Inhaltsverzeichnis

Kernmodul-Regelbetrieb

BI Maintenance

Ziel und Überblick

Die hier bereitgestellten Skripte ermöglichen es, in der BI-Umgebung von HISinOne und in Zukunft auch von SuperX (aktuell können die Scripte in SuperX leider noch nicht verwendet werden) Modul-Updates und -Upgrades zuverlässig über die Shell auszuführen – automatisiert per Cronjob oder manuell. Sie orientieren sich bewusst am bisherigen Vorgehen aus SuperX, wurden jedoch erweitert:

- Ausführung der BI-Modul-Updates/-Upgrades über Java (*ComponentAdminCLI*).
- Vollständige Protokollierung in Logdateien.
- Optional: Protokollierung der Läufe in der Tabelle update_prot.

- Automatischer Mailversand über ein konfigurierbares Mailprogramm.
- Optional: Erkennen interner Fehler im Batch-Job (auch wenn Java Exitcode 0 liefert).
- Möglichkeit, Logdateien automatisch an Mails anzuhängen (erfolgreiche und fehlerhafte Module).

Installation aus dem git Repository

Führen Sie folgenden Shell-Befehl aus:

```
git clone https://git.campussource.de/git/SuperX/BI_Maintenance.git
```

Die weitere Konfiguration wird im Folgenden beschrieben. Alle Einstellungen erfolgen zentral in der Datei BI_ENV, die als Template BI_ENV.sam ausgeliefert wird.

Umgebungsvariablen in der BI_ENV

BI_ENV.sam – Template und lokale BI_ENV

Die Datei BI_ENV.sam wird als Muster ausgeliefert.

Sie muss vor Ort:

1. in BI_ENV kopiert/umbenannt werden:

```
cp BI_ENV.sam BI_ENV
```

1. an die lokalen Gegebenheiten angepasst werden (Pfade, Module, Mailadressen usw.).
2. mit restriktiven Rechten versehen werden:

```
chmod  600  BI_ENV
```

Skripte binden diese Datei später mit

```
.  /pfad/zu/BI_ENV
```

ein.

Bedeutung der Variablen

Im Folgenden die wichtigsten Variablen, die angepasst werden müssen.

Java-Konfiguration

```
JAVA_HOME=/usr/lib/jvm/ja
va-17-openjdk-amd64
```

Diese Variablen stellen sicher, dass die BI-Jobs mit dem vorgesehenen Java (empfohlen: Java 17) ausgeführt werden.

Java-Optionen:

```
JAVA_OPTS="-Xmx1520M
-Djava.awt.headless=true
```

Pfade zur SuperX-BI-Installation

```
WEBAPP=/var/lib/tomcat10/
webapps/superx
```

Diese Pfade müssen an lokale Tomcat-Installation und SuperX-Verzeichnisstruktur angepasst werden.

Modulsteuerung

Für die Update- und Upgrade-Skripte werden die zu bearbeitenden Module festgelegt:

```
export
BI_UPDATE_MODULES="s
```

Hinweis:

Die Modulkürzel müssen klein geschrieben sein (z. B. sos, kenn, zul) und mit Leerzeichen getrennt aufgelistet werden.

Logging

```
LOGPFAD=$WEBAPP/WEB-INF/logs
```

Im Logpfad werden u. a. folgende Dateien erzeugt:

- bi_update.log – Sammellog des Updates
- bi_upgrade.log – Sammellog des Upgrades
- <modul>_update.log
- <modul>_upgrade.log

Java-Batch-Jobs erzeugen ergänzende Logs in:

$WEBAPP/WEB-INF/logs/jobs

Mailversand

Folgende Variablen steuern Empfänger und Format der Benachrichtigungen:

```
export ERRORMAIL="admin@hs.d
```

- ERRORMAIL – Empfänger für Fehlermails
- LOGMAIL – Empfänger für Erfolgs- und Statusmails

Mehrere Adressen werden per Leerzeichen getrennt.

Mailprogramm:

```
export  MAILPROG="s-nail --account=test1  -S
```

Betreffzeilen:

```
export MAIL_BETREFF_UPDATE
```

Steuerung der Log-Anhänge

```
# error  = Logs nur bei
Fehlern anhängen
```

Optionale Prüfung der Modul-Logs

```
# true  = zusätzlich
Modul-Log auf interne
```

Gerade bei dem Java Aufruf von ComponentAdminCLI empfehlenswert, da dieser aktuell noch trotz status: FAILED oft Exitcode 0 liefert.

Module Updates

modules_update.sh – Hauptskript

Dieses Skript führt alle Module aus BI_UPDATE_MODULES nacheinander aus.

Ablauf:

1. Startcheck: Sind WEBAPP, LOGPFAD und BI_UPDATE_MODULES gesetzt?

1. Falls verfügbar: DB-Protokollierung via DOQUERY.

1. Für jedes Modul:
    1. Logdatei anlegen
    2. Start in update_prot protokollieren (update_id = -10000)

1. Java-Update starten:

```
ComponentAdminCLI  -e  <modul>
```

1. Optional: Modul-Logdatei nach internen Fehlern durchsuchen
2. Erfolg:
    1. Modul-Log in SUCCESS_LOG_FILES

1. DB-Update (update_id = -10000)
2. Fehler:
    1. Modul-Log in ERROR_LOG_FILES

1. DB-Update (update_id = -10001)
2. Zuletzt: Java-Joblogs aus $WEBAPP/WEB-INF/logs/jobs ermitteln

1. Nach Abschluss aller Module:
    1. Erfolgs- oder Fehlermail versenden
    2. Anhänge abhängig von MAIL_ATTACH_LOGS_MODE

modules_update_cron.sh – Wrapper für Cron

Damit Updates regelmäßig durchgeführt werden können, existiert ein einfaches Wrapper-Skript.

Vorgehen:

1. Beispieldatei kopieren:

```
cp
modules_update_cron.sh.s
```

1. Pfade zur BI_ENV und zum Update-Skript anpassen.

1. Cronjob eintragen, z. B. werktags um 18 Uhr:

```
0  18  *  *  1-5
/pfad/zu/modules_update_c
```

Inhaltlich:

- Laden der BI_ENV
- Start des Skripts modules_update.sh

Module Upgrades

modules_upgrade.sh – Hauptskript

Das Upgrade-Skript entspricht dem Update-Skript, unterscheidet sich aber in folgenden Punkten:

- Es wird **manuell** ausgeführt – kein Cronjob vorgesehen.
- Es verwendet BI_UPGRADE_MODULES.
- Das eigentliche Upgrade erfolgt über:

```
ComponentAdminCLI  -u  <modul>
```

- Nach Abschluss des Upgrades erfolgt ein Mailversand analog zum Update-Skript.

Aufruf:

```
cd
/var/lib/tomcat10/webapps/s
```

Vorher muss zu Beginn des Skripts der Pfad zur BI_ENV eingetragen sein:

```
. /pfad/zur/BI_ENV
```

Modulverwaltung

Modulkürzel

Die folgenden Modulkürzel sind in einer typischen BI-Installation relevant:

```
kuerzel |
```

Die aktiven Module der eigenen Installation können mit folgendem SQL abgefragt werden:

```
SELECT V.his_system AS
kuerzel,
```