Alephino 4.1 Scripts

Inhalt

Allgemeines

Mithilfe der Alephino Scripts kann ein Datensatz geprüft oder manipuliert werden. Dabei können Fehlermeldungen erzeugt werden oder Felder hinzugefügt, gelöscht oder verändert werden.
Ein Script besteht aus einer Folge von Befehlen, die in einer ASCII-Textdatei erfasst sind. Eine Datei kann mehrere Scripts enthalten.

Die Standardversion von Alephino beinhaltet folgende Scripts:

- Ein Script **CHECK** zur Prüfung von Datensätzen in der Datei **mabscript.txt** bzw. **marcscript.txt**.
- Ein Script **COMPL** zur Vervollständigung von Datensätzen in der Datei mabscript.txt bzw. marcscript.txt.
- Scripts **CHECKxxx** zur Prüfung von Feldern in der Datei mabfield.txt bzw. marcfield.txt. xxx steht für die Feldnummer.

Diesen Scripts werden folgende Parameter übergeben:

- Parameter P1: Dateiname (z.B. B-TIT).
- Parameter P2: Benutzername.
- Parameter P3: Identnummer des Satzes.
- Parameter P4: Feldinhalt (nur bei den Scripts zur Feldprüfung).

Objekte

Folgende Objekte können im Script behandelt werden:

- Konstante Zeichenketten
- Mengen von konstanten Zeichenketten
- Zahlwerte
- Inhalte von Datenfelder
- Variable
- Parameter

Zeichenketten

Zeichenketten (auch Strings genannt) werden durch eine in Hochkommata eingeschlossene Zeichenfolge dargestellt.

Bsp.:

```
"Dies ist ein Text"
```

Mengen von Zeichenketten

Eine Menge von Zeichenketten (auch String-Arrays genannt) wird in geschweiften Klammern angegeben, Elemente durch Komma getrennt.

Bsp.:

```
{ "Text1" , "Text2" , "Text3" }
```

Zahlwerte

Es ist nur die Angabe ganzzahliger Werte (auch Integer genannt) zulässig.

Feldinhalte

Feldinhalte werden wie folgt angesprochen:

```
:feldname (Bsp.: :100)
```

Soll nur der Inhalt eines Feldes mit einem bestimmten Indikator angesprochen werden:

```
:feldname/"indikator" (Bsp.: :100/"a")
```

Bei wiederholbaren Feldern kann der Feldindex mitgegeben werden:

```
:feldname.index bzw. :feldname/"indikator".index (Bsp.: :100.2)
```

Unterfelder werden angesprochen durch :

```
feldname$unterfeldname
```

Für feldname sind die o.g. Kombinationen zulässig.

Bei wiederholbaren Unterfeldern kann der unterfeldname durch .index qualifiziert werden.

Die komplexeste Angabe für einen (Unter-)Feldinhalt lautet also:

```
:feldname/"indikator".index$unterfeldname.index
```

Bsp.:

```
:100/"i".2$u.1 bezeichnet das erste Unterfeld u des zweiten Hauptfeldes 100 mit dem Indikator
i.
```

Variable

Folgende Variable können definiert werden:

Variablen für Zeichenketten durch den Kennzeichner **STRING**.

Variablen für Mengen von Zeichenketten durch den Kennzeichner **STRSET**.

Variablen für Zahlwerte durch den Kennzeichner **INT**.

Bsp. Für eine Zeichenkettenvariablen-Definition:

```
STRING var
```

Variablen können Werte zugewiesen werden.

Bsp.:

```
var = "Inhalt der Variablen"
```

Es ist auch möglich, Variable bei der Definition zu initialisieren.

Bsp.:

```
STRING var = "Inhalt der Variablen"
```

Parameter

Parameter werden angesprochen durch **&P** gefolgt von der Parameternummer, d.h. **&P1** für den 1., **&P2** für den 2. Parameter usw.

Befehle

Die Befehle eines Scripts werden geklammert durch ein einleitendes

PROC scriptname

und ein abschliessendes

END PROC

Folgende Befehlsarten stehen zur Verfügung:

- Zuweisung von Objekten an Datenfelder oder Variable
- Ausgabe (Fehler-)Nachricht
- Aufruf von untergeordneten Scripts (= Prozeduren)
- Kontrollbefehle zur Ablauflogik (IF, WHILE, LOOP, CHOOSE)

Zusätzlich stehen zur Manipulation der Objekte zahlreiche eingebaute Funktionen zur Verfügung.

Zuweisungen

Eine Zuweisung hat die Form

Ziel = Quelle

Ziel einer Zuweisung kann ein Datenfeld des Satzes oder eine Variable sein.

Quelle einer Zuweisung kann sein:

- Ein Datenfeld Bsp.: **:100 = :200**
- Eine Variable Bsp.: **:100 = var**
- Eine konstante Zeichenkette Bsp.: **:100 = "Text"**
- Ein Parameter Bsp.: **:100 = &P2**
- Das Ergebnis des Aufrufs einer eingebauten Funktion

Das Löschen eines Datenfeldes wird durch die Zuweisung einer leeren Zeichenkette vorgenommen.

Bsp.: **:100 = ""**

Ausgabe (Fehler-)Nachricht

Eine Meldung kann vom Script an das aufrufende Programm (z.B. der Cat-Client) übergeben werden mit dem Befehl MESSAGE.

MESSAGE [:feldname] "nummer" [+ "Text"]...

Der optionale Parameter **:feldname** ordnet die Nachricht einem bestimmten Feld zu.

"nummer" ist die Nachrichtennummer, unter der der Text der Nachricht in der Datei **message.xxx** aufgeführt ist.

Handelt es sich um eine Nachricht mit variablen Teilen, können die variablen Teile mit dem +-Zeichen angeführt werden.

Bsp.:

```
MESSAGE :100 "1120" ordnet dem Feld 100 die Meldung mit der Nummer 1120 zu.
```

Aufruf von Prozeduren

Wiederkehrende Befehlsfolgen können in einem Unterscript (Prozedur) zusammengefasst werden.

Eine Prozedur wird wie das Script selbst durch

PROC procname

eingeleitet und durch

END PROC

abgeschlossen.

Der Aufruf einer Prozedur erfolgt durch die Angabe von

```
DO(procname)
```

Einer Prozedur können (beliebig viele) Parameter übergeben werden. Der Aufruf lautet dann

```
DO(procname(parm1, parm2, ...))
```

Innerhalb der Prozedur werden die Parameter dann mit **&P1**, **&P2**, ... angesprochen.

Kontrollbefehle

Die Kontrollbefehle dienen zur Abbildung der Ablauflogik von Befehlsfolgen.

Kontrollbefehle und in ihnen eingeschlossenen Befehlsfolgen können beliebig geschachtelt werden.

IF

Für die Ausführung von Befehlsfolgen, die von einer Bedingung abhängig sind, steht der IF-Befehl zur Verfügung.

IF condition THEN befehle [ELSE befehle] END IF

condition legt die Bedingung durch den Vergleich zweier Objekte fest.

Vergleichsoperatoren sind =, <, <=, >, >= und # (für ungleich).

Mehrere Bedingungen können mit den Operatoren **AND** und **OR** logisch verknüpft werden.

Trifft die Bedingung zu, wird die Befehlsfolge nach **THEN** ausgeführt.

Trifft die Bedingung nicht zu, wird die Befehlsfolge nach **ELSE** ausgeführt, wenn sie angegeben ist.

Es werden alle Befehle des zutreffenden Zweiges bis zum **END IF** ausgeführt.

Bsp.:

```
IF :100 = "YES" THEN :200 = "" END IF
```

```
Dieser Befehl löscht das Feld 200, wenn im Feld 100 YES steht.
```

WHILE

Für Befehlsschleifen stehen die Befehle **WHILE** und **LOOP** zur Verfügung.

Der WHILE-Befehl wiederholt eine Befehlsfolge, solange eine Bedingung erfüllt ist.

WHILE condition befehle END WHILE

condition legt die Bedingung durch den Vergleich zweier Objekte fest.

Vergleichsoperatoren sind =, <, <=, >, >= und # (für ungleich).

Mehrere Bedingungen können mit den Operatoren **AND** und **OR** logisch verknüpft werden.

Solange die Bedingung erfüllt ist, wird die Befehlsfolge bis zum **END WHILE** ausgeführt.

LOOP

Der LOOP-Befehl arbeitet wie der WHILE-Befehl, prüft die Bedingung jedoch erst am Ende der Befehlsfolge, d.h. die Befehle werden auf jeden Fall einmal ausgeführt.

LOOP befehle UNTIL condition

CHOOSE

Für den Fall, dass abhängig vom Inhalt eines Objektes verschiedene alternative Befehlsfolgen durchlaufen werden sollen, steht der CHOOSE-Befehl zur Verfügung.

CHOOSE quelle CASE condition befehle [CASE condition befehle]... END CHOOSE

quelle gibt das Objekt an, dessen Inhalt verglichen werden soll.

condition gibt die jeweilige Vergleichsbedingung an. Trifft sie zu, werden die folgenden Befehle bis zum nächsten **CASE** oder bis zum **END CHOOSE** ausgeführt.

```
Bsp.:
CHOOSE :100
CASE = "A"
  (Befehle, falls Feld 100 den Inhalt A hat)
CASE = "B"
  (Befehle, falls Feld 100 den Inhalt B hat)
CASE = "C"
  (Befehle, falls Feld 100 den Inhalt C hat)
END CHOOSE
```

Eingebaute Funktionen

Die eingebauten Funktionen können benutzt werden, um Feldinhalte zu modifizieren. Parameter der Funktionen können Datenfelder, Konstanten, Variablen oder wiederum Funktionsaufrufe sein.

BEGSTR(s, t)	wenn t ein String: wenn der String s mit der Zeichenfolge t beginnt, wird t zurückgeliefert wenn t ein String-Array: wenn der String s mit einer der Zeichenfolgen aus t beginnt, wird diese zurückgeliefert
CHKFLD(s, x)	prüft den Inhalt von String s gemäß dem Inhalt von String x: wenn x = "ISBN" Formale Prüfung auf ISBN wenn x = "ISMN" Formale Prüfung auf ISMN wenn x = "ISSN" Formale Prüfung auf ISSN wenn x = "DATE" Formale Prüfung auf Datum der Form YYYYMMDD Bei erfolgreicher Prüfung wird ein Leerstring zurückgeliefert, sonst ein Fehlercode
CONCAT(s, t)	konkateniert die Strings s und t
DELSTR(s, t)	wenn t ein String: löscht die Zeichenfolge t aus dem String s wenn t ein String-Array: löscht die Zeichenfolgen von t aus dem String s
DELSTR(s, t, u)	wenn t und u Strings: löscht die Zeichenfolge aus dem String s, die mit der Zeichenfolge t beginnt und mit der Zeichenfolge u endet wenn t und u String-Array: löscht die Zeichenfolgen aus dem String s, die mit einer Zeichenfolge aus t beginnt und mit der entsprechenden Zeichenfolge aus u endet (jeweils inklusive der Zeichenfolgen t und u)
ENDSTR(s, t)	wenn t ein String: wenn der String s mit der Zeichenfolge t endet, wird t zurückgeliefert wenn t ein String-Array: wenn der String s mit einer der Zeichenfolgen aus t endet, wird diese zurückgeliefert
INITCAP(s)	setzt 1. Zeichen von s in Groß-, den Rest in Kleinbuchstaben um (gilt für die Buchstaben des Alphabets)
INSTR(s, t)	wenn t ein String: liefert die Zeichenfolge t zurück, wenn sie im String s enthalten ist wenn t ein String-Array: liefert die Zeichenfolge aus t zurück, die im String s enthalten ist
INSTR(s, t, u)	wenn t und u Strings: liefert die Zeichenfolge aus s zurück, die zwischen den Zeichenfolgen t und u steht wenn t und u String-Array: liefert die Zeichenfolge aus s zurück, die zwischen einer Zeichenfolge aus t und der entsprechenden Zeichenfolge aus u steht (jeweils exklusive der Zeichenfolgen t und u)
LEFT(s, t)	wenn t ein String: liefert den Teil des String s, der links von der Zeichenfolge t steht wenn t ein String-Array: liefert den Teil des String s, der links von einer der Zeichenfolgen aus t steht
LOWER(s)	setzt s in Kleinbuchstaben um (gilt für die Buchstaben des Alphabets)
LPAD(s, n, t)	füllt den String s linksbündig mit der Zeichenfolge t bis zur Länge n auf
LTRIM(s)	eliminiert führende Blanks im String s
LTRIM(s, t)	wenn t ein String: eliminiert die führenden Zeichenfolgen t

	wenn t ein String-Array: eliminiert alle führenden Zeichenfolgen aus t
NL	liefert eine neue Zeile (Newline)
REPLACE(s, t, u)	wenn t und u Strings: ersetzt die Zeichenfolge t im String s durch die Zeichenfolge u
	wenn t und u String-Array: ersetzt die Zeichenfolgen aus t im String s durch die entsprechenden Zeichenfolgen aus u
RIGHT(s, t)	wenn t ein String: liefert den Teil des String s, der rechts von der Zeichenfolge t steht
	wenn t ein String-Array: liefert den Teil des String s, der rechts von einer der Zeichenfolgen aus t steht
RPAD(s, n, t)	füllt den String s rechtsbündig mit der Zeichenfolge t bis zur Länge n auf
RTRIM(s)	eliminiert abschließende Blanks im String s
RTRIM(s, t)	wenn t ein String: eliminiert die abschließenden Zeichenfolgen t
	wenn t ein String-Array: eliminiert alle abschließenden Zeichenfolgen aus t
SUBSTR(s, n)	liefert den String s ab Stelle n
SUBSTR(s, n, m)	liefert den String s ab Stelle n in Länge m
UPPER(s)	setzt s in Großbuchstaben um (gilt für die Buchstaben des Alphabets)
ADD(i, j)	für Integer-Werte : liefert i + j zurück
SUB(i, j)	für Integer-Werte : liefert i - j zurück
MUL(i, j)	für Integer-Werte : liefert i * j zurück
DIV(i, j)	für Integer-Werte : liefert i / j zurück, wenn j ungleich 0
DATE(f)	liefert das aktuelle Datum im Format f, wobei für f zulässig sind: DD.MM.YYYY DD.MM.YY MM/DD/YYYY MM/DD/YY YYMMDD YYYYMMDD YYDDD YYYYDDD
TIME(f)	liefert die aktuelle Uhrzeit im Format f, wobei für f zulässig sind: HH.MM.SS HH.MM HH:MM:SS

Beispiele:

```
STRING s = "Dieses ist ein Text"
```

Aufruf	**Ergebnis**
BEGSTR(s, "Dies")	"Dies"

CHKFLD("1-111-11111-1", "ISBN")	""
CONCAT("Auch ", s)	"Auch Dieses ist ein Text"
DELSTR(s, "ein ")	"Dieses ist Text"
ENDSTR(s, "Text")	"Text"
INITCAP("abcdefg")	"Abcdefg"
INSTR(s, "ist")	"ist"
INSTR(s, "Dies", "Text")	"es ist ein "
LEFT(s, "es")	"Dies"
LOWER(s)	"dieses ist ein text"
LPAD(s, 25, "xyz")	"xyzxyzDieses ist ein Text"
LTRIM(" xyz")	"xyz"
REPLACE(s, "ein", "kein")	"Dieses ist kein Text"
RIGHT(s, "es")	" ist ein Text"
RPAD(s, 25, "xyz")	"Dieses ist ein Textxyzxyz"

RTRIM(s, "xyz ")	"xyz"
SUBSTR(s, 12)	"ein Text"
SUBSTR(s, 12, 3)	"ein"
UPPER(s)	"DIESES IST EIN TEXT"
DATE("DD.MM.YYYY")	"26.09.2001"
TIME("HH:MM:SS")	"10:39:55"
ADD(5, 2)	7
SUB(5, 2)	3
MULT(5, 2)	10
DIV(5, 2)	2

Kommentare

Kommentare werden durch // eingeleitet und gelten bis zum Zeilenende.