Kill Them All

Thorsten Foltz
auxmoney GmbH
Königsallee 60F
40212 Düsseldorf
foltz@auxmoney.com

Zusammenfassung

Es wird ein Makro gezeigt, das die aktiven Sessions aller oder ausgewählter Benutzer des Enterprise Guides beendet. Ein (un)beabsichtigtes Blockieren einzelner SAS-Dateien kann somit unterbunden werden.

Schlüsselwörter: Proc Iomoperate, Makro, Enterprise Guide

1 Einleitung

Die meisten dürften es kennen, man arbeitet mit Daten, die nicht in einer Datenbank gespeichert sind, sondern in Form normaler SAS-Dateien (Format .sas7bndx). Dabei hat irgendjemand die Datei geöffnet und diese kann nicht bearbeitet, ersetzet oder gelöscht werden. Im normalen Betrieb kann man sich noch gut verständigen. Ärgerlich bis problematisch ist es hingegen, wenn SAS-Dateien, die regel- oder unregelmäßig über Nacht erzeugt werden, betroffen sind, da die Erstellung unter Umständen mehrere Stunden in Anspruch nimmt.
Vor diesem Hintergrund ergibt sich die Notwendigkeit eines Scripts, das die Verbindungen des Enterprise Guides mit dem Server trennt. Selbstverständlich wird, bei entsprechender Automatisierung, jede Verbindung unterbrochen, auch wenn es für bestimmte Sessions nicht beabsichtigt ist. Dieses Problem wird mit einer zusätzlichen Datei (in diesem Fall eine Exceldatei), in der sich jeder Nutzer eintragen kann, gelöst.

2 Voraussetzungen

Das Script läuft erfolgreich mit der nachfolgenden Konfiguration:

Seitens der Nutzer:

- SAS Enterprise Guide Version 7.1
- Windows 10
- MS Office 365 Business

Seitens des Servers:

- SAS 9.4
- Windows Server 2012

Seitens des Administrators:

- Benutzername
- Passwort
- Host
- Port
- E-Mailport (optional)
- E-Mailhost (optional)

3 Funktionsweise des Scripts

Im ersten Schritt werden die aktiven Sessions festgestellt. Im zweiten Schritt wird eine Exceldatei eingelesen. Diese liegt frei verfügbar auf beispielsweise dem Abteilungslaufwerk und bietet den Anwendern die Möglichkeit, temporär Programme, trotz des „Kill Them All“ Scripts, ausführen zu können. Anschließend wird eine vorgefertigte Liste mit allen Nutzern mit der Tabelle der aktiven Sessions und den Daten der importierten Exceldatei zusammengeführt. Im letzten Schritt trennt dann ein Marco, mittels einer Schleife, die Verbindungen.[1]

3.1 Aktive Sessions

Um flexibel Anpassungen vornehmen zu können, werden einige Parameter in Makrovariablen überführt.

```
%LET host = "Host";
%LET port = 1234;
%LET user = "admin";
%LET password = "xyz";
```

Im ersten Schritt erfolgt die Feststellung der aktiven Sessions. Dies geschieht folgendermaßen:

```
PROC IOMOPERATE;
     CONNECT HOST = &host.
     PORT = &port.
     USER = &user.
     PASS = &password.
     SERVERTYPE = OBJECTSPAWNER;
     LIST SPAWNED OUT = WORK.spawned(WHERE = (LOGICALNAME NE "SASApp
- Logical Stored Process Server"
          AND processowner NE "admin@company"));
QUIT;
```

[1] Die Verbindungen werden getrennt, der Enterprise Guide wird nicht beendet.

Das Ergebnis wird in der Datei „spawend“ hinterlegt (siehe Abbildung 1). Sollte kein Stored Process vorhanden oder eine Beendigung dieses unproblematisch sein, kann die erste „where“-Bedingung ignoriert werden. Die zweite hingegen wird zwingend benötigt, da ansonsten die eigene Session, die alle anderen beenden soll, auch beendet wird und dies unter Umständen vor den anderen.

PROCESSOWNER	SERVERID
muster@AUXMONEY	641A3206-D016-4883-85FF-81D32E34F9BA
test@AUXMONEY	70327C16-FA73-4948-8577-C4C6CFE9FF1B
muster@AUXMONEY	60448117-8BEB-4656-A5E1-427FA54524D9
sasuser@AUXMONEY	3C6E8E1A-00FA-48C3-BECC-8183D3B66A2D

Abbildung 1: SAS-User mit aktiver Session

3.2 Import der ausgenommenen User

Da es vorkommen kann, dass einzelne Anwender Programme über Nacht oder sogar über mehrere Tage laufen lassen (müssen), sollte eine Möglichkeit vorhanden sein, dass jeder Nutzer selbst entscheiden kann, ob er vorübergehend die automatisierte Beendigung seines Programmes verhindern möchte. Da sich Excel als Standardbürosoftware etabliert hat, einfach zu bedienen ist und SAS das Format problemlos verarbeiten kann, bietet es sich an, für den Zweck der Eintragung eine Excelliste zu führen.
In dieser Liste müssen lediglich 2 Spalten vorhanden sein. Eine Spalte mit dem Benutzernamen und das Datum bis zu dem der User seine Session(s) nicht beendet haben will. Bei dem Datum empfiehlt es, sich, dieses als String zu speichern, insbesondere wenn deutsch- und englischsprachige Versionen benutzt werden. Hier wurde das Datum als String im Format <<DD.MM.YYYY>> gespeichert. Danach erfolgt der Import:

```
PROC IMPORT
     DATAFILE = "Q:\myfolder\myfile.xlsx"
     DBMS = "XLSX"
     OUT = WORK.import_nightrun
     REPLACE;
     SHEET = "free_time";
     GETNAMES = YES;
RUN;

DATA WORK.free_time;
     SET WORK.import_nightrun;
     FORMAT date DATE9.;
     date = INPUT(COMPRESS(TRANWRD(date_end, ".", "")), DDMMYY8.);
     DROP date_end;
RUN;
```

Der Datastep übernimmt nach dem Import den Inhalt der Spalte „date_end“ aus der Exceldatei und transformiert den String zu einem numerischen Wert.

3.3 Zusammenführung der Tabellen

Es wird angenommen, dass eine SAS-Tabelle (hier risk_users in der Bibliothek risk_hlp) mit allen Benutzernamen existiert. An diese Tabelle werden die vorher erzeugten Dateien „spawned“ und „free_time“ angefügt.

```
PROC SQL NOPRINT;
     SELECT spa.server_id                INTO :server SEPARATED BY " "
     FROM RISK_HLP.risk_users      AS rus

     LEFT JOIN (SELECT name
                     , date
                FROM WORK.free_time
                WHERE date GE TODAY()
                GROUP BY name
                HAVING date = MAX(date)
                )                                AS frt
          ON rus.user_risk = frt.name

     LEFT JOIN (SELECT SUBSTR(processowner, 1, INDEX(processowner,
"@") - 1)      AS open_process
                     , serverid
                                    AS server_id
                FROM WORK.spawned
                )
                                    AS spa
          ON rus.user_risk = spa.open_process

     HAVING rus.user_risk NE frt.name
     AND NOT MISSING(open_process)

;QUIT;
```

Das vorstehende Query ist leicht erklärt. An die Tabelle mit allen Benutzern werden die Nutzer angespielt, die aktuell nicht berücksichtigt werden sollen. Die Auswahl erfolgt dabei über die where-Bedingung, während die having-Bedingung dafür sorgt, dass nur die längste Laufzeit verbleibt.
Im zweiten Join werden dann die Namen der aktiven User mit ihrem Benutzernamen übernommen und alle anderen Informationen in diesem String gelöscht bzw. nicht übernommen. Hierfür wird die Funktion <<SUBSTR>> mit der Funktion <<INDEX>> kombiniert und alles nach dem Zeichen „@“ verworfen.
Zuletzt werden alle User, die nicht in der importierten Excelliste stehen und ein Datum vorweisen, das größer als das gegenwärtige ist, in eine Macrovariable „server“ überführt. In dieser befinden sich die IDs, die den jeweiligen Sessions zugeordnet sind (siehe Abbildung 1).

3.4 Beendigung der aktiven Sessions

Die Vorarbeiten sind nun erledigt. Nun folgt ein Macro, das die aktiven Sessions über eine Schleife deaktiviert. Zusätzlich wird eine Datei erzeugt, die die aktiven Sessions in 2 unterschiedliche Gruppen, beendete und nicht-beendete Sessions, aufteilt. Abschließend verwendet eine html-Datei diese Informationen, welche per E-Mail an die gewünschten Empfänger versendet wird.

```
%MACRO kill_session;

     %IF %SYMEXIST(server) = 1 %THEN %DO;

          %GLOBAL i next_session;
          %DO i = 1 %TO %SYSFUNC(COUNTW(&server., ' '));

               %LET next_session = %SCAN(&server., &i., " ");

               DATA _NULL_;
                    macro_variable = "&next_session.";
                    quotes = '"';
                    kill_session = CAT(quotes, macro_variable,
quotes);
                    CALL SYMPUTX("kill_session", kill_session);
               RUN;

               PROC IOMOPERATE;
                    CONNECT HOST = &host
                    PORT = &port.
                    USER = &user.
                    PASS = &password.;
                         STOP spawned server
                         id = &kill_session.;
               QUIT;

          %END;

          PROC SQL;
               CREATE TABLE WORK.not_killed AS
               SELECT DISTINCT rus.user_risk
                                    LENGTH = 20
               FROM RISK_HLP.risk_users      AS rus

               LEFT JOIN (SELECT name
                              , date
                         FROM WORK.free_time
                         WHERE date GE TODAY()
                         GROUP BY name
                         HAVING date = MAX(date)
                         )                              AS frt
                    ON rus.user_risk = frt.name
```

```
               LEFT JOIN (SELECT SUBSTR(processowner, 1,
INDEX(processowner, "@") - 1)      AS open_process
                              , serverid
                                                  AS server_id
                         FROM WORK.spawned
                         )
                                                  AS spa
                    ON rus.user_risk = spa.open_process

               HAVING rus.user_risk = frt.name
          ;QUIT;

          PROC SQL;
               CREATE TABLE WORK.open_sessions AS
               SELECT DISTINCT SUBSTR(spa.processowner, 1,
INDEX(spa.processowner, '@') - 1)  AS User
     LENGTH = 20
                    , COUNT(spa.processowner)
                                                  AS open_sessions
                    , IFN(MISSING(nkl.user_risk), 1, 0)
                                                  AS user_kicked
               FROM WORK.spawned
                                                  AS spa

               LEFT JOIN WORK.not_killed
                                                  AS nkl
                    ON SUBSTR(spa.processowner, 1,
INDEX(spa.processowner, '@') - 1) = nkl.user_risk

               GROUP BY spa.processowner

          ;QUIT;

          ODS HTML FILE =
"E:\SASBATCHPROCESSES\KILL_ALL_SESSIONS\_LOG\killed_sessions.html"
STYLE = JOURNAL;

          PROC REPORT
               DATA = WORK.open_sessions;
          RUN;

          ODS HTML CLOSE;

          OPTIONS EMAILSYS = smtp
          EMAILHOST = EMAILHOST
          EMAILPORT = 123;

          FILENAME mailarr
          EMAIL "foltz@auxmoney.com"
               CC = ("user1@auxmoney.com" "user2@auxmoney.com")
          SUBJECT = "Killed Sessions"
          CONTENT_TYPE = "text/html";
```

```
          DATA _NULL_;
               FILE mailarr;
               INFILE
"E:\SASBATCHPROCESSES\KILL_ALL_SESSIONS\_LOG\killed_sessions.html";
               INPUT;
               PUT _INFILE_;
          RUN;

     %END;

     %ELSE %DO;

     OPTIONS EMAILSYS = smtp
     EMAILHOST = EMAILHOST
     EMAILPORT = 123;

     FILENAME mail
     EMAIL "foltz@auxmoney.com"
      CC = ("user1@auxmoney.com" "user2@auxmoney.com")
     SUBJECT = "Killed Sessions"
     CONTENT_TYPE = "text/html";

     DATA _NULL_;
          FILE mail;
          PUT "<body>";
          PUT "<p>None killed sessions today</p>";
          PUT "</body>";
     RUN;

     %END;

%MEND;

%kill_session;
```

Zu Beginn wird mit <<%SYMEXIST>> überprüft, ob überhaupt aktive Sessions bzw. eine Makrovariable existiert.[2] Ist dies nicht der Fall, wird das Script direkt beendet und eine E-Mail verschickt mit der Information, dass an diesem Tag keine Sessions beendet wurde („None killed sessions today“).
Existiert hingegen diese Makrovariable werden zunächst zwei weitere Makrovariablen „i“ und „next_session“ angelegt.
Bei i = 1 soll die Schleife starten und über alle Sessions iterieren. Die Anzahl wird mit Hilfe von <<COUNTW>> ermittelt. Im nächsten Schritt beinhaltet die Makrovariable „next_session“, die jeweilige Sessions-ID, welche mit „i“ und <<%SCAN>> festgestellt wird. Die Leerstelle zwischen den Anführungszeichen zeigt den Delimiter. Darauf erfolgt eine temporärer Datastep, der die Session-ID um doppelte Anführungszeichen er-

[2] Bitte beachten, die Funktion prüft auf eine Makrovariable per Definition, d.h. „server“ ist in diesem Fall richtig und nicht „&server.“.

gänzt und daraus eine finale Makrovariable erzeugt, die anschließend mit <<PROC IOMPERATE>> die jeweilige Session beendet.
Nachfolgendes ist dann optional. Die in diesem Prozess erzeugte Datei informiert, bei welchem User die Sessions beendet wurden und deren Anzahl. Abschließend bekommt man das Ergebnis per E-Mail zugesandt.