Modul-Upgrade-Skripte, Registrierung) und welche Einstellungen dabei eine Rolle spielen. Zielgruppe sind Systemadministratoren und Consultants.

Die Seite gilt für Webdesk 4.x, das als WAR-Datei in einem Tomcat oder als Container-Image betrieben wird. Das Update mit dem Windows-Installer (setup-...exe) ist auf der Seite Webdesk Update[1] beschrieben; diese Seite ergänzt sie um die Hintergründe und die Abläufe für WAR- und Container-Betrieb. Die Erstinstallation ist unter Webdesk 4[2] beschrieben.

## Unterstützte Java-Versionen je Webdesk-Version

Die Tabelle zeigt, ab welcher Webdesk-Version sich die mindestens erforderliche Java-Version geändert hat und welche weitere Java-Version bereits unterstützt wurde. Die Grenzen sind am Compiler-Level des Builds der jeweiligen Version geprüft (source/target bzw. release in der Maven-Konfiguration) und, soweit vorhanden, in den Release Notes belegt. Die Versionsnummern sind nicht streng chronologisch: Hotfix-Versionen älterer Linien erscheinen auch nach neueren Hauptversionen und behalten die Java-Anforderung ihrer Linie.

| Webdesk-Version | Mindestens Java | Zusätzlich unterstützt |
|---|---|---|
| 4.0.0 bis 4.1.1 | Java 7 (aus dem Build abgeleitet, in den Release Notes nicht ausdrücklich genannt) | keine |
| 4.1.2 bis 4.19.x | Java 8 | keine (Java 11 wird noch nicht unterstützt) |
| 4.20.0 bis 4.57.x | Java 8 (ab 4.56.0 mindestens Update 163, siehe Neue Features in 4.56[3]) | Java 11 ab 4.20.0 (Release Notes 4.20.0[4]: Java 11 benötigt mindestens Tomcat 9) |
| 4.58.0 bis 4.61.x | Java 11 (Release Notes 4.58.0[5]: "Webdesk basiert nun auf Java 11") | keine |
| 4.62.0 bis 4.63.x | Java 11 | Java 17 ab 4.62.0 (Release Notes 4.62.0[6]: Nashorn wird eingebettet, um den Betrieb mit Java 17 zu ermöglichen); zusätzliche JVM-Parameter nötig, siehe Webdesk mit Java17[7] |
| 4.64.0 und höher | Java 17 (Breaking Changes 4.64[8]) | keine; Java 11 reicht nicht aus |

Java 21 oder neuer wird in den Release Notes und im Build nicht genannt; für 4.64 und höher ist Java 17 zu verwenden. Beim Betrieb als Container-Image sind Java und Tomcat bereits im Image enthalten.

Wenn bei einem Update eine Java-Grenze überschritten wird:

- **Update auf 4.58 oder höher:** Java 11 und Tomcat 9 auf dem Webdesk-Server bereitstellen (Java 11 benötigt mindestens Tomcat 9).
- **Update auf 4.64 oder höher:** Java 17 auf dem Webdesk-Server und beim IF6020-TA-Connector installieren. Java 11 reicht nicht aus. Prüfen, welche Java-Version der Windows-Dienst bzw. das Startskript tatsächlich verwendet.

- **Unter Java 17:** die Startparameter --add-opens setzen (setenv.sh bzw. wrapper.conf) und den Parameter -Dnashorn.args=--no-deprecation-warning entfernen (Webdesk mit Java17[9], Neue Features 4.64[10]).
- **Update auf 4.56 oder höher:** mindestens Java 8 Update 163 verwenden und vorher eine Vollsicherung der Datenbanken erstellen (Neue Features in 4.56[11]).
- Kundenspezifische Java-Jobs und Integrationen nach dem Wechsel testen.

## Voraussetzungen

- **Java 17:** Webdesk 4.64 und der IF6020-TA-Connector laufen mit Java 17, Java 11 ist für diese Version nicht ausreichend (siehe Breaking Changes 4.64[12] und die Tabelle oben). Das mitgelieferte Container-Image basiert auf einem JDK 17. Bei mehreren Java-Installationen ist zu prüfen, welche Java-Version der Dienst bzw. das Startskript tatsächlich verwendet.
- **Tomcat 9:** Das Container-Image (webdesk-webapp/Dockerfile) basiert auf einem Tomcat-9-Image mit JDK 17. Auch der Windows-Installer liefert einen Tomcat 9 mit.
- **JVM-Parameter --add-opens:** Je nach Betriebsart sind beim Start der JVM folgende Parameter erforderlich (Liste laut Neue Features 4.64[13]):

  ```
  --add-opens java.base/jdk.internal.loader=ALL-UNNAMED
  --add-opens java.base/jdk.internal.module=ALL-UNNAMED
  --add-opens java.base/jdk.internal.reflect=ALL-UNNAMED
  --add-opens java.base/java.lang.module=ALL-UNNAMED
  --add-opens jdk.crypto.ec/sun.security.ec.point=ALL-UNNAMED
  --add-opens jdk.dynalink/jdk.dynalink.linker.support=ALL-UNNAMED
  --add-opens java.base/sun.util.calendar=ALL-UNNAMED
  --add-opens java.base/java.util.concurrent.locks=ALL-UNNAMED
  --add-opens java.base/java.lang=ALL-UNNAMED
  --add-opens java.base/java.lang.invoke=ALL-UNNAMED
  --add-opens java.base/java.lang.reflect=ALL-UNNAMED
  --add-opens java.base/java.io=ALL-UNNAMED
  --add-opens java.base/java.util=ALL-UNNAMED
  --add-opens java.rmi/sun.rmi.transport=ALL-UNNAMED
  --add-opens java.base/java.net=ALL-UNNAMED
  ```

  Die Parameter werden dort gesetzt, wo die JVM gestartet wird, z. B. über die Umgebungsvariable JDK_JAVA_OPTIONS (so setzt das mitgelieferte catalina.bat des Windows-Installers einige davon) oder über die Tomcat-Startoptionen. Im Container müssen sie an die JVM des Containers übergeben werden; das Dockerfile der Webanwendung setzt sie nicht selbst.
- **Ausgangsversion:** Webdesk-Versionen älter als 3.8 benötigen laut Webdesk Update[14] zuerst ein Update auf die letzte 3.8.x.
- **Hinweise zur Zielversion:** Vor jedem Update sind die Seiten "Neue Features" und "Breaking Changes" der Zielversion zu lesen, z. B. Breaking Changes 4.64[15]. Beim Update auf Webdesk 4.56 oder höher werden gespeicherte Zugangsdaten neu verschlüsselt, siehe Neue Features in 4.56[16].

## Was beim Start einer neuen Version passiert

Ein Update besteht im Kern aus dem Austausch der Programmdateien und einem Neustart. Alle weiteren Schritte führt Webdesk beim Start selbst aus, in dieser Reihenfolge:

1. **Hibernate-Schemaupdate:** Beim Aufbau der Hibernate-Session-Factory werden fehlende Tabellen und Spalten der Webdesk-Datenbank automatisch ergänzt (Eigenschaft webdesk-sessionFactory.schemaUpdate, Standard: aktiv). Umbenennungen, Löschungen und Datenmigrationen leistet das Schemaupdate nicht; dafür gibt es die Modul-Upgrade-Skripte.
2. **Index-Manager:** Vergleicht die vorhandenen Datenbankindizes mit den erwarteten und legt fehlende an (PoStartup.executeIndexManager, Standard: aktiv; Fehler hierbei führen nur zu einer Warnung im Log).

3. **Modul-Upgrade-Skripte:** Für jedes Modul werden die Upgrade-Schritte der neuen Version ausgeführt (siehe unten). Dieser Schritt läuft bei jedem Start und ist nicht abschaltbar.
4. **Prüfungen:** Konsistenzprüfungen der Datenbank (bei MySQL u. a. der Zeichensatz) und eine Zeitleisten-Prüfung der Personen (abschaltbar mit webdesk.disableTimeLineChecks).
5. **Lizenz** und **Systemparameter:** Die Lizenzdefinitionen werden geladen. Neue Systemparameter (Beanproperties) werden in die Datenbank geschrieben, bestehende Werte bleiben erhalten, und alle Werte werden angewendet.
6. **Registrierung:** Aktionen, Konfigurationen, Textbausteine, Jobs usw. der Module werden in die Datenbank übernommen. Ob und wie das geschieht, steuert webdesk.runRegistration (siehe unten).
7. **Volltextindex:** Der Suchindex wird aktualisiert (webdesk.luceneIndexing, Standard: aktiv).

## Relevante Einstellungen in der webdesk.properties

| Eigenschaft | Erklärung | mögliche Wert(e) |
|---|---|---|
| webdesk.runRegistration | Steuert, ob und wie beim Start die Registrierung läuft (Werte siehe unten). Der Wert 1 wird nach der Registrierung automatisch auf 0 zurückgesetzt. Kann auch als Umgebungsvariable WD_WEBDESK_RUNREGISTRATION gesetzt werden. | 0, 1, 2, 3, 4 (Standard: 1, wenn nicht gesetzt oder ungültig; die mitgelieferte Vorlage setzt 1) |
| webdesk.runRegistrationInThread | Bei true läuft die Registrierung in einem Hintergrund-Thread ("Webdesk Artifact Registration"), während Webdesk bereits startet. Bei false wird sie im Vordergrund ausgeführt, der Start ist dann erst danach beendet. | true, false (Standard: true) |
| webdesk-sessionFactory.schemaUpdate | Bei true ergänzt Hibernate beim Start fehlende Tabellen und Spalten. Bei false wird das Schema nicht angepasst; Webdesk setzt dann voraus, dass die Datenbank bereits zum Programmstand passt. Für diese Einstellung gibt es keine WD_-Umgebungsvariable. | true, false (Standard: true) |
| PoStartup.executeIndexManager | Bei true werden beim Start die Datenbankindizes geprüft und fehlende angelegt. Das kann bei großen Datenbanken lange dauern. Die Vorlage webdesk.properties.template verwendet den älteren Namen webdesk-SessionFactory.executeIndexManager und setzt ihn auf false. | true, false (Standard: true) |

## Modul-Upgrade-Skripte

### Wie die Version entscheidet, was ausgeführt wird

Jedes Modul (po, wf, ta, gw, ...) hat eine eigene Versionsnummer, eine fortlaufende ganze Zahl, die unabhängig von der Webdesk-Releasenummer ist. Der Stand pro Modul ist in der Datenbank gespeichert und wird in der Aktion "Module" in der Spalte "Version" angezeigt. Das Programm bringt für jedes Modul die Zielversion mit. Beim Start gilt:

- Stimmen gespeicherte Version und Zielversion überein, passiert nichts (Log: Module ... is up-to-date).
- Ist die gespeicherte Version kleiner, werden für jede Version dazwischen die Dateien im Ordner .../update/vNN/ des Moduls ausgeführt (NN ist die jeweilige Versionsnummer). Zuerst laufen die prepare.xml aller anstehenden Versionen, danach die upgrade.xml Version für Version. Nach jeder erfolgreich abgeschlossenen upgrade.xml wird die neue Versionsnummer in der Datenbank gespeichert.
- Ein neu hinzukommendes Modul (z. B. nach Lizenzerweiterung) wird direkt mit der Zielversion angelegt, es werden keine Upgrade-Schritte ausgeführt.
- Module, die in der neuen Version nicht mehr vorhanden bzw. lizenziert sind, werden auf "deaktiviert" (detached) gesetzt; sie werden wieder aktiviert, sobald sie erneut vorhanden sind.
- Die Module werden immer in derselben, durch ihre Abhängigkeiten festgelegten Reihenfolge verarbeitet, und jedes Modul wird vollständig (prepare und upgrade) abgeschlossen, bevor das nächste beginnt.

Ist die gespeicherte Version größer als die Zielversion (ältere Programmversion gegen neuere Datenbank), wird kein Skript ausgeführt; die gespeicherte Versionsnummer wird lediglich auf die niedrigere Zielversion gesetzt. Siehe Abschnitt "Rückweg".

### Aufbau einer Upgrade-Datei

Die Skripte liegen im Modul unter /at/workflow/webdesk/<modul>/<implFolder>/update/vNN/ und heißen prepare.xml bzw. upgrade.xml. Das Wurzelelement ist beliebig (üblich ist <upgrade>); die Kindelemente werden der Reihe nach abgearbeitet:

| Element | Erklärung | Attribute |
|---|---|---|
| <sql> | Führt die im Element stehende SQL-Anweisung aus. | dataSource (webdesk oder shark), failOnError, applies und excludes (Datenbanktyp, kommagetrennt: mysql, mssql, oracle, db2, hsql; bei applies auch *) |
| <js> | Führt die JavaScript-Datei aus, deren Name im Element steht. Die Datei liegt im selben vNN-Ordner. | failOnError |
| <java> | Führt die im Element genannte Java-Klasse (vollständiger Name) aus. Die Klasse muss von PoAbstractUpgradeScript abgeleitet sein; sie läuft in einer Transaktion, sofern die Klasse nichts anderes festlegt. | failOnError |
| <rename> | Benennt Systemparameter (Beanproperty) um. | type (derzeit nur beanproperty), old, new, new-module, failOnError |

| | | |
|---|---|---|
| <condition> | Umschließt beliebige andere Elemente, die nur ausgeführt werden, wenn alle angegebenen Bedingungen erfüllt sind. Ohne Attribute ist die Bedingung immer erfüllt. | dataSource, tablesExist, columnsExist (Tabelle.Spalte), columnsNotExist |

Wichtig für die Fehlerbehandlung: failOnError ist standardmäßig **false**. Ein fehlgeschlagener Schritt wird dann nur im Log protokolliert, und das Upgrade läuft weiter und gilt als erfolgreich.

### Ergebnisse und Fehler finden

Es gibt keine eigene Maske mit einem Protokoll der Upgrade-Schritte. Die Ergebnisse stehen im Webdesk-Log (Datei webdesk.log bzw. error.log im Ordner webdesk-logs unterhalb des Cocoon-Arbeitsverzeichnisses, sowie auf der Konsole bzw. im Container-Log). Relevante Meldungen:

| Meldung im Log | Bedeutung | Handlungsbedarf |
|---|---|---|
| Starting to upgrade module ... from vX to vY und Finished upgrading module ... | Beginn und Ende des Upgrades eines Moduls. | Keiner. |
| Executing .../vNN/upgrade.xml (bzw. prepare.xml) | Welche Datei gerade ausgeführt wird. | Keiner. |
| Upgrade script couldn't be executed normally! mit anschließend Failed sql command is: ... | Ein Schritt mit failOnError false ist fehlgeschlagen; das Upgrade lief weiter. | Jede Meldung prüfen. Manche Schritte sind bewusst so angelegt, dass sie auf bestimmten Datenbanken fehlschlagen dürfen; im Zweifel den Support fragen. |
| Couldn't upgrade module ... vX to vY | Ein Schritt mit failOnError true oder eine unlesbare Upgrade-Datei hat das Upgrade abgebrochen. | Siehe "Wenn ein Upgrade-Schritt fehlschlägt". |

## Registrierung

### Was die Registrierung macht

Die Programmdateien einer neuen Version bringen Aktionen (act-descr.xml), Konfigurationen, Textbausteine (i18n.xml), Jobs, Konnektoren, Schlüsselwerte, Hilfetexte und Menüvorlagen mit. Die Registrierung überträgt diese in die Datenbank. Ohne Registrierung bleiben die Datenbankinhalte auf dem Stand der Vorversion: neue Aktionen, Menüeinträge und Beschriftungen sind dann nicht vorhanden.

Die Registrierung läuft in zwei Phasen: Phase 1 ("ESSENTIALS") registriert alles außer den Textbausteinen, Phase 2 ("LABELS (i18n)") die Textbausteine. Sie ist rechenintensiv und kann mehrere Minuten dauern.

### Werte von webdesk.runRegistration

| Wert | Verhalten beim Start | Typische Verwendung |
|---|---|---|
| 0 | Es wird nicht registriert. | Normalbetrieb ohne Update. |
| 1 | Vollständige Registrierung (beide Phasen) **unter Ignorieren des INI-File-Caches**, danach wird der Wert in der webdesk.properties automatisch auf 0 gesetzt. | Standard für ein Update. Läuft nur einmal. |

| | | |
|---|---|---|
| 2 | Bei jedem Start vollständige Registrierung (zuerst alles außer Textbausteine, dann die Textbausteine). Der INI-File-Cache wird verwendet. | Test- und Entwicklungssysteme. |
| 3 | Bei jedem Start Registrierung ohne Textbausteine. | Wenn Textbausteine nicht angefasst werden sollen. |
| 4 | Wie 2, aber nach Abschluss beendet sich die Webdesk-JVM mit Exit-Code 0. Laut Quelltext nur für automatisierte Deployments und Updates mit Kubernetes gedacht (nicht in der Vorlage dokumentiert). | Einmaliger Update-Container, siehe Abschnitt Container. |

Besonderheiten:

- Ist die Datenbank noch leer (Neuinstallation), wird unabhängig von der Einstellung immer eine vollständige Registrierung im Vordergrund ausgeführt.
- Das automatische Zurücksetzen von 1 auf 0 schreibt in die Datei webdesk.properties. Gibt es diese Datei nicht (z. B. im Container), wird nichts zurückgesetzt, und der Wert 1 gilt bei jedem Start. Wird der Wert über die Umgebungsvariable WD_WEBDESK_RUNREGISTRATION gesetzt, bleibt diese maßgeblich: sie ist nach dem Update zu entfernen oder auf 0 zu setzen, sonst wird bei jedem Start vollständig registriert.
- Die Registrierung läuft standardmäßig im Hintergrund, die Anwendung ist währenddessen schon erreichbar. Anwender erst zulassen, wenn im Log REGISTRATION HAS FINISHED (bzw. REGISTRATION All Phases finished!) erscheint.
- Der **INI-File-Cache** (Dateien actions.ini und textmodules.ini im Unterordner inifiles des Cache-Verzeichnisses) merkt sich pro act-descr.xml und i18n.xml, wann die Datei zuletzt geändert war. Ist eine Datei nicht neuer als der gemerkte Stand, wird sie übersprungen. Wird der Cache ignoriert, werden alle Dateien neu gelesen. Der Cache gilt nur für die zugehörige Datenbank: bei geänderter Datenbankverbindung wird er verworfen.
- Kundenspezifische Änderungen werden geschützt: Ausgelieferte Konfigurationen und Textbausteine werden von der Registrierung nur überschrieben, wenn bei ihnen das Kennzeichen "Änderung bei Versionswechsel" (allowUpdateOnVersionChange) gesetzt ist. Neu angelegte Konfigurationen erhalten es nicht (sie werden danach nicht mehr überschrieben). Das Kennzeichen aller Textbausteine kann über die Aktionen "Entsperre alle Textmodule" bzw. "Sperre alle Textmodule" in "Erw. Funktionen" (siehe Erweiterte Funktionen[17]) gesammelt gesetzt bzw. entfernt werden; "Entsperre alle Textmodule" erlaubt der Registrierung, alle ausgelieferten Textbausteine mit den Ursprungswerten zu überschreiben.

## Registrierung manuell auslösen

In der Aktion "Erw. Funktionen" (Erweiterte Funktionen[18]) lässt sich die Registrierung im laufenden Betrieb anstoßen, z. B. wenn webdesk.runRegistration auf 0 stand oder einzelne Module fehlgeschlagen sind:

- "Registrierungsmodus": *complete* (alles), *Everything except Textmodules, Actions and Flowscripts* (ohne Jobs und Job-Konfigurationen) oder *Only text modules*.
- "INI-File cache ignorieren": nur bei *complete* auswählbar; entspricht dem Verhalten bei Wert 1.
- "Einschränkung der Registrierung auf:": registriert nur das gewählte Modul.
- Button "Starte Registrierung".
- Zusätzlich gibt es "INI-File Caches leeren" (die nächste Registrierung liest dann alle Dateien neu) und "Starte Modulupdate" (führt die Modul-Upgrades wie beim Start aus, ohne Neustart).

**Achtung:** Die Meldung "Registration successfully finished!" erscheint auch dann, wenn die Registrierung einzelner Module fehlgeschlagen ist, denn Fehler werden je Modul nur ins Log geschrieben. Der Erfolg ist daher immer im Log zu prüfen. Registrierung nicht während der Geschäftszeit ausführen.

## Checkliste vor dem Update

- Release-Hinweise ("Neue Features", "Breaking Changes") der Zielversion gelesen; Java-Version laut Tabelle oben und die JVM-Parameter geprüft.
- Wartungsfenster vereinbart, Anwender informiert. In "Aktive Jobs" (Aktive Jobs[19]) prüfen, dass keine Jobs mehr laufen.
- **Sicherung** (Details auf der Seite "Backup, Restore und Verschlüsselungsschlüssel"):
  - Vollsicherung der Datenbanken **webdesk** und **shark**.
  - Dateien webdesk.properties und enc.key (liegen im Klassenpfad der Webanwendung, üblicherweise WEB-INF/classes) sowie ggf. tacon.properties des TA-Connectors. Ohne den Verschlüsselungsschlüssel lassen sich gespeicherte Zugangsdaten nicht mehr entschlüsseln.
- Bei Systemen mit Kundenanpassungen: Update zuvor an einer Kopie der Datenbanken testen.
- Die aktuellen Logdateien sichern: Die Logdateien werden laut log4j2.xml nicht angehängt, sondern beim Start der neuen Version neu begonnen.
- Bei Betrieb mit mehreren Knoten (Cluster): Vorgehen für den ersten Knoten festlegen (siehe unten).

## Update mit klassischem Tomcat (WAR)

1. Alle Vorbereitungen der Checkliste abschließen, Anwender abmelden.
2. Tomcat bzw. den Webdesk-Dienst beenden.
3. Sicherung durchführen (Datenbanken, webdesk.properties, enc.key, Logs).
4. In der webdesk.properties den Wert webdesk.runRegistration = 1 setzen. Hintergrund: Nach dem ersten Start steht der Wert auf 0, die Registrierung würde sonst ausbleiben und neue Aktionen, Menüeinträge und Beschriftungen wären nicht vorhanden. Die Modul-Upgrade-Skripte laufen unabhängig davon immer.
5. Die alte Webanwendung (entpackter Ordner) entfernen und das neue WAR bereitstellen bzw. entpacken. Die gesicherte webdesk.properties und enc.key müssen anschließend wieder im Klassenpfad (WEB-INF/classes) der neuen Webanwendung liegen, ebenso weitere dort abgelegte kundenspezifische Dateien.
6. Tomcat starten und das Webdesk-Log beobachten, bis der Start beendet und die Registrierung abgeschlossen ist (Meldungen WebdeskStartup PoStartupImpl.start() finished after ... und REGISTRATION HAS FINISHED AFTER ...; diese Meldung kommt je Phase).
7. Prüfung nach dem Update durchführen (siehe unten).

## Update mit Container-Image

1. Das neue Image enthält keine webdesk.properties. Die Konfiguration und der Schlüssel (webdesk.properties, enc.key) müssen wie bisher bereitgestellt werden. Folgende Einstellungen können alternativ über Umgebungsvariablen der Form WD_<EIGENSCHAFT> gesetzt werden (Punkte werden zu Unterstrichen, Großschreibung, z. B. webdesk.connection.url wird zu WD_WEBDESK_CONNECTION_URL): Datenbankverbindungen (webdesk.connection.*, shark.connection.*, webdesk.dialect), webdesk.ta.if6020impl.rmi, Cluster-Einstellungen, webdesk.runRegistration, webdesk.disableTimeLineChecks und einige weitere.
2. Datenbanken sichern, bisherige Container beenden.
3. Registrierung festlegen, entweder
   - (empfohlen für automatisierte Abläufe) einen einmaligen Update-Container mit webdesk.runRegistration = 4 starten. Er führt Schemaupdate, Modul-Upgrades und die

vollständige Registrierung aus und beendet sich danach selbst. Erst danach die regulären Container mit Wert 0 starten; oder
    - die regulären Container mit Wert 1 starten.
4. Bei mehreren Knoten, die dieselbe Datenbank verwenden, zuerst nur einen Knoten starten und dessen vollständigen Start abwarten, bevor weitere gestartet werden. Hintergrund: Die Upgrade-Schritte werden beim Start jedes Knotens ausgeführt.
5. Umgebungsvariable WD_WEBDESK_RUNREGISTRATION nach dem Update entfernen bzw. auf 0 setzen (siehe oben).
6. Prüfung nach dem Update durchführen.

**Hinweis:** Das Image der Webanwendung enthält Tomcat 9 und die Sitzungsablage über Redis (context.xml). Der Redis-Dienst muss für den neuen Container erreichbar sein.

## Prüfung nach dem Update

1. **Log:** Es gibt keine Meldungen Couldn't upgrade module, Registration of ... module failed oder Action Registration failed!. Meldungen Upgrade script couldn't be executed normally! wurden geprüft.
2. **Anmeldung:** Anmelden als Administrator ist möglich; die Aktion "Versions-Popup" zeigt Version, Revision und Build-Zeitpunkt der neuen Programmversion.
3. **Module:** In der Aktion "Module" sind alle erwarteten Module vorhanden, die Spalte "deaktiviert" steht bei den lizenzierten Modulen auf false. Nach einem weiteren Neustart zeigt das Log für alle Module is up-to-date, d. h. alle Upgrade-Schritte sind abgeschlossen.
4. **Jobs:** In "Aktive Jobs" sind die erwarteten Jobs geplant und laufen. Falls nicht, in "Erw. Funktionen" die Aktion "Neuinitialisierung JobService" ausführen.
5. **Caches:** Die Registrierung leert die Caches der Aktionen und Zugriffsrechte selbst. Werden Berechtigungen trotzdem nicht wirksam, in "Erw. Funktionen" "Berechtigungs Cache leeren" ausführen; bei geänderten Flowscripts "Flowscripts neu laden".
6. **Registrierung:** webdesk.runRegistration steht wieder auf 0 (Betrieb nicht dauerhaft mit 2, 3 oder 4).
7. **Fachliche Stichproben:** Zentrale Abläufe (Zeiterfassung, Workflows, Reports, Schnittstellen und kundenspezifische Skripte) testen. Bei IF6020 den TA-Connector und tacon.properties gemäß den Hinweisen der Zielversion prüfen.

## Wenn ein Upgrade-Schritt fehlschlägt

Was der Code im Fehlerfall tut:

- Bricht eine upgrade.xml oder prepare.xml ab (Meldung Couldn't upgrade module ...), startet Webdesk nicht. Die weiteren Module werden dann nicht mehr verarbeitet.
- Die Versionsnummer des betroffenen Moduls bleibt auf der letzten vollständig abgeschlossenen Version stehen. Beim nächsten Start setzt Webdesk bei der fehlgeschlagenen Version wieder an. Dabei laufen auch die bereits ausgeführten Schritte dieser Version und alle prepare.xml der anstehenden Versionen erneut.
- Es gibt keine automatische Rücknahme (Rollback) der bereits ausgeführten Schritte.

Vorgehen:

1. Logdateien sichern (werden beim nächsten Start überschrieben) und die Ursache im Log suchen: Meldung Couldn't upgrade module ... mit der zugehörigen Fehlerursache.
2. Ursache beheben (z. B. fehlende Datenbankrechte für DDL-Anweisungen) und Webdesk erneut starten. Alternativ in "Erw. Funktionen" die Aktion "Starte Modulupdate" ausführen; Fehler werden dort in der Maske ausgegeben.

3. Lässt sich die Ursache nicht beheben, die Sicherung einspielen (siehe nächster Abschnitt) und den Support kontaktieren. Datenbank nicht manuell nacharbeiten, solange unklar ist, welche Schritte bereits gelaufen sind.

## Rückweg (Rollback)

**Achtung:** Webdesk 4 führt keine Downgrade-Skripte aus. Die readme.txt im Update-Ordner des Moduls erwähnt zwar eine optionale downgrade.xml, im Programm wird eine solche Datei aber nirgends ausgeführt (und im Repository existiert keine). Wird eine ältere Programmversion gegen eine bereits aktualisierte Datenbank gestartet, werden keine Schritte rückgängig gemacht, und das Hibernate-Schemaupdate entfernt keine Spalten; Webdesk setzt lediglich die gespeicherten Modulversionen auf die niedrigeren Werte. Der Rückweg ist deshalb das Einspielen der Sicherung (Datenbanken **webdesk** und **shark** sowie webdesk.properties und enc.key) gemeinsam mit der alten Programmversion.

## Registrierung fehlgeschlagen

Fehler bei der Registrierung eines Moduls werden im Log gemeldet (Registration of ... module failed), Webdesk läuft jedoch weiter. Nach Beheben der Ursache die Registrierung in "Erw. Funktionen" erneut auslösen, bei Bedarf eingeschränkt auf das betroffene Modul und mit "INI-File cache ignorieren".

1. /daisy/webdesk-manual/installation/7572-dsy.html
2. /daisy/webdesk-manual/installation/7506-dsy.html
3. /daisy/webdesk-manual/2455-dsy/g9/10823-dsy.html
4. /daisy/webdesk-manual/2455-dsy/g45/8858-dsy.html
5. /daisy/webdesk-manual/2455-dsy/g7/10971-dsy.html
6. /daisy/webdesk-manual/2455-dsy/g3/11333-dsy.html
7. /daisy/webdesk-manual/2455-dsy/g2/11359-dsy.html
8. /daisy/webdesk-manual/2455-dsy/g1/11499-dsy.html
9. /daisy/webdesk-manual/2455-dsy/g2/11359-dsy.html
10. /daisy/webdesk-manual/2455-dsy/g1/11418-dsy.html
11. /daisy/webdesk-manual/2455-dsy/g9/10823-dsy.html
12. /daisy/webdesk-manual/2455-dsy/g1/11499-dsy.html
13. /daisy/webdesk-manual/2455-dsy/g1/11418-dsy.html
14. /daisy/webdesk-manual/installation/7572-dsy.html
15. /daisy/webdesk-manual/2455-dsy/g1/11499-dsy.html
16. /daisy/webdesk-manual/2455-dsy/g9/10823-dsy.html
17. /daisy/webdesk-manual/admin/po-overview/3731-dsy/463-dsy.html
18. /daisy/webdesk-manual/admin/po-overview/3731-dsy/463-dsy.html
19. /daisy/webdesk-manual/admin/po-overview/3731-dsy/2662-dsy/1146-dsy.html