Serverprozesse in JDito

AID 026 DE

Autor: FA, MW, JK, KN. Version 10.3. Zuletzt geändert 04.09.2017

Version	Änderungen
10.3	Anpassung der Formatierungen
10.2	Serverprozesse aus dem Referenzsystem eingefügt
10.1	Code-Conventions angepasst
10.0	Anpassung an ADITO4.6
2.0	Letzter Stand vor Übernahme in Versionierung

Inhaltsverzeichnis

1. Einführung

In vielen Installationen von ADITO4 wird eine zyklische Ausführung von Prozessen gewünscht, die auch dann ausgeführt werden, wenn kein Client angemeldet ist. Mögliche Anwendungen reichen von regelmäßigen Datenübernahmen über Prüfroutinen bis hin zur Bereinigung von Datenbankeinträgen. Dieses Dokument beschreibt die Integration eines Systems für die zeitgesteuerte Ausführung von JDito-Prozessen auf dem ADITO4 Server.

Bitte beachten Sie, dass die Ausführung solcher serverbasierten Prozesse ebenfalls Last auf dem Anwendungsserver verursacht. Beschränken Sie den Einsatz solcher Prozesse auf das notwendige Minimum und legen Sie die Ausführungsintervalle möglichst lang aus, damit noch genügend Leistung für die Bearbeitung von Clientanfragen vorhanden ist. Sollten Sie eine Vielzahl von zeitgesteuerten Prozessen benötigen, empfehlen wir die Installation einer eigenen Serverinstanz nur für die Ausführung von solchen Prozessen.

2. Einrichten des Timersystems

2.1. Systemtabellen erzeugen

2.1.1. ASYS_TIMER

Diese Tabelle ist nach der Installation von ADITO4 bereits vorhanden. Hier sind keine Änderungen notwendig.

2.1.2. AOSYS_TIMER

Falls diese Tabelle noch nicht vorhanden ist, muss sie erzeugt werden. Diese Tabelle muss in der gleichen Datenbank / im gleichen Schema wie die Tabelle ASYS_TIMER liegen und besitzt den in der folgenden Tabelle geschilderten Aufbau.

Spalte	Datentyp	Nullwerte, Constraints
TIMERID	varchar max. 255 Zeichen	**NOT NULL / primary key**
AO_INTERVAL	integer	**NULL**
TIMERSTATE	integer	**NULL**
NEXTSTART	Zeitangabe (sekundengenau)	**NULL**
PROCESSNAME	varchar max. 255 Zeichen	**NULL**
EXITCODE	integer	**NULL**
PROCESSSTATE	integer	**NULL**
STARTTIME	decimal(18,0)	**NULL**

2.2. Serverautostart anpassen

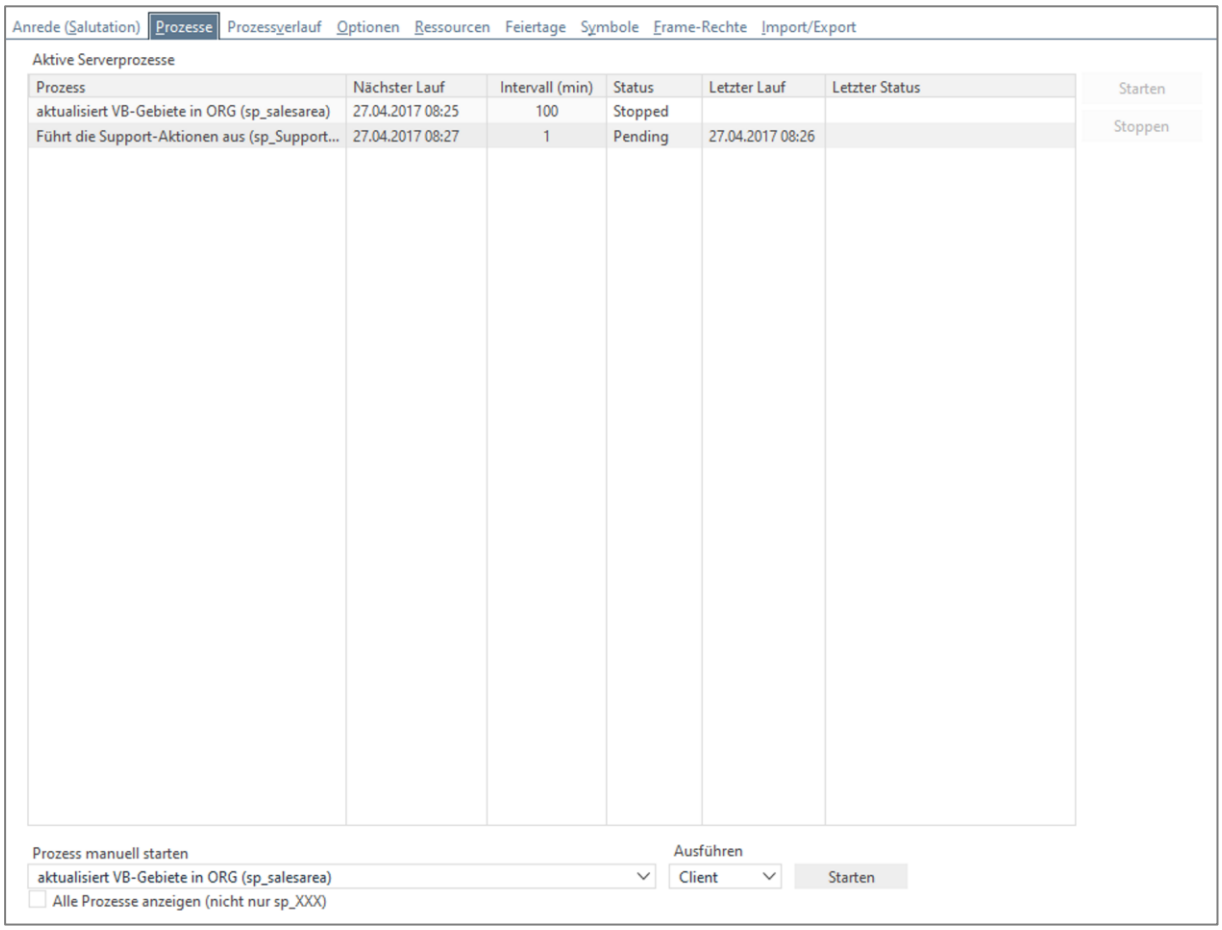
Ergänzen Sie den Inhalt für den Serverautostart-Prozess um die folgenden Zeilen, wenn noch nicht eingetragen. Beachten Sie, dass alle import-Anweisungen immer zu Beginn des Prozesses stehen müssen.

```
import("lib_timer");
initTimers();
```

3. Serverprozesse verwalten

3.1. Verwaltung in der Systemverwaltung

Die untenstehende Abbildung zeigt den Verwaltungsreiter für die Hintergrundprozesse im Frame Systemverwaltung. Die Tabelle zeigt alle für die Ausführung als Serverprozess eingetragenen JDito-Prozesse.



3.1.1. Hinzufügen eines Prozesses

Über die Combobox "Prozess" lässt sich ein neuer Prozess eintragen. Dafür muss eine Startzeit im Bereich von 00:00 bis 23:59 eingetragen sein und ein Ausführungsintervall in Minuten gesetzt werden. Danach wird die Schaltfläche [Eintragen] aktiviert und der Prozess kann für die Ausführung im Hintergrund registriert werden.

Wichtiger Hinweis: Die Liste der Combobox enthält nur Prozesse, die im Prozess-Editor einen Titel erhalten haben. Prozesse ohne einen Eintrag für das Feld "Titel" werden nicht in die Liste aufgenommen. Das kleinste zulässige Intervall für die Ausführung ist eine Minute. Nach Möglichkeit sollten Sie Prozesse nicht öfter als einmal pro Viertelstunde ausführen, um dem Server nicht zuviele Kapazitäten für die Abarbeitung von Clientanfragen zu entziehen.

Alle neu eingetragenen Prozesse erhalten den Status "Stopped". Erst nach Auswahl des jeweiligen Prozesses und einem Klick auf die Schaltfläche [Starten] wird der Prozess in die Liste der aktiven und auf die Ausführung wartenden Prozesse eingetragen.

3.1.2. Anhalten und Entfernen von Prozessen

Ein Klick auf die Schaltfläche [Stoppen] hält den ausgewählten Prozess an. Er verbleibt aber in der Liste der registrierten Serverprozesse und kann bei Bedarf wieder gestartet werden.

Ein Klick auf die Schaltfläche [Entfernen] löscht die Registrierung für den ausgewählten Prozess und entfernt diesen aus der Liste.

3.1.3. Manuelles Starten von Hintergrundprozessen

Soll ein für die Ausführung im Hintergrund registrierter Prozess manuell gestartet werden, kann dafür die Combobox "Prozess manuell starten" verwendet werden. Aus dieser wird ein Prozess ausgewählt und nach dem Klick auf die untere Schaltfläche [Starten] ausgeführt. In dieser Combobox sind in der Liste nur solche Prozesse verfügbar, die sich im Status "Gestoppt" befinden.

Bitte beachten Sie, dass die Prozesse unter Umständen erst nach einem Serverneustart oder bei Erreichen des letzten Startzeitpunktes (also nach 24 Stunden) wieder in den Timer eingegliedert werden. Der Status kann deshalb manchmal länger auf „Pending“ verbleiben.

4. Erstellen von Server-Prozessen

Generell kann nahezu jeder Prozess auch auf dem Server ausgeführt werden. Sie sollten allerdings daran denken, dass bei einer zeitgesteuerten Ausführung im Hintergrund keine Interaktionen mit einem Benutzer möglich sind. Die Verwendung von `question.showMessage()` oder anderen Konstrukten ist daher nicht möglich.

Bevor Sie einen Prozess zeitgesteuert auf dem Server ausführen lassen, sollten Sie überprüfen, ob der Code für diesen Prozess die folgenden Voraussetzungen erfüllt. Falls nicht, ergänzen bzw. ändern Sie bitte den Code entsprechend ab.

4.1. Voraussetzungen

Ein Prozess, der im Hintergrund ausgeführt werden soll, muss die Bibliothek `lib_timer` importieren, in der die entsprechenden Hilfsfunktionen enthalten sind. Für die Benutzung wichtig sind die folgenden Funktionen:

- `timerProcessStartup()` Beim Start ausführen (s.u.)
- `timerProcessShutdown()` Beim Beenden ausführen (s.u.)

4.2. Jdito-Code

4.2.1. Code-Konventionen

Jeder Serverprozess sollte in seinem Grundgerüst sicherstellen, dass keine Exception nach außen gelangt, die nicht abgefangen wurde. Zusätzlich wird von jedem Prozess gefordert, dass er sich beim Timersystem „anmeldet“ und bei der Beendigung wieder „abmeldet“.

Das nachfolgende Listing zeigt das empfohlene Grundgerüst für einen Serverprozess.

```
import("lib_log");


/*
   Grundgeruest eines Timer-Prozesses
*/
runProcessTimerWithLog("sp_test", function()
{
     var proccesRes = LogProcessResult.success;

     //weiterer Code, in dem bestimmte Aktionen erledigt werden

     return proccesRes;
})
```

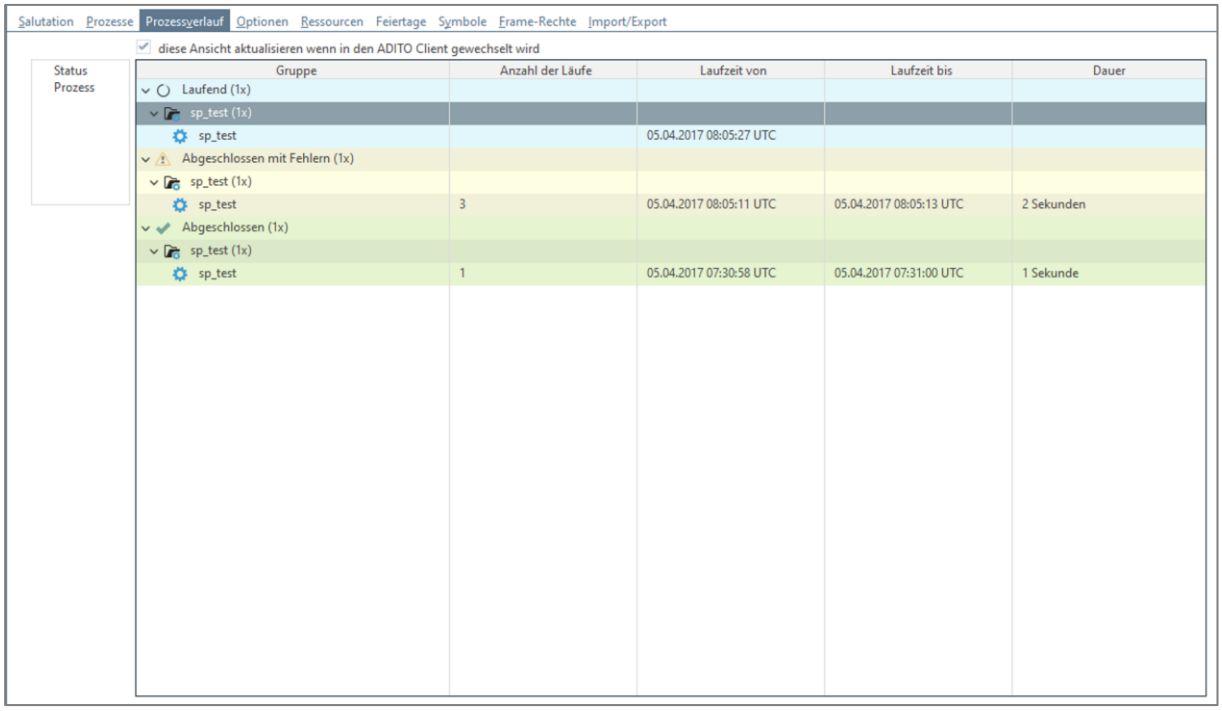
`runProcessTimerWithLog` muss verwendet werden, damit der Prozess bei der Ausfühhrung das Ergebnis in den `Prozessverlauf` der `Systemverwaltung` einträgt. Wird dieser nicht verwendet, kann das Ergebnis nicht mehr abgerufen werden.

`LogProcessResult` stellt in diesem Fall das Ergebnis der Ausführung dar. Dieses muss vom Prozess zurückgegeben werden, kann aber vom Entwickler selbst auf den jeweiligen Status gesetzt werden. `LogProcessResult` kann folgenden Möglichkeiten entsprechen:

- `completetWithErrors`: Der Prozess wurde abgeschlossen und während der Ausführung traten Fehler auf.
- `failed`: Die Ausführung war nicht erfolgreich.
- `running`: Der Prozess läuft noch und ist aktuell nicht abgeschlossen.
- `success`: Der Prozess wurde erfolgreich abgeschlossen.
- `unknown`: Über den Status der Ausführung ist nichts genaueres bekannt.

4.2.2. Status der letzten Ausführung

Den Status der letzten Ausführung können Sie, soweit der Serverprozess dem Grundgerüst entspricht, dem Prozessverlauf der Systemverwaltung entnehmen.



4.3. Zeitzonen

Alle Prozesse, die auf dem Server ausgeführt werden, laufen mit der Zeitzoneneinstellung UTC. Insbesondere bei allen Datumskonvertierungen ist darauf zu achten, die korrekte Zeitzone zu verwenden. Soll ein JDito-Prozess sowohl manuell als auch unter der Kontrolle des Timersystems verwendet werden, dann sollten explizit die Zeitzonen bei allen Aufrufen angegeben werden.

5. Prozessverwaltung mit JDito

Falls Sie die Hintergrundprozesse selbst über JDito steuern wollen oder eine eigene Oberfläche für die Verwaltung erstellen wollen, finden Sie hier die Dokumentation zu den einzelnen Funktionen der Timer-Bibliothek

5.1. Bibliothek lib_timer

Die Funktionen sind in der Bibliothek `lib_timer` enthalten. Innerhalb dieser Bibliothek sind keine Anpassungen erforderlich. Sie dient nur zur Gruppierung der darin enthaltenen Funktionen.

Alle in der folgenden Funktionsreferenz genannten Funktionen, die sich sonst noch in der Bibliothek befinden, sind für die interne Nutzung innerhalb der Bibliothek gedacht. Diese Funktionen und ihre Parameter können sich ohne Ankündigung ändern und sollten nicht verwendet werden. Beschränken Sie die Nutzung auf die unten aufgelisteten Funktionen.

5.2. Funktionsreferenz

5.2.1. initTimers()

Diese Funktion darf nur innerhalb des Serverautostart-Prozesses einmal aufgerufen werden und startet das Timersystem.

5.2.2. timerProcessStartup(pProcess)

Diese Funktion muss beim Start eines Timerprozesses aufgerufen werden, damit der Ausführungszustand im Timersystem aktualisiert werden kann. Als Parameter wird der Name des Prozesses übergeben.

`pProcess` Der Name des JDito-Prozesses in der gleichen Schreibweise im Designer vorhanden.

Die Funktion liefert kein Ergebnis.

5.2.3. timerProcessShutdown(pProcess, pExitCode)

Diese Funktion muss beim Beenden eines Timerprozesses aufgerufen werden, damit der Ausführungszustand im Timersystem aktualisiert werden kann. Als Parameter werden der Name des Prozesses und ein Exitcode übergeben.

`pProcess` Der Name des JDito-Prozesses in der gleichen Schreibweise im Designer vorhanden.

`pExitCode` Der Beendigungscode für diesen Prozess (0 = erfolgreich). Alle Werte größer Null werden als Fehlercodes angesehen.

Die Funktion liefert kein Ergebnis.

5.2.4. updateTimer(pProcess, pColumn, pValue)

Diese Funktion setzt den Ausführungsstatus eines Prozesses.

`pProcess` Der Name des JDito-Prozesses in der gleichen Schreibwese im Designer vorhanden.

`pColumn` Spaltenname

`pValue` Werte der übergebenen Spalte.

Die Funktion liefert kein Ergebnis.

6. Technische Details

Beim Server-Start (Prozess `serverautostart`) wird der Prozess `start_timer` über die Funktion `initTimers` der `lib_timer` mit einem Intervall von 30 Sekunden gestartet. Dieser überprüft die Timertabellen, ob ein Prozess auf seinen Start als Timer wartet und führt diesen Start bei Bedarf aus und aktualisiert die entsprechend Informationen.

7. Serverprozesse im Referenzsystem

Im Referenzsystem sind bereits einige Serverprozesse implementiert. Diese werden im Folgenden beschrieben.

7.1. ADITO Designer Admin: XMPP Rolle, XMPP Ids und News-Feed setzen

Prozessname: `sp_setIMoptions`

Funktion: Setzt die Initialwerte für das `InstantMessaging`, also den Chat und die Beobachtungen. Dieser Prozess wird beispielsweise benötigt, wenn ein Altsystem migriert wird und diese Funktionen genutzt werden sollen. In diesem Prozess wird unter anderem die XMPP-JID bei allen Benutzern gesetzt.

Ausführbar am: Client

7.2. Aktualisiert VB-Gebiete in ORG

Prozessname: `sp_salesarea`

Funktion: Setzt anhand der Länderdaten im System das Vertriebsgebiet von Firmen, bei denen es bisher nicht gesetzt wurde. Um festzustellen, zu welchem Vertriebsgebiet die Firmen gehören, werden die Daten der Standardadresse der Firma verwendet.

Ausführbar am: Client

7.3. ASYS_Daten übernehmen

Prozessname: `sp_copy_Sysdata_into_SYSTEMDB`

Funktion: Kopiert die Daten der Tabellen „ASYS_USER“ und „ASYS_ICONS“ in einen anderen Datenbank-Alias. Das ist sinnvoll, wenn man eine gemeinsame Entwicklungs-Datendatenbank und verschiedene Systemdatenbanken hat. Man kann hiermit die Daten von AO_DATEN in die Systemdatenbanken kopieren.

Ausführbar am: Client

7.4. Beispiel Serverprozess mit Logging

Prozessname: `sp_test`

Funktion: Dieser Prozess dient als Beispielprozess für das Logging. Das Ergebnis dieses Prozesses wird im Frame Systemverwaltung im Prozessverlauf angezeigt.

Ausführbar am: Client und Server

7.5. Daten leeren

Prozessname: `sp_emptytable`

Funktion: Dieser Prozess löscht alle Daten, sofern es sich nicht um Mitarbeiter oder die eigene Firma handelt. Daten (z.B. Historien, Attribute, etc.), die mit diesen Mitarbeitern oder der eigenen Firma verknüpft sind, werden ebenfalls behalten.

Ausführbar am: Client

7.6. Datenbank kopieren

Prozessname: `sp_copyRepositoryData`

Funktion: Kopiert die Daten aus einem Datenbank-Alias in einen anderen.

Ausführbar am: Client

7.7. Datenbestand prüfen und bereinigen

Prozessname: `sp_CheckDatabase`

Funktion: Dieser Prozess prüft, ob nicht zugeordnete Datensätze (z.B. Historien) vorhanden sind und löscht diese auf Wunsch. Außerdem wird auf möglicherweise problemverursachende Spaltennamen hingewiesen.

Ausführbar am: Client

7.8. E-Mail Client setzen

Prozessname: `sp_setMailClient`

Funktion: Setzt den bevorzugten E-Mail-Client des Benutzers.

Ausführbar am: Client

7.9. Erzeugt einen techn. LDAP Account

Prozessname: `sp_createLdapAccount`

Funktion: Dieser Prozess erzeugt einen Benutzer, der verwendet wird, um über ein ADITO-Plugin auf das LDAP zuzugreifen.

Ausführbar am: Client und Server

7.10. Exchangesynchronisation

Prozessname: `sp_ExchangeSync`

Funktion: Dieser Prozess synchronisiert die Kontaktdaten aus ADITO mit den Kontaktdaten, die in Exchange hinterlegt sind. Hierfür ist ein Plugin von ADITO notwendig.

Ausführbar am: Server

7.11. Firmen-Dubletten neu berechnen

Prozessname: `sp_getOrgDuplicate`

Funktion: Berechnet die Firmendubletten mit Hilfe des Schlüsselwortes `DuplicateOrgPattern` neu.

Ausführbar am: Client und Server

7.12. FrameIDs anzeigen

Prozessname: `sp_showFrameIDs`

Funktion: Zeigt die IDs aller Frames an. Die IDs werden im Frame-Kommentar des jeweiligen Frames gesetzt. Um zu sehen, ob IDs doppelt vergeben wurden, kann dieser Prozess ausgeführt werden.

Ausführbar am: Client

7.13. Führt den Neuigkeiten Abgleich für die Beobachtungen durch

Prozessname: `sp_log_subscribe`

Funktion: Dieser Prozess sorgt dafür, dass die Benachrichtigungen von Änderungen an abonnierten Datensätzen beim Benutzer angezeigt werden.

Ausführbar am: Server

7.14. Führt die Support-Aktionen aus

Prozessname: `sp_SupportAktion`

Funktion: Es handelt sich hier um einen Beispielprozess um automatisiert aus dem Supportticket E-Mails an die betroffenen Personen zu versenden und Aufgaben für Mitarbeiter zu erstellen. Damit dieser Prozess funktioniert, muss die Absender-Adresse (`Support EMail`) in den Optionen der Systemverwaltung ausgewählt werden. Wird hier eine neue Adresse benötigt, wird sie im Schlüsselwort `Optionen (OPTIONS)` hinzugefügt. Zusätzlich wird der User „support“ mit konfiguriertem SMTP benötigt.

Um Ticketantworten von Kunden automatisch zu verarbeiten, wird zudem eine aktivierte Support-Mailbridge benötigt.

Ausführbar am: Client und Server

Weitere Informationen zum Einrichten der Mailbridge finden Sie im AID019-DE – Konfiguration der Mailbridge.

7.15. Für CTI Telefonnummer bereinigen

Prozessname: `sp_SetComSearchAddr`

Funktion: Setzt für alle Telefon-Kommunikationseinträge eine interne Spalte für CTI. So wird erkannt, ob ein Anrufer bereits hinterlegt ist. Oft ist es sinnvoll, diesen Prozess vor dem Go-Live auszuführen.

Ausführbar am: Client und Server

7.16. Gebiete zuordnen

Prozessname: `sp_assignArea`

Funktion: Ordnet Firmen Gebiete zu, wenn noch kein Gebiet gesetzt ist. Dies geschieht anhand der PLZ und des Landes, sofern diese Daten in der AOSYS_LOCATION gepflegt wurden.

Ausführbar am: Client und Server

7.17. Historien duplizieren für Peformancetest

Prozessname: `sp_insert_hist`

Funktion: Dupliziert die Historien, die im System gespeichert wurden, damit die Performance getestet werden kann.

Ausführbar am: Client

7.18. Import LDAP User

Prozessname: `sp_ActiveDirectoryImport`

Funktion: Synchronisiert ADITO mit dem LDAP. Initial muss `sp_createLdapAccount` ausgeführt werden, sodass der technische Benutzer korrekt angelegt wird.

Ausführbar am: Server

7.19. Import Post-Daten

Prozessname: `sp_ImportDPostData`

Funktion: Legt die Tabellen für die Post-Daten an und importiert diese.

Ausführbar am: Client

Weitere Informationen zu den Post-Daten finden Sie im AID094-DE – Anleitung zum Post-Daten-Import.

7.20. Importer-Modul Beispiel

Prozessname: `sp_importExample`

Funktion: Dieser Prozess dient als Beispiel für den Importer. -Um diesen zu nutzen, muss eine Datei mit dem Namen „Testdaten_Firma_Person.csv“ im Ordner „C:/temp/“ abgelegt werden. Der Aufbau dieser Datei sollte folgendermaßen aussehen:

```
kdnr;Firmenname;Strasse;Hausnummer;PLZ;Ort;Vorname;Nachname;eMail;Telefon;Anrede;Landkreis;Bundesland;Land;eMail (Firma);Telefon (Firma);Fax (Firma);COMRESTRICTION
```

Ausführbar am: Client und Server

7.21. Personen-Dubletten neu berechnen

Prozessname: `sp_getPersDublicate`

Funktion: Berechnet die Personendubletten mit Hilfe des Schlüsselwortes `DuplicatePersPattern` neu.

Ausführbar am: Client und Server

7.22. Portraits für den Messenger kopieren

Prozessname: `sp_copy_xmpp_portraits`

Funktion: Kopiert die Bilder aus den Personen, damit diese auch für den Chat genutzt werden können. So können die Chat-Bilder gecached werden, was zu einer besseren Performance, vor allem bei niedriger Bandbreite, führt.

Ausführbar am: Client und Server

7.23. Rollen der Benutzer bereinigen

Prozessname: `sp_reorg_users_roles`

Funktion: Entfernt nicht vorhandene Rollen aus den Benutzermodellen.

Ausführbar am: Client

7.24. Setzt internationales Nummernformat

Prozessname: `sp_formatTelefonnumbers`

Funktion: Formatiert die gespeicherten Telefonnummern mit Hilfe des `PhoneNumberPlugins`, damit diese international gültig sind. Ist das Plugin nicht hinterlegt, wird auf den Mechanismus des `sp_SetComSearchAddr` zurückgegriffen.

Ausführbar am: Client und Server

7.25. Suchindex neu aufbauen

Prozessname: `sp_runSearchIndexer`

Funktion: Baut den Suchindex für die Indexsuche neu auf.

Ausführbar am: Client und Server

7.26. Terminaktualisierung f.Präsentation

Prozessname: `sp_timeshift`

Funktion:Dieser Prozess ändert die gespeicherten Daten ab, damit diese wieder dem aktuellen Datum entsprechen.

Ausführbar am: Client

7.27. Transfer von ORG in Audit eintragen

Prozessname: `sp_gebietsaenderung`

Funktion: Beispielprozess für die Gebietsänderung, der auch mit Offline-Systemen ausgeführt werden kann, da die Änderungen ins Audit geschrieben werden. Folgende Änderungen werden übernommen: Daten löschen und Daten mit Hilfe der V-Aktion übertragen.

Ausführbar am: Client

Weitere Informationen zur Offline-Synchronisation finden Sie im AID056-DE – Datensynchronisation DBSync.

7.28. Was wird geloggt

Prozessname: `sp_showHistoryLog`

Funktion: Dieser Prozess gibt an, wie viele Tabellen geloggt werden und erstellt eine CSV-Datei, die eine Übersicht über die geloggten Tabellen und Spalten bietet.

Ausführbar am: Client