



Handbuch

Berichte im RTF/Word-Layout

© 2001-2023 by delight software gmbh

Berichte im RTF/Word-Layout 2023

Basierend auf delight base 11.x

© 2001-2023 by *delight software gmbh*



Berichte im RTF/Word-Layout 2023

© 2001-2023 by delight software gmbh

Es gelten die Allgemeinen Geschäftsbedingungen und die Allgemeinen Lizenzvereinbarungen der
delight software gmbh

Inhaltsverzeichnis

Kapitel 1 - RTF Layout-Datei	2
1.1 Aufbau	3
1.1.1 Scans	3
1.1.1.1 Einfacher Scan	3
1.1.1.2 Erweiterter Scan	3
1.1.1.3 Scanfunktionen	4
1.1.1.3.1 Spezielle Scan Funktionen	4
1.1.2 Variablen	5
1.1.3 Konstanten	5
1.1.4 Bedingungen	5
1.1.5 Funktionen	5
1.1.5.1 Funktionen mit Rückgabewert	6
1.1.5.1.1 FIMG	6
1.1.5.1.2 FORMATF	6
1.1.5.1.3 STRREPLACE	8
1.1.5.1.4 RESTOREVAL	8
1.1.5.1.5 PRNEQINT	8
1.1.5.1.6 PRNNEQINT	8
1.1.5.1.7 PRNNEQ	8
1.1.5.1.8 PRNEQ	8
1.1.5.1.9 DIV	9
1.1.5.1.10 MUL	9
1.1.5.1.11 FORMATDATE	9
1.1.5.1.12 PERCENT	10
1.1.5.1.13 INC	10
1.1.5.1.14 DEC	11
1.1.5.1.15 PRNNEG	11
1.1.5.1.16 PRNPOS	11
1.1.5.1.17 FLNK	11
1.1.5.1.18 EQ	11
1.1.5.1.19 NEQ	12
1.1.5.1.20 MONTH	12
1.1.5.1.21 TRANSLATE	12
1.1.5.1.22 FRTF	12
1.1.5.2 Funktionen ohne Rückgabewert	12
1.1.5.2.1 SETV	12
1.1.5.2.2 VAR	13
1.1.5.2.3 STOREVAL	13
1.1.5.2.4 SETVAR	13
1.1.5.2.5 CTS	13
1.1.5.2.6 CTN	14
1.1.5.2.7 SUM	14
1.2 Beispiel 1	14
1.3 Beispiel 2	15

RTF Layout-Datei

1

1 RTF Layout-Datei

Die Designdefinition der Berichte wird in einer RTF Datei beschrieben, einer sogenannten Layout-Datei.

Es ist eine Datei im RTF Format, die mit jedem gängigen RTF Editor (z.B. Microsoft Word) erstellt und bearbeitet werden kann.

In einer RTF Layout-Datei kann mit einer speziellen Syntax definiert werden, wie die Daten in einem Bericht dargestellt werden.

1.1 Aufbau

Wie man eine Vorlage und das zugehörige Layout erstellt wird in Berichte beschrieben.

siehe auch:

Vorlagen

1.1.1 Scans

In einem Scan werden alle Datensätze einer Abfrage durchlaufen und die Daten der Abfrage ausgegeben.

Scans können verschachtelt werden. Auf diese Weise können Master-Detail Reports realisiert werden.

1.1.1.1 Einfacher Scan

In einem einfachen Scan wird der Abschnitt zwischen den Linien *Scan* und *EndScan* für jeden Datensatz in der Abfrage repetiert.

Code

```
\Scan(Abfrage) [, while(FUNCTION(...))] [,page] [,noeof] [,function1,...,functionN]\
```

```
\EndScan [,function1,...,functionN]\
```

Beispiel

```
\scan(a)\  
Hallo Welt  
\EndScan\
```

Hier wird für jeden Datensatz in der Abfrage *a* **Hallo Welt** auf dem Report platziert. Wenn die Abfrage also 10 Datensätze beinhaltet, wird 10-mal **Hallo Welt** auf dem Report platziert.

1.1.1.2 Erweiterter Scan

In einem erweiterten Scan wird der Abschnitt zwischen den Linien *ScanEntry* und *ScanFooter* für jeden Datensatz in der Abfrage repetiert.

Beim Eintreten in den Scan wird einmalig der Abschnitt zwischen *Scan* und *ScanEntry* platziert.

Nachdem alle Datensätze durchlaufen sind wird einmalig der Abschnitt zwischen *ScanFooter* und *EndScan* platziert.

Diese Funktionalität kann z.B. bei Tabellen angewendet werden:

Zwischen *Scan* und *ScanEntry* wird der Tabellenkopf platziert.

Zwischen *ScanEntry* und *ScanFooter* werden die Daten der Tabelle platziert.

Zwischen *Scanfooter* und *EndScan* wird die Zusammenfassung platziert.

ScanEntry und *ScanFooter* müssen nicht zwingend verwendet werden. Es ist möglich beide, oder nur eines der beiden zu verwenden.

Erweiterter Scan

\Scan(Abfrage) [, while(FUNCTION(...))] [,page] [,noeof] [,function1,...,functionN]\

\Scanentry [,function1,...,functionN]\

\Scanfooter [,function1,...,functionN]\

\Endscan [,function1,...,functionN]\

Beispiel

```
\scan(a)\
Titel
\ScanEntry\
Hallo Welt
\ScanFooter\
Zusammenfassung
\EndScan\
```

Hier wird für jeden Datensatz in der Abfrage **a** **Hallo Welt** auf dem Report platziert. Wenn die Abfrage also 10 Datensätze beinhaltet, wird 10-mal **Hallo Welt** auf dem Report platziert. Vor das erste **Hallo Welt** wird **Titel** platziert und nach dem letzten wird **Zusammenfassung** platziert.

1.1.1.3 Scanfunktionen

In jedem Scanelement können beliebig viele Funktionen ausgeführt werden.

Die Funktionen werden durch Kommas voneinander getrennt.

Diese Funktionen können z.B. verwendet werden, um Beträge zusammenzuzählen. Das Resultat kann dann, z.B. in einem ScanFooter, als Total ausgegeben werden.

Die Verfügbaren Funktionen finden sich im Kapitel [Funktionen](#) beschrieben.

Beispiel

```
\Scan(a, FUNCTION(PARAMETER1, PARAMETER2)\
```

1.1.1.3.1 Spezielle Scan Funktionen

Es gibt spezielle Funktionen die nur in einem Scanelement verwendet werden können.

Scanelementfunktionen

while	Der Scan läuft vom Ersten bis zum Letzten Datensatz der Abfrage solange die Funktion in while <i>true</i> ist. Liefert die Funktion in while <i>false</i> zurück, wird der Scan abgebrochen und verlassen.
page	Fügt bei jedem Durchlauf eine neue Seite ein, ausgenommen beim Ersten.
noeof	Der Scanblock wird nur platziert wenn die Abfrage Daten enthält. Wird meistens bei Master-Detail Reports für den Detail-Scan verwendet.

1.1.2 Variablen

Auf einem Report können Variablen verwendet werden, um temporäre Berechnungen durchzuführen oder um gewisse Sachen zwischenspeichern.

Eine Reportvariable kann auf einfache Weise über \VARIABLENAME\ auf dem Report platziert werden. Die meisten Funktionen verwenden Variablen um Werte zu speichern. Viele der Funktionen erzeugen bei Bedarf eine neue Variable wenn diese noch nicht existiert.

Beispiel

```
\CTN('1', a_var)\
```

Wenn beim Ausführen dieser Funktion die Variable `a_var` noch nicht existiert, wird die Variable neu angelegt und das Ergebnis in diese gespeichert.

1.1.3 Konstanten

Einer Funktion können nicht nur Variablen und Datenbankfelder einer Abfrage übergeben werden, sondern es können auch konstante Werte übergeben werden.

Ein konstanter Wert muss immer in " oder ~~ Zeichen eingeschlossen werden.

Beispiel

```
\CTN('1', a_var)\  
\CTN(~1~, a_var)\
```

Hier wird der Funktion CTN als erster Parameter der konstante Wert 1 übergeben.

1.1.4 Bedingungen

Bedingungen können mit IF-ELSE-ENDIF realisiert werden.

Syntax:

```
\If(<boolean value>)\  
.....  
\else\  
.....  
\endif
```

Hinweis: \else\ ist optional.

1.1.5 Funktionen

Mit den Funktionen können Berechnungen durchgeführt werden, und Ausgaben formatiert werden.

Jede Funktion benötigt gewisse Parameter. Bei manchen Funktionen gibt es optionale Parameter die nur bei Bedarf übergeben werden müssen.

Die Parameter können Reportvariablen, Konstanten und andere Funktionen sein.

Der Rückgabewert einer Funktion kann auf einfache Weise über \FUNCTION(a:feld, a:feld2)\ auf dem Report platziert werden.

Wenn die Funktionen keinen Rückgabewert liefert wird eine leerer String platziert.

Allgemeiner Syntax

FUNKTION(a:feldname, b:feldname)

FUNKTION('konstante', a:feldname)

FUNKTION(report_variable, 'konstante')

FUNKTION(FUNKTION2(a:feldname, b:feldname), FUNKTION(report_variable, 'konstante'))

1.1.5.1 Funktionen mit Rückgabewert

Funktionen mit einem Rückgabewert werden normalerweise benötigt um Daten formatiert auf dem Report zu platzieren.

1.1.5.1.1 FIMG

Importiert eine Grafikdatei und gibt sie im korrekten RTF Format zurück, damit Sie im Report platziert werden kann.

Syntax

FIMG(FileName)

Importiert die Grafikdatei *FileName* und gibt sie im korrekten RTF Format zurück.

Beispiel

\FIMG('.\custom\logo.bmp')\

Importiert die Grafikdatei *.\custom\logo.bmp* und platziert sie auf dem Report.

Hinweis:

Diese Funktion ist bei Reports die mit dem ML2-Server-Module erzeugt werden nicht verfügbar.

1.1.5.1.2 FORMATF

Platziert eine Kommazahl formatiert auf dem Report.

Syntax

FORMATF(Wert, Format)

Formatiert den Wert in *Wert* nach den Vorgaben in *Format* und gibt das Ergebnis zurück.

Format funktionen

Bezeichner	Beschreibung
0	Platzhalter für eine Ziffer. Enthält der zu formatierende Wert an der Position eine Ziffer, an der im Format-String '0' steht, wird diese in den Ausgabe-String kopiert. Andernfalls wird das Zeichen '0' an dieser Position im Ausgabe-String gespeichert.

#	Platzhalter für eine Ziffer. Enthält der zu formatierende Wert an der Position eine Ziffer, an der im Format-String '#' steht, wird diese in den Ausgabe-String kopiert. Andernfalls wird an dieser Position kein Zeichen im Ausgabe-String gespeichert.
.	Dezimaltrennzeichen. Das erste '.'-Zeichen im Format-String bestimmt die Position des Dezimaltrennzeichens im formatierten Wert. Alle weiteren dieser Zeichen werden ignoriert. Das tatsächlich im Ausgabe-String verwendete Zeichen wird mit der globalen Variable <code>DecimalSeparator</code> festgelegt. Die Variable erhält als Standardwert den entsprechenden Eintrag in der Registerkarte Zahlen im Modul Ländereinstellungen der Windows-Systemsteuerung.
,	Tausendertrennzeichen. Enthält der Format-String ein oder mehrere ','-Zeichen, werden in den Ausgabe-String links des Dezimaltrennzeichens nach jeder Gruppe von drei Ziffern Ausendertrennzeichen eingefügt. Die Position und Anzahl der Trennzeichen im Format-String wirkt sich nicht auf die Ausgabe aus. Sie geben nur an, dass Trennzeichen eingefügt werden sollen. Das tatsächlich im Ausgabe-String verwendete Zeichen wird mit der globalen Variable <code>ThousandSeparator</code> festgelegt. Die Variable erhält als Standardwert den entsprechenden Eintrag in der Registerkarte Zahlen des Moduls Ländereinstellungen der Windows-Systemsteuerung.
E+	Wissenschaftliche Schreibweise. Sind die Zeichen 'E+', 'E-', 'e+' oder 'e-' im Format-String enthalten, wird die Zahl in der wissenschaftlichen Schreibweise formatiert. Bis zu vier '0'-Zeichen können direkt nach 'E+', 'E-', 'e+' oder 'e-' angegeben werden, um die minimale Anzahl der Stellen im Exponenten festzulegen. Bei den Formaten 'E+' und 'e+' wird für positive Exponenten ein Pluszeichen und für negative Exponenten ein Minuszeichen in den String eingefügt. Bei den Formaten 'E-' und 'e-' wird lediglich für negative Exponenten ein Vorzeichen ausgegeben.
'xx'/'xx"	In halbe oder ganze Anführungszeichen eingeschlossene Zeichen wirken sich nicht auf die Formatierung aus und werden wie eingegeben angezeigt.
;	Trennt Abschnitte für positive, negative und Nullwerte im Format-String.

Die Zeichen zwischen dem äussersten linken '0' vor dem Dezimaltrennzeichen und dem äussersten rechten '0' nach dem Dezimaltrennzeichen werden immer im Ausgabe-String angezeigt.

Die zu formatierende Zahl wird immer auf so viele Dezimalstellen gerundet, wie Ziffernplatzhalter ('0' oder '#') rechts des Dezimaltrennzeichens vorhanden sind. Enthält der Format-String kein Dezimaltrennzeichen, wird der Wert auf die nächste ganze Zahl gerundet.

Hat die zu formatierende Zahl mehr Vorkommastellen, als Ziffernplatzhalter links des '.' im Format-String vorhanden sind, werden die zusätzlichen Stellen vor dem ersten Platzhalter ausgegeben.

Damit für positive, negative und Nullwerte unterschiedliche Formate festgelegt werden können, kann der Format-String zwischen einem und drei durch Semikolons getrennte Abschnitte enthalten.

Ein Abschnitt: Der Format-String wird für alle Werte verwendet.

Zwei Abschnitte: Der erste Abschnitt wird für positive und Nullwerte, der zweite Abschnitt für negative Werte verwendet.

Drei Abschnitte: Der erste Abschnitt wird für positive, der zweite für negative, und der dritte Abschnitt für Nullwerte verwendet.

Wenn der Abschnitt für negative oder Nullwerte keine Angaben enthält, wird statt dessen der Abschnitt für positive Werte verwendet.

Ist der Abschnitt für positive Werte oder der gesamte Format-String leer, wird die Zahl im allgemeinen Gleitkommaformat mit 15 signifikanten Stellen formatiert. Das allgemeine Gleitkommaformat wird auch verwendet, wenn mehr als 18 Vorkommastellen vorhanden sind und im Format-String nicht die wissenschaftliche Schreibweise angegeben wird.

1.1.5.1.3 STRREPLACE

Ersetzt alle Teilstrings in einem String durch einen anderen.

Syntax

STRREPLACE(Source, OldStr, NewStr)

Ersetzt alle Vorkommen von *OldStr* in *Source* durch *NewStr* und gibt das Ergebnis zurück.

1.1.5.1.4 RESTOREVAL

Stellt einen Wert der mit [STOREVAL](#) gespeichert wurde, wieder her.

Syntax

RESTOREVAL(Bezeichner [, Destination])

Wenn der Parameter *Destination* nicht angegeben wird, gibt die Funktion den gespeicherten Wert von *Bezeichner* zurück.

Wird der Parameter *Destination* angegeben, hat die Funktion keinen Rückgabewert sondern speichert den gespeicherten Wert von *Bezeichner* in *Destination*.

1.1.5.1.5 PRNEQINT

Hat die gleiche Funktionalität wie [PRNEQ](#), arbeitet aber mit Zahlenvariablen und gibt bei Ungleichheit den Wert 0 zurück.

1.1.5.1.6 PRNNEQINT

Hat die gleiche Funktionalität wie [PRNNEQ](#), arbeitet aber mit Zahlenvariablen und gibt bei Gleichheit den Wert 0 zurück.

1.1.5.1.7 PRNNEQ

Gibt nur einen Rückgabewert zurück wenn zwei Werte ungleich sind.

Syntax

PRNNEQ(Wert1, Wert2, Source)

Gibt als Rückgabewert *Source* zurück wenn *Wert1* ungleich *Wert2* ist.

1.1.5.1.8 PRNEQ

Gibt nur einen Rückgabewert zurück wenn zwei Werte übereinstimmen.

Syntax

PRNEQ(Wert1, Wert2, Source)

Gibt als Rückgabewert *Source* zurück wenn *Wert1* gleich *Wert2* ist.

1.1.5.1.9 DIV

Dividiert zwei Werte und gibt das Ergebnis zurück. Optional kann das Ergebnis in einer Variable gespeichert werden.

Syntax

DIV(Wert1, Wert2 [, Destination])

Dividiert *Wert1* durch *Wert2* und gibt das Ergebnis zurück. Wenn der optionale Parameter *Destination* angegeben wurde, wird das Ergebnis zusätzlich in *Destination* gespeichert.

1.1.5.1.10 MUL

Hat die gleiche Funktionalität wie [DIV](#), mit dem Unterschied das mit MUL die Werte multipliziert werden.

1.1.5.1.11 FORMATDATE

Gibt ein Datum- oder Zeitwertwert formatiert zurück.

Syntax

FORMATDATE(Source [, Format])

Gibt den in *Source* übergebenen Datum- oder Zeitwert nach den Vorgaben in *Format* formatiert zurück.

Format funktionen

Bezeichner	Anzeige
d	Zeigt den Tag als Zahl ohne führende Null an (1-31).
dd	Zeigt den Tag als Zahl mit führender Null an (01-31).
ddd	Zeigt den Wochentag als Abkürzung (Son-Sam) an.
dddd	Zeigt den ausgeschriebenen Wochentag (Sonntag-Samstag) an.
e	Zeigt das Jahr des aktuellen Zeitalters als eine Zahl ohne führende Null an (nur bei japanischen, koreanischen und taiwanesischen Ländereinstellungen).
ee	Zeigt das Jahr des aktuellen Zeitalters als eine Zahl mit führender Null an (nur bei japanischen, koreanischen und taiwanesischen Ländereinstellungen).
g	Zeigt das aktuelle Zeitalter als Abkürzung an (nur bei japanischen und taiwanesischen Ländereinstellungen).
gg	Zeigt das ausgeschriebene Zeitalter an (nur bei japanischen und taiwanesischen Ländereinstellungen).
m	Zeigt den Monat als Zahl ohne führende Null an (1-12). Wenn der Bezeichner m unmittelbar hinter dem Bezeichner h oder hh steht, wird anstelle des Monats die Minute angezeigt.
mm	Zeigt den Monat als Zahl mit führender Null an (01-12). Wenn der Bezeichner mm unmittelbar hinter dem Bezeichner h oder hh steht, wird anstelle des Monats die Minute angezeigt.
mmm	Zeigt den Monatsnamen als Abkürzung (Jan-Dez) an.
mmmm	Zeigt den ausgeschriebenen Monatsnamen (Januar-Dezember) an.
yy	Zeigt das Jahr als zweistellige Zahl an (00-99).
yyyy	Zeigt das Jahr als vierstellige Zahl an (0000-9999).
h	Zeigt die Stunde ohne führende Null an (0-23).
hh	Zeigt die Stunde mit führender Null an (00-23).
n	Zeigt die Minute ohne führende Null an (0-59).
nn	Zeigt die Minute mit führender Null an (00-59).
s	Zeigt die Sekunde ohne führende Null an (0-59).

ss	Zeigt die Sekunde mit führender Null an (00-59).
z	Zeigt die Millisekunde ohne führende Null an (0-999).
zzz	Zeigt die Millisekunde mit führender Null an (000-999).
t	Zeigt die Uhrzeit wie in den Ländereinstellungen des Betriebssystems definierte an.
tt	Zeigt die Uhrzeit wie in den Ländereinstellungen des Betriebssystems definierte an.
am/pm	Verwendet die 12-Stunden-Zeitanzeige für den vorhergehenden Bezeichner h oder hh und zeigt alle Stunden vor Mittag mit dem String 'am' und alle Stunden nach Mittag mit dem String 'pm' an. Für den Bezeichner am/pm können Kleinbuchstaben, Großbuchstaben und jede Kombination davon angegeben werden.
a/p	Verwendet die 12-Stunden-Zeitanzeige für den vorhergehenden Bezeichner h oder hh und zeigt alle Stunden vor Mittag mit dem Zeichen 'a' und alle Stunden nach Mittag mit dem Zeichen 'p' an. Für den Bezeichner a/p können Kleinbuchstaben, Großbuchstaben und jede Kombination davon angegeben werden.
/	Zeigt als Datumstrennzeichen das in den Ländereinstellungen des Betriebssystems definierte Zeichen an.
:	Zeigt als Uhrzeittrennzeichen das in den Ländereinstellungen des Betriebssystems definierte Zeichen an.
'xx'/'xx'	Zeichen, die in einfache oder doppelte Anführungszeichen eingeschlossen sind, werden ohne spezielle Formatierung übernommen.

1.1.5.1.12 PERCENT

Rechnet den prozentualen Anteil eines Wertes aus und gibt das Ergebnis zurück. Optional kann das Ergebnis in einer Variable gespeichert werden.

Syntax

PERCENT(Source, Percent [, Destination])

Rechnet den prozentualen Anteil *Percent* von *Source* aus und gibt das Ergebnis zurück. Wenn der optionale Parameter *Destination* angegeben wurde, wird das Ergebnis zusätzlich in *Destination* gespeichert.

Beispiel

```
\PERCENT('1000', '10')
```

Platziert den Wert 100 auf dem Report.

1.1.5.1.13 INC

Addiert zwei Werte und gibt das Ergebnis zurück. Optional kann das Ergebnis in einer Variable gespeichert werden.

Syntax

INC(Wert1, Wert2 [, Destination])

Addiert *Wert1* zu *Wert2* und gibt das Ergebnis zurück. Wenn der optionale Parameter *Destination* angegeben wurde, wird das Ergebnis zusätzlich in *Destination* gespeichert.

1.1.5.1.14 DEC

Hat die gleiche Funktionalität wie [INC](#), mit dem Unterschied das mit DEC die Werte subtrahiert werden.

1.1.5.1.15 PRNNEG

Gibt nur einen Rückgabewert zurück wenn der Wert kleiner als 0 ist.

Syntax

PRNNEG(Source [, Format])

Gibt als Rückgabewert *Source* zurück wenn *Source* kleiner als 0 ist.
Wenn der optionale Parameter *Format* angegeben wurde, wird das Ergebnis nach den Vorgaben in *Format* formatiert.

siehe auch:

[Format funktionen](#)

1.1.5.1.16 PRNPOS

Hat die gleiche Funktionalität wie [PRNNEG](#), mit dem Unterschied das mit PRNPOS nur ein Wert zurückgegeben wird wenn der Wert grösser als 0 ist.

1.1.5.1.17 FLNK

Gibt einen Wert als Hyperlink zurück.

Syntax

FLNK(Wert)

Gibt den in *Wert* angegebenen Hyperlink zurück.

Beispiel

```
\FLNK('http://www.delight.ch')\
```

Platziert den Link www.delight.ch auf dem Report.

1.1.5.1.18 EQ

Vergleicht zwei Werte und gibt Wahr oder Falsch zurück.

Syntax

EQ(Wert1, Wert2)

Beispiel

```
\if(EQ(a:feld1, '1'))\
Feld1 hat den Wert 1
\else\
Feld1 hat einen Wert ungleich 1
\endif
```

1.1.5.1.19 NEQ

Hat die gleiche Funktionalität wie [EQ](#), mit dem Unterschied das mit NEQ auf ungleich (<>) abgefragt wird.

1.1.5.1.20 MONTH

Gibt den übergebenen Monat als String zurück.

Syntax

MONTH(AMonth [, Format])

Gibt den (als Nummer in AMonth übergeben) Monat als String mit dem in Format angegebenen String zurück.

Beispiel

\MONTH(3)\

Platziert den String *März* auf dem Report.

1.1.5.1.21 TRANSLATE

Gibt den Übersetzten Text des übergebenen Übersetzungsschlüssels zurück.

Syntax

TRANSLATE(AKey)

Beispiel

\TRANSLATE('IDS_Name')\

Platziert den String der unter IDS_Name in der Übersetzung angegeben wurde auf dem Report.

1.1.5.1.22 FRTF

Fügt RTF-Code direkt in die Ausgabedatei ein.

Syntax

FRTF(RTFCode)

1.1.5.2 Funktionen ohne Rückgabewert

Funktionen ohne Rückgabewert werden normalerweise in einem Scapelement verwendet.

1.1.5.2.1 SETV

Mit SETV kann einer Report Variable explizit ein Wert zugewiesen werden.

Syntax

SETV(VariableXY, '0')

Setzt die Report Variable *VariableXY* auf den Wert 0.

1.1.5.2.2 VAR

Mit VAR können Report Variablen explizit erstellt werden.

Syntax

VAR(Var1[, Var2, .., VarN])

Es können beliebig viele Variablen erzeugt werden, die einzelnen Variablen werden durch Kommas getrennt. Wenn die Variable schon existiert, hat die Funktion keinen Einfluss auf die Variable.

1.1.5.2.3 STOREVAL

Mit STOREVAL können Werte unter einem Bezeichner zwischengespeichert werden.

Syntax

STOREVAL(Bezeichner, Wert)

Speichert den in *Wert* übergebenen Wert unter dem Bezeichner *Bezeichner*. Der Wert kann später mit [RESTOREVAL](#) wieder hergestellt werden.

1.1.5.2.4 SETVAR

Mit SETVAR kann explizit eine Reportvariable erzeugt werden.

Syntax

SETVAR(a:feldname)

Erzeugt eine neue Reportvariable mit dem Namen *feldname* und weist ihr den aktuellen Wert von *a:feldname* zu. Wenn bereits eine Variable mit dem selben Namen existiert, wird diese auf den neuen Wert gesetzt.

1.1.5.2.5 CTS

Die Funktion CTS inkrementiert eine Variable wenn ein String nicht leer ist. Ein String der nur aus Leerzeichen besteht wird als leer interpretiert.

Diese Funktion sollte nur in einem Scanelement verwendet werden.

Syntax

CTS(a:feld, report_variable [, NORESET])

Wenn der Wert in *a:feld* einen nicht leeren String enthält, wird die Variable *report_variable* inkrementiert. Wird der optionale Parameter NORESET angegeben, wird die Variable *report_variable* beim ersten Eintritt in den Scan nicht mit 0 initialisiert.

Beispiel

```
\scan(a)\
\a:number\ - \a:svalue\
\endscan, ctn(a:feldname, c_value)\
Anzahl nicht leerer Strings in Abfrage a: \c_value\
```



```
\scan(b)\
\b:number\ - \b:svalue\
\endscan, ctn(b:feldname, c_value, noreset)\
```

Anzahl nicht leerer Strings in Abfrage a und b: \c_value\

1.1.5.2.6 CTN

Hat die selbe Funktion wie [CTS](#) mit dem Unterschied, dass die Variable bei Werten ungleich 0 inkrementiert wird.

Diese Funktion sollte nur in einem Scaelement verwendet werden.

1.1.5.2.7 SUM

Die Funktion SUM summiert Werte und speichert das Ergebnis in eine Variable.

Diese Funktion sollte nur in einem Scaelement verwendet werden.

Syntax


SUM(a:amount, amount_sum [, NORESET])

Der Wert in *a:amount* wird auf die Variable *amount_sum* summiert.

Wird der optionale Parameter NORESET angegeben, wird die Variable *amount_sum* beim ersten Eintritt in den Scan nicht mit 0 initialisiert.

1.2 Beispiel 1

February 12, 2002
© by delight software gmbh



Beispielreport 1

```
\Scan(a)\
```

Name	Vorname	Adresse	PLZ	Ort	Land
<pre>\ScanEntry, CTS(a:id, a_count)\</pre>					
<pre>\a:name\</pre>	<pre>\a:surname\</pre>	<pre>\a:adress\</pre>	<pre>\a:zip\</pre>	<pre>\a:city\</pre>	<pre>\a:countryid\</pre>

```
\ScanFooter\
```

Es sind \a_count\ Adressen vorhanden.

```
\endscan\
```

1.3 Beispiel 2

February 12, 2002

© by delight software gmbh

\FIMG('..custom\aboutlogo.bmp')\

Beispielreport 2 – Master Detail

\scan(a)\

\a:name\ \a:surname\

\scan(b), NOEOF\

Produkt	Kunde	Investment
\scanentry\		
\b:prtypeid\	\b:customerid\	\b:investment\
\endscan\		

\endscan\

Index

- A -

Abfrage 3, 5
addieren 10, 13, 14
Addiert 10
Ausgaben 5

- B -

Berechnungen 5
Berechnungen 5
Berichte 3
Bild 6

- C -

Create 13
Currency 6

- D -

Dateiname 6
Datenbankfelder 5
Datensatz 3
Datensätze 3
Datum 9
Datumstrennzeichen 9
definieren 13
Dezimaltrennzeichen 6
dividieren 9
Dividiert 9
Double 6

- E -

E+ 6
Editor 3
EndScan 3
ersetzen 8
erstellen 13
erzeugen 13
Exponenten 6

- F -

FileName 6
Flieskommazahl 6
Float 6
Format 3
formatiert 5
Funktion 5
Funktionen 4

- G -

generieren 13
gleich 8
Grafikdatei 6
größer als 11

- H -

Hyperlink 11

- I -

inkrementieren 13, 14
integer 8

- J -

Jahr 9

- K -

kleiner als 11
Kommazahl 6
konstante 5
Konstanten 5

- L -

Ländereinstellungen 9
Layout 3
Link 11

- M -

Master-Detail 3, 4
Millisekunde 9
Minuszeichen 6

Minute 9
Mittag 9
Monat 9
Monatsnamen 9
multiplizieren 9
multipliziert 9

- N -

negativ 11
negative 6
nicht leer 13
noeof 4

- O -

optionale Parameter 5

- P -

page 4
Parameter 5
Platzhalter 6
positiv 11
positive 6
Prozent 10
prozentual 10

- R -

Reportvariablen 5
Reportvariable 5, 13
RTF 3
Rückgabewert 5
runden 6

- S -

Scan 3
Scanelement 4, 13, 14
ScanEntry 3
ScanFooter 3
Sekunde 9
setzte 12
signifikanten 6
speichern 8, 13
Strings 8
Stunde 9
subtrahieren 11

subtrahiert 11
summieren 14

- T -

Tabellen 3
Tag 9
Tausendertrennzeichen 6
Teilstrings 8
Template 3

- U -

übereinstimmen 8
Uhrzeit 9
Uhrzeittrennzeichen 9
ungleich 8
URL 11

- V -

Variablen 5
Variable 12, 13
Vorkommastellen 6
Vorlage 3
Vorlagen 3
Vorzeichen 6

- W -

Wert zuweisen 12
while 4
Wissenschaftliche Schreibweise 6
Wochentag 9
Word 3

- Z -

Zahlen 8
Zeichenfolge 8
Zeit 9
Ziffer 6
Ziffernplatzhalter 6
zwischenspeicher 8
zwischenspeichern 13